

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

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчиков
прерываний

Студентка гр. 8383

Максимова А.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

Основные теоретические положения.

Клавиатура содержит микропроцессор, который воспринимает каждое нажатие на клавишу и посылает скан-код в порт микросхемы интерфейса с периферией. Когда скан-код поступает в порт, то вызывается аппаратное прерывание клавиатуры (int 09h). Процедура обработки этого прерывания считывает номер клавиши из порта 60h, преобразует номер клавиши в соответствующий код, выполняет установку флагов в байтах состояния, загружает номер клавиши и полученный код в буфер клавиатуры.

В прерывании клавиатуры можно выделить три основных шага:

1. Прочитать скан-код и послать клавиатуре подтверждающий сигнал.
2. Преобразовать скан-код в номер кода или в установку регистра статуса клавиш-переключателей.
3. Поместить код клавиши в буфер клавиатуры.

Текущее содержимое буфера клавиатуры определяется указателями на начало и конец записи. Расположение в памяти необходимых данных представлено в таблице.

Адрес в памяти	Размер в байтах	Содержимое
0040:001A	2	Адрес начала буфера клавиатуры
0040:001C	2	Адрес конца буфера клавиатуры
0040:001E	32	Буфер клавиатуры
0040:0017	2	Байты состояния

Рисунок 1 – Необходимые данные

Процедуры, используемые в программе.

Таблица 1 - Процедуры

Название:	Предназначение:
INT_HANDLER	Обработчик прерывания: сохраняет значение регистров в стеке для восстановления при выходе, анализирует скан-код нажатой клавиши, и в зависимости от полученного значения, либо записывает в буфер клавиатуры другое значение, либо передает управление стандартному обработчику прерывания.
CHECK	Процедура, проверяющая состояние установки (установлено или нет) пользовательского прерывания с вектором 09h.
SETTING_INTERRUPT	Процедура, устанавливающая резидентную функцию для обработки прерывания и настраивающая вектор прерывания (если прерывание не установлено)
CHECK_UNLOAD	Процедура, используемая для проверки запроса выгрузки (есть ли в хвосте командной строки "/un").

UNLOAD_INTERRUPT	Процедура, используемая для выгрузки прерывания по параметру командной строки "/un". Восстанавливает стандартный вектор прерывания и освобождает память, занимаемую резидентом.
PRINTF	Процедура печати.
BEGIN	Главная процедура.

Выполнение работы.

Шаг 1-2.

Был написан и отлажен программный модуль типа **.EXE**, который выполняет следующие функции:

- 1) Проверяет, установлено ли пользовательское прерывание с вектором 09h.
- 2) Если прерывание не установлено то, устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. Осуществляется выход по функции 4Ch прерывания int 21h.

Выполнение пунктов 1 и 2 см. на рис. 2.



```

C:\>lr5.exe
The interruption has not yet been established. Start interrupt setup.
C:\>Lr5:)_

```

Рисунок 2 – Установка прерывания (были введены символы abcde)

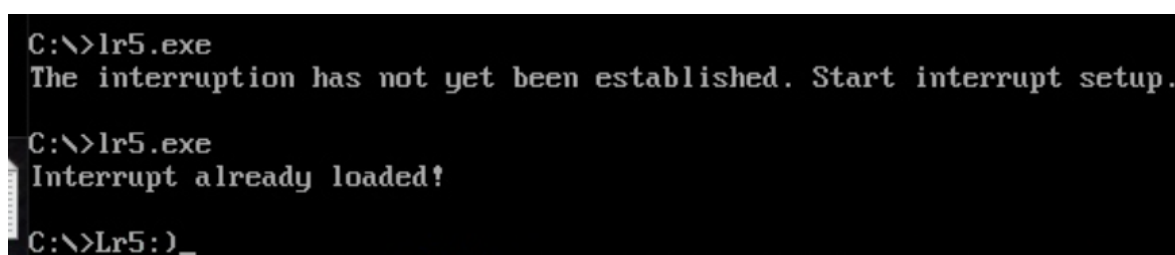
Пользовательский обработчик прерывания преобразует символы в соответствие с таблицей 2. Оставшиеся клавиши обрабатываются стандартным обработчиком прерывания.

