

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

Студент гр. 9382

Юрьев С.Ю.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Построить обработчик прерываний сигналов таймера.

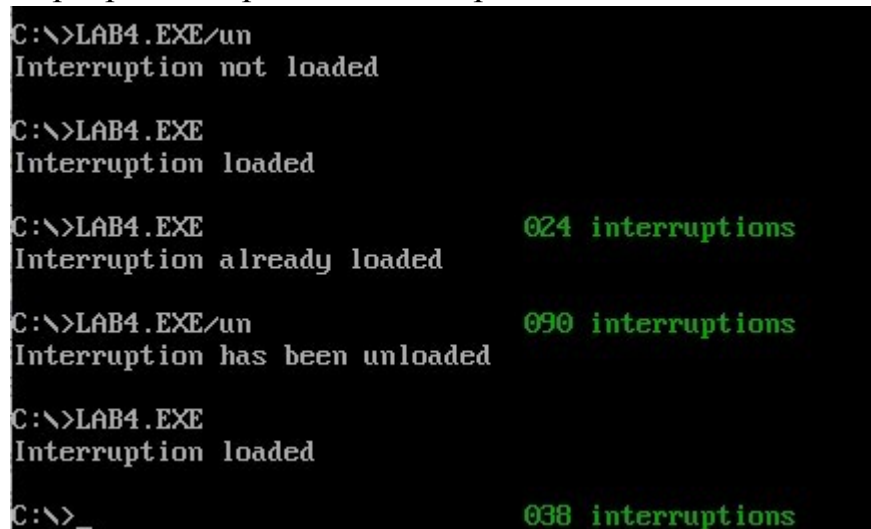
Ход работы.

Был написан и отлажен программный модуль lab4.exe.

Процедуры, используемые в программе:

- 1) ROUT - процедура обработчика прерываний.
- 2) CHECK - процедура проверки установки резидента.
- 3) LOADP - процедура загрузки резидента.
- 4) UNLOADP - процедура выгрузки резидента.
- 5) CHECKUN - процедура проверки наличия “/un”.
- 6) PRINT - вывод строки из DX на экран.

Вывод программы представлен на рис. 1.



```
C:\>LAB4.EXE/un
Interruption not loaded

C:\>LAB4.EXE
Interruption loaded

C:\>LAB4.EXE                                024 interruptions
Interruption already loaded

C:\>LAB4.EXE/un                                090 interruptions
Interruption has been unloaded

C:\>LAB4.EXE
Interruption loaded

C:\>_                                           038 interruptions
```

Рис. 1. Результат выполнения lab4.exe.

По рисунку видно, что при первом запуске программа сообщает о загрузке, при повторном запуске - о том, что загрузка уже была выполнена. При запуске с параметром “/un” - о том, что выгружена, если была загружена, и о том, что не была загружена при соответствующих событиях.

Вывод программы lab3_1.com после запуска lab4.exe представлен на рис. 2.

```
C:\>LAB4.EXE
Interruption loaded

C:\>LAB3_1.COM                                     394 interruptions
Available memory: 644256 b
Extended memory: 15360 kb
MCB: 1 Owner: MS DOS                               Size: 16
last 8 bytes:
MCB: 2 Owner: free                                  Size: 64
last 8 bytes:
MCB: 3 Owner: 0040                                  Size: 256
last 8 bytes:
MCB: 4 Owner: 0192                                  Size: 144
last 8 bytes:
MCB: 5 Owner: 0192                                  Size: 4480
last 8 bytes: LAB4
MCB: 6 Owner: 02B5                                  Size: 4144
last 8 bytes:
MCB: 7 Owner: 02B5                                  Size: 644256
last 8 bytes: LAB3_1
```

Рис. 2. Результат выполнения lab3_1.exe после запуска lab4.exe.

По рисунку видно, что процедура прерывания осталась резидентной в памяти и располагается в пятом блоке.

