

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

**ОТЧЕТ**  
**по лабораторной работе №4**  
**по дисциплине «Операционные системы»**  
**Тема: Обработка стандартных прерываний**

Студентка гр. 8383

\_\_\_\_\_

Максимова А.А.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Построить обработчик прерываний сигналов таймера. Написанная программа должна устанавливать резидентную функцию для обработки прерывания.

### **Основные теоретические положения.**

Резидентный обработчики прерываний - это программные модули, которые вызываются при возникновении прерываний определенного типа (сигнал таймера, нажатие клавиши и т.д.), которым соответствуют определенные вектора прерываний. Когда вызывается прерывание, процессор переключается на выполнение кода обработчика, а затем возвращается на выполнение прерванной программы. Адрес возврата в прерванную программу (CS:IP) запоминается в стеке вместе с регистром флагов. Затем в CS:IP загружается адрес точки входа программы обработки прерывания и начинает выполнять его код. Обработчик прерывания должен заканчиваться инструкцией IRET (возврат из прерывания).

Вектор прерывания имеет длину 4 байта. В первом хранится значение IP, во втором - CS. Младшие 1024 байта памяти содержат 256 векторов. Вектор для прерывания 0 начинается с ячейки 0000:0000, для прерывания 1 - с ячейки 0000:0004 и т.д.

## Процедуры, используемые в программе.

Таблица 1 - Процедуры

| Название:         | Предназначение:   |
|-------------------|---|
| INT_HANDLER       | Обработчик прерывания, используемый для обработки прерывания: вывод по таймеру количество вызовов прерывания  |
| OUTPUT_BP         | Процедура, предназначенная для вывода строки по адресу ES:BP на экран   |
| SET_CURS          | Процедура, используемая для установки позиции курсора   |
| GET_CURS          | Процедура, используемая для получения позиции курсора   |
| CHECK             | Процедура, проверяющая состояние установки (установлено или нет) пользовательского прерывания с вектором 1Ch  |
| 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  |

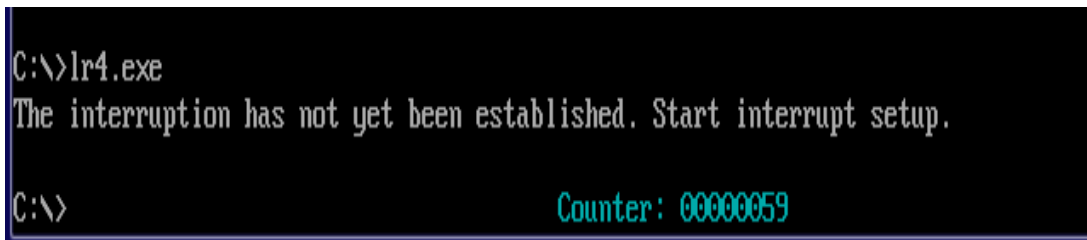
## Выполнение работы.

### Шаг 1.

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

1. Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
2. Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.

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

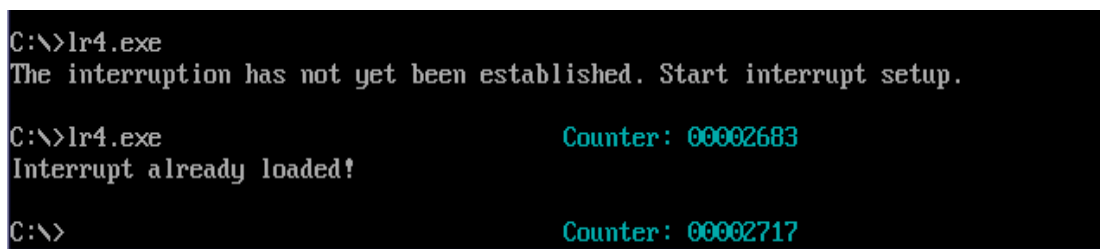


```
C:\>lr4.exe
The interruption has not yet been established. Start interrupt setup.
C:\>                                     Counter: 00000059
```

Рисунок 1 - Установка прерывания

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

Выполнение пункта 3 см. на рис. 2.



```
C:\>lr4.exe
The interruption has not yet been established. Start interrupt setup.
C:\>lr4.exe                                     Counter: 00002683
Interrupt already loaded!
C:\>                                             Counter: 00002717
```