Таблица 2 – Замена символов

Символ, который заменяется.	Символ, на который заменяется.
A, a	L
B, b	R
C, c	5
D, d	:
E, e)

- 3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

Выполнение пункт 3 см. на рис. 3.



```

C:\>lr5.exe
The interruption has not yet been established. Start interrupt setup.
C:\>lr5.exe
Interrupt already loaded!
C:\>Lr5:)_

```

Рисунок 3 – Повторный запрос установки прерывания

- 4) Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Выполнение пункт 4 см. на рис. 4.

```
C:\>lr5.exe
The interrupt has not yet been established. Start interrupt setup.

C:\>lr5.exe
Interrupt already loaded!

C:\>lr5.exe/un
Interrupt already loaded!
Interrupt unloaded!

C:\>abcde_
```

Рисунок 4 – Выгрузка прерывания

Код написанной программы см. в приложении А.

Шаг 3.

Запуск программы и проверка размещения прерывания в памяти, используя написанную ранее программу, отображающую карту памяти в виде списка блоков MCB.

```
RES5_0.TXT – Блокнот
Файл  Правка  Формат  Вид  Справка

Programm: first.
1. Amount of available memory:      648912 bytes.
2. Extended memory size:           15360  kbytes.
3. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:           0008h
Memory size:      16  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:                 0000h
Memory size:      64  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                 0040h
Memory size:      256 bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                 0192h
Memory size:      144 bytes.
Last eight bytes:
-----
MCB TYPE: 5Ah
Segment address PSP:                 0192h
Memory size: 648912 bytes.
Last eight bytes: LR3
```

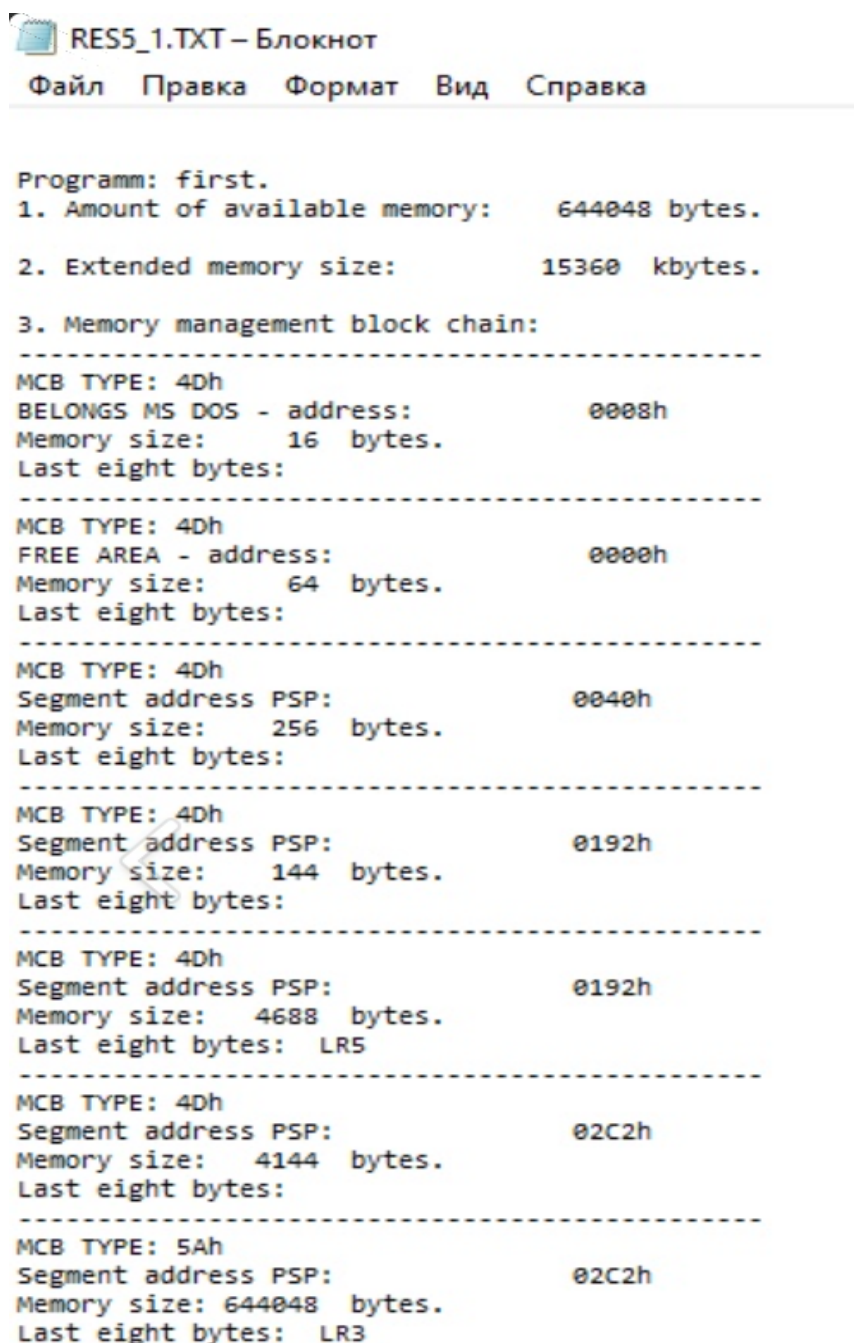
Рисунок 5 – Состояние памяти до запуска программы

```

C:\>lr5.exe
The interruption has not yet been established. Start interrupt setup.
C:\>lr3>res5_1.txt