Вывод программы lab3_1.com после запуска lab4.exe/un представлен на рис. 3.

```

C:\>LAB4.EXE
Interruption loaded

C:\>LAB4.EXE/un                                680 interruptions
Interruption has been unloaded

C:\>LAB3_1.COM
Available memory: 648912 b
Extended memory: 15360 kb
MCB:  1  Owner:  MS DOS                Size:    16
last 8 bytes:
MCB:  2  Owner:  free                  Size:    64
last 8 bytes:
MCB:  3  Owner:  0040                  Size:   256
last 8 bytes:
MCB:  4  Owner:  0192                  Size:   144
last 8 bytes:
MCB:  5  Owner:  0192                  Size: 648912
last 8 bytes: LAB3_1

```

Рис. 3. Результат выполнения lab3_1.exe после запуска lab4.exe/un.

По рисунку видно, что память для резидентного обработчика была освобождена.

Ответы на контрольные вопросы

1) Как реализован механизм прерывания от часов?

Ответ: когда происходит сигнал часов, то сохраняется состояние регистров для того, чтобы вернуться в текущую программу, затем определяется источник прерывания, из вектора прерывания считываются CS и IP обработчика прерывания, прерывание обрабатывается, затем управление возвращается прерванной программе

2) Какого типа прерывания использовались в работе?

Ответ: аппаратные(1Ch) и программные(int 10h, int 21h)

Выводы.

В ходе выполнения данной работы была реализована программа, загружающая и выгружающая прерывание от системного таймера в память.

ПРИЛОЖЕНИЕ А.

Исходный код программы.

```
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack
; Процедуры
;-----
ROUT PROC FAR
; обработчик прерываний
    jmp starttrout
routdata:
    counter        DB    '000 interruptions'
    signature       DW    2910h

    keep_ss         dw    ?
    keep_sp         dw    ?
    keep_ax         dw    ?

    KEEP_IP        DW    0
    KEEP_CS        DW    0
    KEEP_PSP       DW    0

    rout_stack     dw    16    dup(?)
    end_stack      dw    ?

starttrout:
    mov    keep_ss, ss
    mov    keep_sp, sp
    mov    keep_ax, ax
    mov    ax, seg    rout_stack
    mov    ss, ax
    mov    sp, offset end_stack

    PUSH AX ; сохранение изменяемых регистров
    push bx
    push cx
    push dx
    push si
    push es
    push ds

    mov    ax, seg counter
    mov    ds, ax

    mov    AH, 03h
    mov    BH, 0h
    int    10h ; получение позиции курсора
; выход: DH,DL = текущие строка, колонка курсора
; CH,CL = текущая начальная, конечная строки курсора
    push dx

    mov    ah, 02h
```

```

    mov  bh, 0h
    mov  dx, 1820h
    int  10h          ; установка курсора

    mov  ax, seg counter
    push ds
    mov  ds, ax
    mov  si, offset counter
    add  si, 2
    mov  cx, 3
cycle:
    mov  ah, [si]
    inc  ah
    mov  [si], ah
    cmp  ah, ':'
    jne  endc
    mov  ah, '0'
    mov  [si], ah
    dec  si
    loop cycle
endc:
    pop  ds

; print
    push es
    push bp
    mov  ax, seg counter
    mov  es, ax
    mov  bp, offset counter
    mov  ah, 13h
    mov  al, 1h
    mov  bl, 2h
    mov  bh, 0
    mov  cx, 17
    int  10h          ; вывод

    pop  bp
    pop  es

    pop  dx          ; восстановление курсора
    mov  ah, 02h
    mov  bh, 0h
    int  10h

    pop  ds
    pop  es
    pop  si
    pop  dx
    pop  cx
    pop  bx
    POP  AX          ; восстановление регистров

    mov  ax, keep_ax
    mov  ss, keep_ss