Рисунок 2 - 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 представлено на рис. 3.

```
C:\>lr4.exe
The interruption has not yet been established. Start interrupt setup.

C:\>lr4.exe                               Counter: 00002683
Interrupt already loaded!

C:\>lr4.exe/un                             Counter: 00005961
Interrupt already loaded!
Interrupt unloaded!
```

Рисунок 3 - Выгрузка прерывания

## Шаг 2.

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

```
C:\>lr4.exe
The interruption has not yet been established. Start interrupt setup.

C:\>lr3_1.com>res1.txt                      Counter: 00000363

C:\>                                          Counter: 00000403
```

Рисунок 4 - Вызов программ

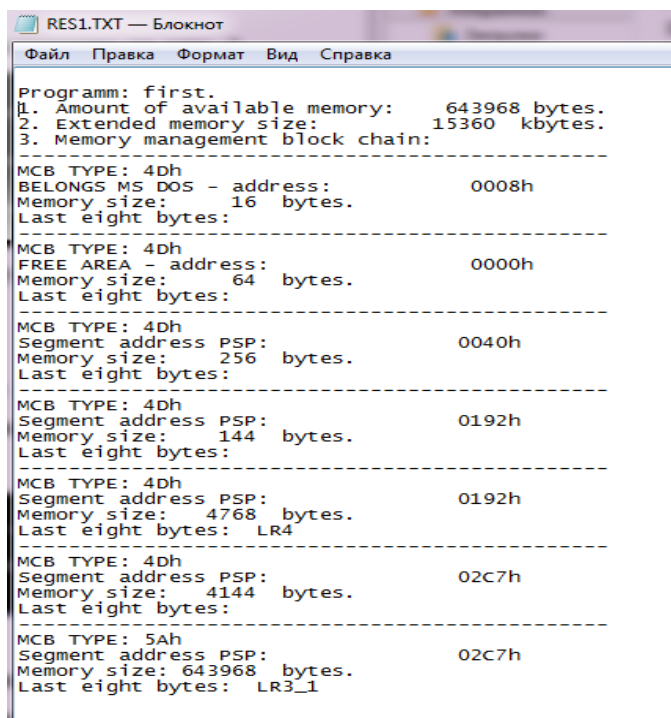


Рисунок 5 - блоки MCB, где четвертый и пятый, принадлежат данной программе

### Шаг 3.

Повторный запуск программы.

```
C:\>lr4.exe
The interrupt has not yet been established. Start interrupt setup.

C:\>lr3_1.com>res1.txt                               Counter: 00000363

C:\>lr4.exe                                           Counter: 00008795
Interrupt already loaded!

C:\>lr3_1.com>res2.txt                               Counter: 00009079

C:\>                                                  Counter: 00009120
```