```

Рисунок 6 – Вызов программы



RES5_1.TXT – Блокнот

Файл Правка Формат Вид Справка

```

Programm: first.
1. Amount of available memory:      644048 bytes.
2. Extended memory size:           15360  kbytes.
3. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:           0008h
Memory size:      16  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:                0000h
Memory size:       64  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0040h
Memory size:     256  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0192h
Memory size:     144  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0192h
Memory size:    4688  bytes.
Last eight bytes: LR5
-----
MCB TYPE: 4Dh
Segment address PSP:                02C2h
Memory size:    4144  bytes.
Last eight bytes:
-----
MCB TYPE: 5Ah
Segment address PSP:                02C2h
Memory size:   644048 bytes.
Last eight bytes: LR3

```

Рисунок 7 – блоки MCB, где 4 и 5 принадлежат LR5

Шаг 4.

```
C:\>lr5.exe
Interrupt already loaded!
C:\>lr3>res5_2.txt
```

Рисунок 8 – Повторный запуск программы

```
RES5_2.TXT – Блокнот
Файл  Правка  Формат  Вид  Справка

Programm: first.
1. Amount of available memory:      644048 bytes.
2. Extended memory size:           15360  kbytes.
3. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:           0008h
Memory size:      16  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:                0000h
Memory size:      64  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0040h
Memory size:     256  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0192h
Memory size:     144  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0192h
Memory size:    4688  bytes.
Last eight bytes: LR5
-----
MCB TYPE: 4Dh
Segment address PSP:                02C2h
Memory size:    4144  bytes.
Last eight bytes:
-----
MCB TYPE: 5Ah
Segment address PSP:                02C2h
Memory size:   644048 bytes.
Last eight bytes: LR3
```

Рисунок 9 – блоки МСВ, где 4 и 5 принадлежат программе, запущенной повторно

Шаг 5.

```
C:\>lr5.exe/un
Interrupt already loaded!
Interrupt unloaded!

C:\>lr3>res5_3.txt

C:\>abcdefgh
```

Рисунок 10 – Вызов программы с ключом выгрузки

Programm: first.

1. Amount of available memory: 648912 bytes.
2. Extended memory size: 15360 kbytes.
3. Memory management block chain:

MCB TYPE: 4Dh
BELONGS MS DOS - address: 0008h
Memory size: 16 bytes.
Last eight bytes:

MCB TYPE: 4Dh
FREE AREA - address: 0000h
Memory size: 64 bytes.
Last eight bytes:

MCB TYPE: 4Dh
Segment address PSP: 0040h
Memory size: 256 bytes.
Last eight bytes:

MCB TYPE: 4Dh
Segment address PSP: 0192h
Memory size: 144 bytes.
Last eight bytes:

MCB TYPE: 5Ah
Segment address PSP: 0192h
Memory size: 648912 bytes.
Last eight bytes: LR3

Рисунок 11 – Карта памяти после выгрузки



Рисунок 12 – Дополнительный пример работы обработчика прерывания

Ответы на контрольные вопросы.

