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. 7383 _____ Корякин М.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт управление по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

В лабораторной работе № 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Описание функций.

- ROUT - пользовательский обработчик прерываний, считающий и печатающий количество его вызовов.
- setCurs - устанавливает курсор в строку dh, колонку dl.
- getCurs - возвращает положение курсора в dh, dl.
- outputBP - функция вывода строки по адресу ES:BP.
- PRINT - вызывает функцию печати строки.
- PROV_ROUT - проверяет, установлен ли пользовательский обработчик прерывания, и если нет – устанавливает его. В ином случае, если хвост равен ‘/un’, восстанавливает стандартное.
- SET_ROUT - устанавливает пользовательское прерывание.
- DEL_ROUT - удаляет пользовательское прерывание.
- SAVE_STAND - сохраняет адрес стандартного прерывания в KEEP_IP, KEEP_CS.
- BYTE_TO_HEX - переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah.

- TETR_TO_HEX - вспомогательная функция для работы функции BYTE_TO_HEX.
- WRD_TO_HEX - переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры.

Результаты работ написанной программы.

```
Object filename [l4.OBAmount of pins: 002A
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  49898 + 453267 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors

C:\>link l4.obj

Microsoft (R) Overlay Linker  Version 3.64
Copyright (C) Microsoft Corp 1983-1988.  All rights reserve

Run File [L4.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>l4
Res is load
C:\>_
```

Рисунок 1 – Результат работы программы

```
C:\>l4/un
Res unload              Amount of pins: 00F8

C:\>l3-1
available memory:  648912 byte
extended memory:  15360 Kb
Address    Owner         Size Name
 016F      0008           16
 0171      0000           64
 0176      0040          256
 0187      0192          144
 0191      0192       648912  L3-1

C:\>l4/un
Res still unload

C:\>l4
Res is load
C:\>l4
Res already load
```