Рисунок 6 - Повторный вызов программ

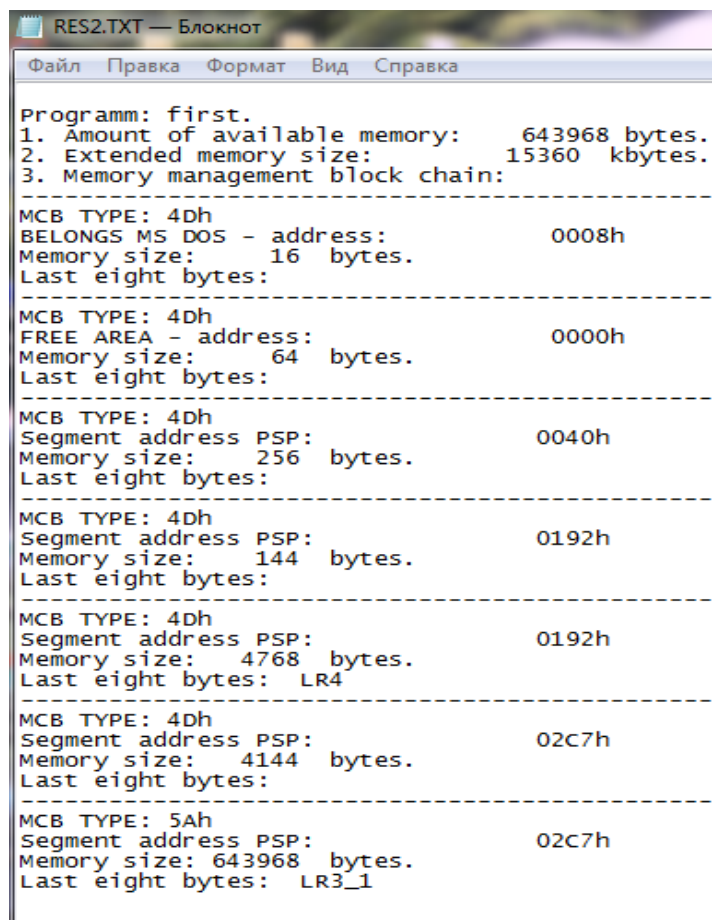


Рисунок 7 - блоки МСВ, где четвертый и пятый, принадлежат программе, запущенной повторно

#### Шаг 4.

Запуск программы с ключом выгрузки и проверка карты памяти после этого (произошло ли освобождение памяти).

```
C:\>lr4.exe
The interruption has not yet been established. Start interrupt setup.

C:\>lr3_1.com>res1.txt                               Counter: 00000363

C:\>lr4.exe                                           Counter: 00008795
Interrupt already loaded!

C:\>lr3_1.com>res2.txt                               Counter: 00009079

C:\>lr4.exe/un                                       Counter: 00016258
Interrupt already loaded!
Interrupt unloaded!

C:\>lr3_1.com>res3.txt
```

Рисунок 8 - Вызов программы с ключом выгрузки

```
RES3.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_1
```

Рисунок 9 - Карта памяти после выгрузки

## **Шаг 5.**

### **Ответы на контрольные вопросы.**

#### **1) Как реализован механизм прерывания от часов?**

Персональные компьютеры содержат два устройства для управления процессами, одно из которых - системный таймер, который используется для управления контроллером прямого доступа и динамиком, и как генератор импульсов, вызывающий прерывание IRQ0 (int 8h) 18,2 раза в секунду.

При инициализации BIOS устанавливает стандартный обработчик для прерывания таймера, увеличивающий каждый раз на единицу четырехбайтовую переменную, располагающуюся в области данных BIOS по адресу 0000:0046Ch (счетчик тиков таймера). При переполнении счетчика, в ячейку 0000:0470h заносится единица.

Кроме этого обработчик прерывания таймера вызывает прерывание int 1Ch. После инициализации системы вектор int 1Ch указывает на команду IRET - ничего не выполняется, поэтому программа может установить собственный обработчик прерывания для выполнения каких-либо периодических действий.

Прерывание int 1Ch вызывается обработчиком прерывания int 8h до сброса контроллера прерывания, поэтому во время выполнения прерывания int 1Ch все аппаратные прерывания запрещены.

После завершения обработки прерывания, происходит восстановление регистров, значения которых заранее сохраняются при вызове прерывания, восстанавливаются значения флагов, происходит возврат управления прерванной программе, разрешается выполнение прерываний с более низким приоритетом.



## 2) Какого типа прерывания использовались в работе ?

|     |   |
|-----|---|
| 1Ch | Аппаратное (асинхронное) прерывание. (Пользовательское прерывание по таймеру) |
| 10h | Программное прерывание (прерывание BIOS)                                      |
| 21h | Программное прерывание (прерывание DOS)                                       |

### Выводы.

В результате выполнения лабораторной работы был построен собственный резидентный обработчик прерывания от системного таймера, а также было произведено восстановление стандартного вектора прерываний и освобождение памяти.