```

```

        mov    sp, keep_sp

        MOV AL, 20H
        OUT 20H,AL
        IRET
ROUT ENDP
;-----
last_byte:
;-----
CHECK    proc
        push ax
        push bx
        push si

        MOV AH, 35H ; функция получения вектора
        MOV AL, 1CH ; номер вектора
        INT 21H
        mov    si, offset signature
        sub     si, offset rout
        mov     ax, es:[bx+si]
        cmp     ax, signature
        jne     endcheck
        mov     loaded, 1

endcheck:
        pop si
        pop     bx
        pop     ax
        ret
CHECK    endp
;-----
LOADP    proc
        push ax
        push bx
        push cx
        push dx
        push es
        push ds

        MOV AH, 35H ; функция получения вектора
        MOV AL, 1CH ; номер вектора
        INT 21H
        MOV KEEP_IP, BX ; запоминание смещения
        MOV KEEP_CS, ES ; и сегмента
        MOV DX, OFFSET ROUT ; смещение для процедуры в DX
        MOV AX, SEG ROUT    ; сегмент процедуры
        MOV DS, AX          ; помещаем в DS
        MOV AH, 25H         ; функция установки вектора
        MOV AL, 1CH         ; номер вектора
        INT 21H             ; меняем прерывание
        POP DS
        mov DX,offset LAST_BYTE ; размер в байтах от начала
        mov CL,4              ; перевод в параграфы
        shr DX,CL

```

```

    add    dx, 10fh
    inc DX                                ; размер в параграфах
    mov AH,31h
    int 21h

    pop es
    pop    dx
    pop    cx
    pop    bx
    pop    ax
    ret
LOADP      endp
;-----
UNLOADP    proc
    cli
    push ax
    push bx
    push dx
    push ds
    push es
    push si

    MOV AH, 35H ; функция получения вектора
    MOV AL, 1CH ; номер вектора
    INT 21H
    mov  si, offset keep_ip
    sub  si, offset rout
    mov  dx, es:[bx+si]
    mov  ax, es:[bx+si+2]
    push ds
    mov  ds, ax
    MOV AH, 25H ; функция установки вектора
    MOV AL, 1CH ; номер вектора
    INT 21H     ; меняем прерывание
    POP DS
    mov  ax, es:[bx+si+4]
    mov  es, ax
    push es
    mov  ax, es:[2ch]
    mov  es, ax
    mov  ah, 49h
    int  21h
    pop  es
    mov  ah, 49h
    int  21h

    sti

    pop  si
    pop  es
    pop  ds
    pop  dx
    pop  bx
    pop  ax

```



```

        ret
UNLOADP    endp
;-----
CHECKUN    proc
    push ax
    push es

    mov ax, keep_psp
    mov es, ax
    cmp byte ptr es:[82h], '/'
    jne endun
    cmp byte ptr es:[83h], 'u'
    jne endun
    cmp byte ptr es:[84h], 'n'
    jne endun
    mov un, 1

endun:
    pop es
    pop ax
    ret
CHECKUN    endp
;-----
PRINT proc near
    mov ah, 09h
    int 21h
    ret
PRINT endp
;-----
; Код
MAIN PROC
    push DS    ;\ Сохранение адреса начала PSP в стеке
    sub AX,AX  ; > для последующего восстановления по
    push AX    ;/ команде ret, завершающей процедуру.
    mov AX,DATA    ; Загрузка сегментного
    mov DS,AX      ; регистра данных.
    mov keep_psp, es

    call check
    call checkun
    cmp un, 1
    je unload1

    mov al, loaded
    cmp al, 1
    jne load1
    mov dx, offset loaded_inf
    call print
    jmp exit

load1:
    mov dx, offset load_inf
    call print
    call loadp

```

```

        jmp    exit
unload1:
        cmp    loaded, 1
        jne    notloaded1
        call UNLOADP
        mov    dx, offset unload_inf
        call print
        jmp    exit
notloaded1:
        mov    dx, offset not_load_inf
        call print

exit:
; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
MAIN ENDP
CODE ENDS

AStack SEGMENT STACK
        DW 128 DUP(0)
AStack ENDS

DATA SEGMENT
load_inf      db    'Interruption loaded',0DH,0AH,'$'
loaded_inf    db    'Interruption already loaded',0DH,0AH,'$'
unload_inf    db    'Interruption has been unloaded',0DH,0AH,'$'
not_load_inf  db    'Interruption not loaded',0DH,0AH,'$'

loaded        db    0
un            db    0
DATA ENDS

        END MAIN

```