1) Какого типа прерывания использовались в работе?

09h	Аппаратное прерывание от клавиатуры
16h	Программное прерывание (прерывание BIOS)
21h	Программное прерывание (прерывание DOS)

2) Чем отличается скан-код от кода ASCII?

ASCII код – код в кодировочной таблице ASCII, используемый для представления и хранения в памяти символа (десятичных цифр, латинского и национального алфавита, а также знаков препинания и управляющих символов) в виде числа, а также для его печати на экран.

Скан-код – код, присвоенный каждой клавише, используемый драйвером клавиатуры для распознавания того, какая конкретно клавиша была нажата.

Выводы.

В результате выполнения лабораторной работы была изучена возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Была написана программа, реализующая сопряжение обработчиков, способная загружать и выгружать прерывание от клавиатуры.

ПРИЛОЖЕНИЕ А

Содержимое файлы LR5.ASM

COMMENT *

Максимова Анастасия, группа 8383 - лабораторная 5

*

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack, ES:NOTHING

;-----

INT_HANDLER PROC FAR ;обработчик
прерываний

JMP HANDLER_START

SIGNATURE DW 4321h ;для
проверки

KEEP_AX DW 0

KEEP_SS DW 0

KEEP_SP DW 0

KEEP_IP DW 0

KEEP_CS DW 0

KEEP_PSP DW 0

REPL_CHAR DB 0

CHAR_A DB 1Eh ;скан-
коды

CHAR_B DB 30h

CHAR_C DB 2Eh

CHAR_D DB 20h

CHAR_E DB 12h

;Чтобы иметь возможность работать со стеком
уменьшенного размера,

;каждому обработчику прерывания выделяется свой
стек, отдельный для каждого процессора.

INT_HANDLER_Stack DW 64 dup(?) ;стек
прерывания

;-----

HANDLER_START:

mov KEEP_SS, SS

; "переключаемся на стек прерывания"

mov KEEP_SP, SP

mov KEEP_AX, AX

mov AX, SEG INT_HANDLER_Stack

mov SS, AX

mov AX, OFFSET INT_HANDLER_Stack

add AX, 128 ;64*2

mov SP, AX ;SP

устанавливается на конец стека

;-----

;сохранение изменяемых регистров

push ES

push CX

push DS

;-----

mov AX, SEG REPL_CHAR

mov DS, AX

;замена: abcde-->1r5:)

;

действия по обработке прерывания 09h

in AL, 60h ;

получение скан-кода последней нажатой клавиши

```
cmp     AL, CHAR_A
jne     OTHER1
mov     REPL_CHAR, 'L'
jmp     HANDWARE_INTERR
```

OTHER1:

```
cmp     AL, CHAR_B
jne     OTHER2
mov     REPL_CHAR, 'r'
jmp     HANDWARE_INTERR
```

OTHER2:

```
cmp     AL, CHAR_C
jne     OTHER3
mov     REPL_CHAR, '5'
jmp     HANDWARE_INTERR
```

OTHER3:

```
cmp     AL, CHAR_D
jne     OTHER4
mov     REPL_CHAR, ':'
jmp     HANDWARE_INTERR
```

OTHER4:

```
cmp     AL, CHAR_E
jne     CONTINUE
mov     REPL_CHAR, ')'
jmp     HANDWARE_INTERR
```

CONTINUE:

```
pushf
call DWORD PTR CS:KEEP_IP
jmp     EXIT
```

HANDWARE_INTERR:

;

обработка аппаратного прерывания

;

61h - регистр управления клавиатурой

;

установка 7 бита порта 61h и возвращение исходного состояния

```
in      AL, 61h
```

;

взять значение порта управления клавиатурой

```
mov     AH, AL
```

;

сохраняем

```
or      AL, 80h
```

;

установить бит разрешения для клавиатуры

```
out     61h, AL
```

;

вывести его в управляющий порт

```
xchg    AH, AL
```

;

извлечь исходное значение порта

```
out     61h, AL
```

;

и

записать его обратно

```
mov     AL, 20h
```

;

послать сигнал "конец прерывания"

```
out     20h, AL
```

;

контроллеру прерываний 8259

;-----

CHARACTER_BUFFER:

```
mov     AH, 05h
```

;