## ПРИЛОЖЕНИЕ А

Содержимое файлы LR4.ASM

COMMENT \*

Максимова Анастасия, группа 8383 - лабораторная 4

\*

CODE      SEGMENT

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

;-----

INT\_HANDLER      PROC FAR      ;обработчик  
                 прерываний

                 JMP      HANDLER\_START

                 EOF      EQU    '\$'

                 SETPR      EQU    16

                 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

                 COUNTER\_      DB      'Counter: 00000000', 0DH,  
0AH, EOF

                 ;Чтобы иметь возможность работать со стеком уменьшенного  
размера,

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

```
INT_HANDLER_Stack    DW    128 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, 256                      ;128*2  
  
        mov        SP, AX                      ;SP  
устанавливается на конец стека  
;-----
```

;сохранение изменяемых регистров

```
push BX  
push CX  
push DX  
push DI  
push DS  
push ES  
push BP  
  
;-----
```

;действия по обработке прерывания

```
call GET_CURS
```

```

                push    DX                                ;курсор для
восстановления

                mov     DX, 1828h                        ;строка - столбец
                call    SET_CURS

                push    DS
                mov     AX, SEG COUNTER_
                mov     DS, AX

                mov     DI, OFFSET COUNTER_              ;строка для
счетчика
                add     DI, SETPR                        ;в конец
строки

                mov     CX, 8

CALCULATED:
                ;обработка счетчика

                mov     AH, [DI]
                inc     AH
                mov     [DI], AH

                cmp     AH, ':'                            ;ASCII символ :
идет после 9ки
                je      NEXT_RANK
                jmp     EXIT

NEXT_RANK:
                mov     AH, '0'
                mov     [DI], AH
                dec     DI
                loop    CALCULATED                        ;CX != 0

EXIT:

```

```

        pop            DS
        mov            AX, SEG COUNTER_

        push ES
        mov            ES, AX

        push BP
        mov            BP, OFFSET COUNTER_

        call OUTPUT_BP

        pop            BP
        pop            ES

        pop            DX
        call SET_CURS

        ;-----
        pop            BP
;восстановление регистров
        pop            ES
        pop            DS
        pop            DI
        pop            DX
        pop            CX
        pop            BX
        ;-----

        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

;ret
;-----
INT_HANDLER     ENDP
LAST_BYTE:
;-----
OUTPUT_BP      PROC NEAR                                ; функция вывода
строки по адресу ES:BP на экран

; CX = длина
строки (подсчитываются только символы)

mov     AX, 1301h                                ; AL = код
подфункции 1 = использовать атрибут в BL; курсор - в конец строки

mov     BX, 0003h                                ; BH = номер
страницы BL - цвет

mov     CX, 17                                    ; число символов для
записи

int     10h

retn
OUTPUT_BP      ENDP
;-----
SET_CURS      PROC NEAR                                ;установка позиции
курсора

;на строку 25 -
невидимый

mov     AH, 02h
mov     BH, 0h                                    ;видео
страница

```

```

                                int      10h

                                retn

SET_CURS      ENDP
;-----
GET_CURS      PROC NEAR                                ;читать позицию и размер
    курсора                                           ;на строку 25 -
                                                    ;невидимый
                                mov      AH, 03h
                                mov      BH, 0h                                ;видео
    страница
                                int      10h
                                                    ;ВЫХОД -
    DH,DI - текущие строка и колонка
                                                    ;CH, CL
    - текущие начальная, конечная строки курсора
                                retn
GET_CURS      ENDP
;-----
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, 351Ch                                ; AH - 35h - считать
    адрес обработчика прерываний
                                                    ; AL = 1Ch -
    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      CHANGE_
                jmp      END_CHECK

CHANGE_:                      ; изменяем флаг - 0 -
    прерывание не установлено
                mov      CHECK_flag, 0

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, 351Ch           ;AH = 35h -
считать адрес обработчика прерываний
                                   ;AL = 1Ch -
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             ;в DS
mov     DX, OFFSET INT_HANDLER ;смещение в DX

mov     AX, 251Ch          ;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

```

```

16      mov      CL, 4h                ;деление на
      shr      DX, CL                ;в
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, 351Ch                ;AH = 35h -
считать адрес обработчика прерываний
                                                                ;AL = 1Ch -
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, 251Ch
                                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  NEAR

```

```

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

```

```

        mov     DS, AX

        mov     KEEP_PSP, ES

        call    CHECK
        cmp     CHECK_flag, 0
        je      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

```

```

        call    SETTING_INTERRUPT           ;установка
        прерывания
        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

```