

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

**Кафедра Математического обеспечения электронно-вычислительных
машин**

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Организация ЭВМ и систем»
ТЕМА: РАЗРАБОТКА СОБСТВЕННОГО ПРЕРЫВАНИЯ.
ВАРИАНТ 1А.

Студентка гр. 0382

Рубежова Н.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Разработать программу на языке Ассемблер с использованием новой программы обработки прерываний.

Задание.

В лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка цикла задержки в вывод сообщения или включение звукового сигнала

В соответствии с выданным вариантом:

номер и назначение заменяемого вектора прерывания: *I - ICh* - прерывание от часов - генерируется автоматически операционной системой 18 раз в сек;

действия, реализуемые программой обработки прерываний: *A* - Печать сообщения на экране;

Порядок выполнения работы.

1. В сегменте данных *DATA* объявим двухбайтовые переменные *keep_ip*, *keep_cs* для хранения смещения и сегмента вектора прерывания.

2. Как сказано в примечании к лабораторной работе, под стек выделим 1024 байта или *512 dw dup(?)*.

3. Реализуем программу обработки прерываний *SUBR_INT*. Сначала сохраним изменяемые регистры, кладя их на стек. Затем, так как по условию задания необходимо вывести сообщение на экран, получим функцию *09h* печати строки на экран и вызовем прерывание *int 21h*. В конце не забудем восстановить регистры, извлекая значения со стека в соответствующие регистры. Также перед выходом из прерывания установим разрешение на вызов прерываний более низкого уровня.

4. Реализуем главную функцию *main*, с которой начинается выполнение программы. Загрузим сегмент данных. Так как программа, использующая новые программы обработки прерываний при своем завершении должна восстанавливать оригинальные векторы прерываний, получим оригинальный вектор с помощью вызова:

```

MOV AH, 35H ; функция получения вектора
MOV AL, 1CH ; номер вектора
INT 21H

```

5. Запишем полученные в *BX* и *ES* смещение и сегмент вектора прерывания в соответствующие переменные *keep_ip* и *keep_cs*.

6. Для установки адреса нового обработчика прерывания в поле векторов прерываний используем функцию 25H прерывания 21H, которая помещает заданные адреса сегмента и смещения обработчика в вектор прерывания с заданным номером.

```

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, 1CH
INT 21H ; change interrupt
POP DS

```

7. Вызовем новый обработчик прерываний, предварительно поместив в *dx* текст сообщения, которое будем выводить на экран.

```

mov dx, offset message
int 1Ch

```

8. Не забудем восстановить оригинальный вектор прерывания по значениям, которые мы хранили в *keep_ip*, *keep_cs*. И по ним восстановим старый вектор с помощью функции 25h прерывания 21h.


```

CLI
PUSH DS
MOV DX, keep_ip
MOV AX, keep_cs
MOV DS, AX
MOV AH, 25H
MOV AL, 1CH
INT 21H ; restore the old interrupt vector
POP DS
STI

```

Тестирование.

При запуске программа выводит сообщение: “Message successfully sent!”, что говорит о корректности работы программы. Результаты тестирования см. на рисунке 1.



```
C:\>main.exe  
Message successfully sent!
```

Рисунок 1 – Результаты тестирования

Вывод.

Были разработана программа, которая запоминает старый вектор прерывания, устанавливает и вызывает новый обработчик прерывания, который выводит сообщение на экран, а затем восстанавливает старый вектор прерывания.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: main.asm

```
DATA SEGMENT
    keep_cs dw 0
    keep_ip dw 0
    message DB 'Message successfully sent!$'
DATA ENDS

AStack SEGMENT STACK
    DW 512 DUP(?)
AStack ENDS

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

SUBR_INT PROC FAR
    push dx ; remember the value of changable register
    push ax;
    mov ah, 09h
    int 21h
    pop ax
    pop dx

    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, 1Ch
    INT 21H ; change interrupt
    POP DS

    ;call interrupt
```

```
    mov dx, offset message
    int 1Ch

    ;restore the old interrupt
    CLI
    PUSH DS
    MOV DX, keep_ip
    MOV AX, keep_cs
    MOV DS, AX
    MOV AH, 25H
    MOV AL, 1CH
    INT 21H ; restore the old interrupt vector
    POP DS
    STI

    ret
Main ENDP

CODE ENDS
    END Main
```