запись символа в буфер клавиатуры

```

                                mov     CL, REPL_CHAR           ; CL - символ
ASCII
                                mov     CH, 00h                 ;
CH - скан-код
                                int     16h
                                or      AL, AL                 ;
проверка переполнения буфера
                                JNZ     SKIP                   ; если
переполнение
                                JMP     EXIT

                                SKIP:
                                mov     AX, 0040h
                                mov     ES, AX
                                mov     AX, ES:[1Ah]
                                ;0040:001Ah - указатель на начало
                                mov     ES:[1Ch], AX           ;0040:001Ch -
указатель на конец буфера
                                jmp     CHARACTER_BUFFER       ;повтор
                                ;-----
EXIT:
;восстановление регистров
                                pop     DS
                                pop     CX
                                pop     ES
                                ;-----
                                mov     AX, KEEP_SS
                                ;"переключаемся на внешний стек"
                                mov     SS, AX

                                mov     SP, KEEP_SP
                                mov     AX, KEEP_AX
                                ;-----

```



```

                                mov     AL, 20h                ;выход
                                out     20h, AL
                                iret                          ;popf + retf -
возврат из прерывания

INT_HANDLER     ENDP
LAST_BYTE:
;-----
CHECK          PROC NEAR                                ;1)проверяет,
установлено ли

;пользовательское прерывание с вектором 1Ch
                                push    AX
                                push    BX
                                push    DI
                                push    ES

;чтение адреса, записанного в векторе прерывания
                                mov     AX, 3509h            ; AH - 35h -
считать адрес обработчика прерываний
                                                                ; AL =
09h - номер прерывания
                                int     21h                  ; ES:BX =
адрес обработчика прерывания

                                mov     DI, OFFSET SIGNATURE ;смещение
сигнатуры относительно
                                sub     DI, OFFSET INT_HANDLER ;начала
обработчика прерывания

                                mov     AX, ES:[BX + DI]
                                cmp     AX, SIGNATURE        ;если
совпадают, значит резидент установлен
                                jne     END_CHECK

```

```

mov          CHECK_flag, 0

;изменяем флаг - 0 - если установлен

END_CHECK:

pop          ES
pop          DI
pop          BX
pop          AX

retn

CHECK        ENDP

;-----
;2 задание: установить резидентную функцию для обработки прерывания
;настроить вектор прерываний - если не установлен
;осуществить выход по функции 4Ch int21h
SETTING_INTERRUPT    PROC NEAR                                ;установка
прерывания

push AX
push BX
push CX
push DX
push ES
push DS

;запоминаем адрес предыдущего обработчика

mov          AX, 3509h                                ;AH =
35h - считать адрес обработчика прерываний

;AL =

09h - прерывание

int          21h

```

```

mov        KEEP_IP,          BX
;запоминаем смещение

mov        KEEP_CS, ES
;запоминаем сегментный адрес

push DS
mov        DX, SEG INT_HANDLER
;сегментный адрес

mov        DS, DX           ;B
DS

mov        DX, OFFSET INT_HANDLER
;смещение в DX

mov        AX, 2509h        ;AH =
25h - установить адрес обработчика прерывания
int        21h

pop        DS

;оставить процедуру резидентной в памяти
mov        DX, OFFSET LAST_BYTE
;определение размера

;резидентной части программы Fh для округления вверх
mov        CL,              4h
;деление на 16

shr        DX, CL           ;B
параграфах

add        DX, 10Fh
inc        DX

xor        AX, AX
mov        AH,              31h
;оставить программу резидентной

```

```

                                int      21h

                                pop       DS
                                pop       ES
                                pop       DX
                                pop       CX
                                pop       BX
                                pop       AX

                                retn

SETTING_INTERRUPT      ENDP

;-----
CHECK_UNLOAD      PROC NEAR                                ;проверка
есть ли запрос на выгрузку

                                push  ES
                                push  AX

;проверяем хвост командной строки 0081h..

                                mov      AX, KEEP_PSP
                                mov      ES, AX

                                cmp      BYTE PTR ES:[0082h], '/'
                                je        NEXT1
                                jmp      EXIT_

NEXT1:

                                cmp      BYTE PTR ES:[0083h], 'u'
                                je        NEXT2
                                jmp      EXIT_

NEXT2:

                                cmp      BYTE PTR ES:[0084h], 'n'
                                je        CHANGE__

