

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

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

Студентка гр. 1303

Чернуха В.В.

Преподаватель

Ефремов М.А

Санкт-Петербург

2022

Цель работы.

Разработать собственное прерывание на языке Ассемблера.

Задание.

Вариант 26.

Написать прерывание 16h – прерывание от клавиатуры. По заданному скан-коду клавиши вывести в консоль время в формате BCD (часы и минуты. Отвести программе под стек не менее 1К байт

Выполнение работы.

В сегменте данных было выделено место для переменных `keep_cs` и `keep_ip`, которые необходимы для хранения сегмента заменяемого прерывания и для хранения смещения заменяемого прерывания, так же под стек выделяется 1 Кб.

Сначала в программе сохраняется адрес старого прерывания с помощью функции `35h`, потом устанавливается адрес нового прерывания с помощью функции `25h`. Далее программа будет ждать пока пользователь введёт нужный символ на клавиатуре. Это реализовано на метке `enter_key` – в регистр `al` считывается информация из порта клавиатуры `60h` и сравнивается с `12h` (соответствует клавише `e`), если значения совпадают то вызывается прерывание, если нет, то опять происходит переход на метку `enter_key`.

В начале прерывания сначала сохраняются значения регистров до начала прерывания. Далее вызывается функция `02h` прерывания `1Ah`, после выполнения которого в `CH`, `CL`, `DH` будут лежать часы, минуты, секунды в формате BCD соответственно. Потом необходимо вывести в консоль значения часов и минут, для этого необходимо выполнить дополнительные преобразования и сопоставить цифрам их аски коды. Для этого мы последовательно делим число на основание системы счисления и копируем остатки, в `cx` сохраняем количество символов, когда делить число уже не получится со стека будут забираться значения и выводиться в консоль с

помощью функции 2 прерывания 21h (при этом к ним прибавляется „0“ для того чтобы соответствовать аски символу). Далее восстанавливаются прежние регистры и происходит возврат из прерывания iret.

Далее в main сбрасывается флаг прерывания с помощью cli, восстанавливается старый вектор прерывания и устанавливается флаг прерывания с помощью sti.

Вывод.

В результате выполнения лабораторной работы было написано собственное прерывание, которое выводит время (часы, минуты) в формате BCD.

ПРИЛОЖЕНИЕ А

Текст исходного файла программы lab5

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

```
stack segment stack
    db 1024 dup(0)
stack ends
```

```
data segment
    keep_cs dw 0
    keep_ip dw 0
data ends
```

```
code segment
```

```
SUBR_INT proc far
```

```
    push ax
    push dx
    push cx
    push bx
```

```
    mov ah, 02h
    int 1Ah
    mov ah, 0
    mov ax, cx
```

```
    mov bx, 16
    sub cx, cx
```

```
a1:
    sub dx, dx
    div bx
    push dx
    add cx, 1
    test ax, ax
    jnz a1
```

```
    mov ah, 2
```

```
a2:
    pop dx
    add dl, '0'
    int 21h
    loop a2
```

```
    pop bx
    pop cx
    pop dx
    pop ax
    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, 35h
    mov al, 16h
    int 21h
    mov keep_ip, bx
    mov keep_cs, es

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

enter_key:
    in al, 60h
    cmp al, 12h
    jne enter_key
    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

```

ПРИЛОЖЕНИЕ Б

Листинг программы lab5

#Microsoft (R) Macro Assembler Version 5.10
03:34:4

11/29/22

Page

1-1

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

0000                                stack segment stack
0000 0400[                          db 1024 dup(0)
                                ]

0400                                stack ends

0000                                data segment
0000 0000                                keep_cs dw 0
0002 0000                                keep_ip dw 0
0004                                data ends

0000                                code segment

0000                                SUBR_INT proc far
0000 50                                push ax
0001 52                                push dx
0002 51                                push cx
0003 53                                push bx

0004 B4 02                                mov ah, 02h
0006 CD 1A                                int 1Ah
0008 B4 00                                mov ah, 0
000A 8B C1                                mov ax, cx

000C BB 0010                                mov bx, 16
000F 2B C9                                sub cx, cx
0011                                a1:
0011 2B D2                                sub dx, dx
0013 F7 F3                                div bx
0015 52                                push dx
0016 83 C1 01                                add cx, 1
0019 85 C0                                test ax, ax
001B 75 F4                                jnz a1

001D B4 02                                mov ah, 2

001F                                a2:
001F 5A                                pop dx
0020 80 C2 30                                add dl, '0'
0023 CD 21                                int 21h
0025 E2 F8                                loop a2
```

```

0027 5B                pop bx
0028 59                pop cx
0029 5A                pop dx
002A 58                pop ax
002B B0 20            mov  al, 20h
002D E6 20            out  20h, al

```

```

002F CF                ired
0030                SUBR_INT endp
#Microsoft (R) Macro Assembler Version 5.10
03:34:4

```

11/29/22

Page

1-2

```

0030                main proc far
0030 1E                push ds
0031 2B C0            sub  ax, ax
0033 50                push ax
0034 B8 ---- R        mov  ax, data
0037 8E D8            mov  ds, ax

0039 B4 35            mov  ah, 35h
003B B0 16            mov  al, 16h
003D CD 21            int  21h
003F 89 1E 0002 R    mov  keep_ip, bx
0043 8C 06 0000 R    mov  keep_cs, es

0047 1E                push ds
0048 BA 0000 R        mov  dx, offset SUBR_INT
004B B8 ---- R        mov  ax, seg SUBR_INT
004E 8E D8            mov  ds, ax
0050 B4 25            mov  ah, 25h
0052 B0 16            mov  al, 16h
0054 CD 21            int  21h
0056 1F                pop  ds

0057                enter_key:
0057 E4 60            in   al, 60h
0059 3C 12            cmp  al, 12h
005B 75 FA            jne  enter_key
005D CD 16            int  16h

005F FA                cli

0060 1E                push ds
0061 8B 16 0002 R    mov  dx, keep_ip
0065 A1 0000 R        mov  ax, keep_cs
0068 8E D8            mov  ds, ax
006A B4 25            mov  ah, 25h
006C B0 16            mov  al, 16h
006E CD 21            int  21h
0070 1F                pop  ds

```

```

0071  FB          sti
0072  CB          ret
0073          main endp
0073          code ends
          end main
#Microsoft (R) Macro Assembler Version 5.10      11/29/22
03:34:4
Symb
ols-1

```

Segments and Groups:

Class	N a m e	Length	Align	Combine
CODE	0073	PARA	NONE
DATA	0004	PARA	NONE
STACK	0400	PARA	STACK

Symbols:

	N a m e	Type	Value	Attr
A1	L NEAR	0011	CODE
A2	L NEAR	001F	CODE
ENTER_KEY	L NEAR	0057	CODE
KEEP_CS	L WORD	0000	DATA
KEEP_IP	L WORD	0002	DATA
MAIN	F PROC	0030	CODE Length
= 0043				
SUBR_INT	F PROC	0000	CODE Length
= 0030				
@CPU	TEXT	0101h	
@FILENAME	TEXT	LAB5	
@VERSION	TEXT	510	

```

99 Source  Lines
99 Total   Lines
15 Symbols

```

48020 + 461287 Bytes symbol space free

0 Warning Errors

0 Severe Errors