```

```

                                jmp          EXIT_

CHANGE__:
                                mov          CHECK2_flag, 0

EXIT_:
                                pop          AX
                                pop          ES
                                retn

CHECK_UNLOAD                    ENDP
;-----
UNLOAD_INTERRUPT                PROC NEAR                                ;выгрузка
обработчика прерываний

                                CLI

;запретить прерывания

                                push AX

;сохранение регистров

                                push BX
                                push DX
                                push ES
                                push DS
                                push DI

                                mov          AX, 3509h                    ;AH =
35h - считать адрес обработчика прерываний

                                ;AL =

09h - прерывание

                                int          21h

                                mov          DI, OFFSET KEEP_IP
                                sub          DI, OFFSET INT_HANDLER

```

```

mov     DX, ES:[BX + DI]
add     DI, 2
mov     AX, ES:[BX + DI]
add     DI, 2

push    DS
mov     DS, AX
mov     AX, 2509h
int     21h

;восстановление вектора

pop     DS

mov     AX, ES:[BX + DI]
mov     ES, AX

push    ES
mov     AX, ES:[2Ch]
mov     ES, AX

mov     AH, 49h

;Освободить распределенный блок памяти
int     21h

;ES = сегментный адрес (параграф) освобождаемого блока памяти

pop     ES
mov     AH, 49h
int     21h

pop     DI
pop     DS
pop     ES
pop     DX
pop     BX

```

```

                                pop        AX

                                STI

;Разрешение аппаратных прерываний

                                retn

UNLOAD_INTERRUPT             ENDP
;-----
PRINTF      PROC  NEAR

                                push    AX
                                mov     AH, 09h
                                int     21h
                                pop     AX

                                retn

PRINTF      ENDP
;-----
BEGIN       PROC  FAR

                                push    DS
                                xor     AX, AX
                                push    AX
                                mov     AX, DATA
                                mov     DS, AX

                                mov     KEEP_PSP, ES

                                call    CHECK
                                cmp     CHECK_flag, 0
;если не равен 0, устанавливаем
                                jne     CASE1
                                jmp     CASE2

```

CASE1:

;прерывание не было установлено и его нужно установить

```
call    CHECK_UNLOAD
cmp     CHECK2_flag, 0
je      CASE4
```

```
mov     DX, OFFSET SMS1
call    PRINTF
```

прерывания

```
call    SETTING_INTERRUPT                ;установка
```

```
jmp     END_BEGIN
```

CASE2:

;прерывание уже загружено

```
mov     DX, OFFSET SMS2
call    PRINTF
```

```
call    CHECK_UNLOAD
cmp     CHECK2_flag, 0
```

```
je      CASE3
jmp     END_BEGIN
```

CASE3:

```
call    UNLOAD_INTERRUPT
mov     DX, OFFSET SMS3
call    PRINTF
jmp     END_BEGIN
```

CASE4:

;не было загружено -- не может быть выгружено

```
mov     DX, OFFSET SMS4
call    PRINTF
```



```

END_BEGIN:
                xor            AL, AL
;выход в DOS
                mov            AH, 4Ch
                int            21h

BEGIN          ENDP
;-----
CODE ENDS

AStack SEGMENT STACK
            DW 128 dup(?)
AStack ENDS

DATA        SEGMENT                                ;ДАННЫЕ
EOF          EQU            '$'
SMS1          DB            'The interruption has not yet been
established. Start interrupt setup.', 0DH, 0AH,      EOF
                                ;Прерывание еще не было установлено. Запуск
установки прерывания.
SMS2          DB            'Interrupt already loaded!', 0DH, 0AH,
EOF          ;прерывание уже загружено
SMS3          DB            'Interrupt unloaded!', 0DH, 0AH,
EOF          ;прерывание выгружено
SMS4          DB            'The interrupt cannot be unloaded,
because it is not set.', 0DH, 0AH,      EOF
                                ;прерывание не может быть выгружено, так как
оно не установлено.

                                ;flags
CHECK_flag    DB            1
CHECK2_flag   DB            1
DATA          ENDS

```

END

BEGIN