Рисунок 2 – Тестирование повторного запуска программы

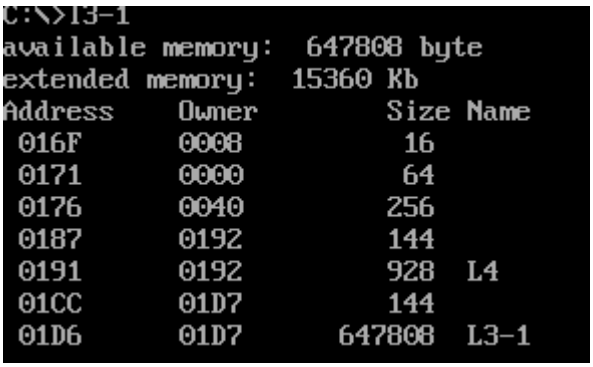

Рисунок 3 – Состояние памяти во время работы обработчика прерываний

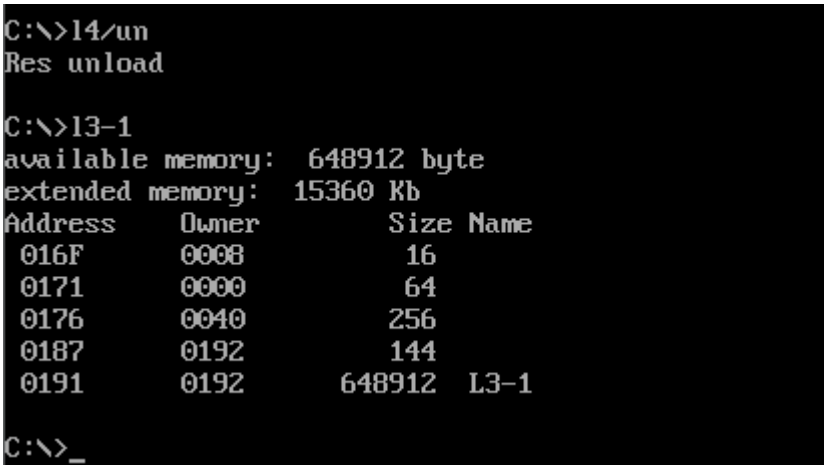

Рисунок 4 – Состояние памяти после выгрузки обработчика прерываний

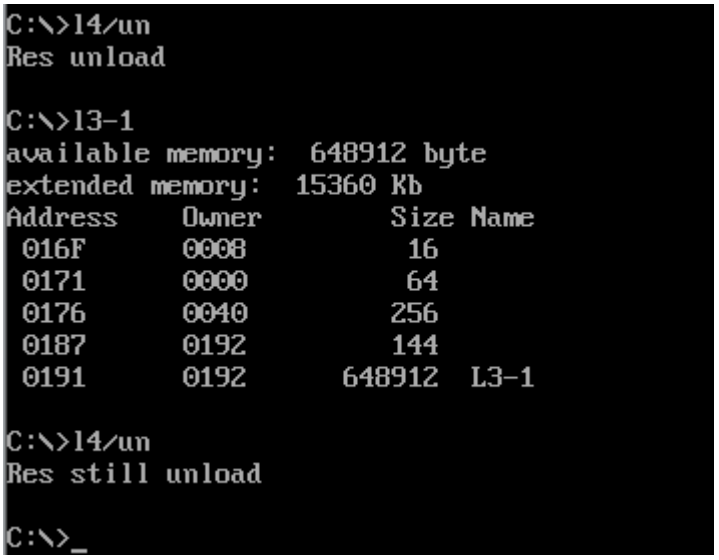

Рисунок 5 – Тестирование повторной выгрузки обработчика прерываний

Выводы.

В лабораторной работе был изучен и построен обработчик прерываний сигналов таймера.

Ответы на контрольные вопросы.

1. Как реализован механизм прерывания от часов?

   Сперва сохраняется содержимое регистров. После чего определяется источник прерывания, по которому идентифицируется смещение в таблице векторов прерывания и сохраняется в CS:IP, по этому же адресу передается управление обработчика. После выполнения обработчика управление передается обратно прерванной программе.

2. Какого типа прерывания использовались в работе?

   В работе использовалось два типа прерываний: программные – это int 21h, int 10h; и аппаратные – это 1Ch.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK

ROUT PROC FAR
      jmp go_
      SIGNATURA dw 0ABCDh
      KEEP_PSP dw 0
      KEEP_IP dw 0
      KEEP_CS dw 0
      INT_STACK DW 100 dup (?)
      COUNT dw 0
      KEEP_SS DW 0
      KEEP_AX      DW ?
      KEEP_SP DW 0
      VIVOD db 'Amount of pins:     $'

go_:
      mov KEEP_SS, SS
      mov KEEP_SP, SP
      mov KEEP_AX, AX
      mov AX,seg INT_STACK
      mov SS,AX
      mov SP,0
      mov AX,KEEP_AX

      push ax
      push bp
      push es
      push ds
      push dx
      push di

      mov ax,cs
      mov ds,ax
      mov es,ax
      mov ax,CS:COUNT
      add ax,1
```

```
      mov CS:COUNT,ax
      mov di,offset vivod+19
      call WRD_TO_HEX
      mov bp,offset vivod
      call outputBP

      pop di
      pop dx
      pop ds
      pop es
      pop bp
      mov al,20h
      out 20h,al
      pop ax
      mov AX,KEEP_SS
      mov SS,AX
      mov AX,KEEP_AX
      mov SP,KEEP_SP
      iret
ROUT ENDP
; -------------------------------------
TETR_TO_HEX PROC near
      and AL,0Fh
      cmp AL,09
      jbe NEXT
      add AL,07
NEXT: add AL,30h
      ret
TETR_TO_HEX ENDP
;-------------------------------------
BYTE_TO_HEX PROC near
      push CX
      mov AH,AL
      call TETR_TO_HEX
      xchg AL,AH
      mov CL,4
      shr AL,CL
      call TETR_TO_HEX
      pop CX
      ret
BYTE_TO_HEX ENDP
```

```
;-------------------------------------
WRD_TO_HEX PROC near
      push BX
      mov BH,AH
      call BYTE_TO_HEX
      mov [DI],AH
      dec DI
      mov [DI],AL
      dec DI
      mov AL,BH
      call BYTE_TO_HEX
      mov [DI],AH
      dec DI
      mov [DI],AL
      pop BX
      ret
WRD_TO_HEX ENDP
;-------------------------------------
outputBP PROC near
      push ax
      push bx
      push dx
      push cx
      mov ah,13h
      mov al,0
      mov bl,04h
      mov bh,0
      mov dh,4
      mov dl,22
      mov cx,20
      int 10h
      pop cx
      pop dx
      pop bx
      pop ax
      ret
outputBP ENDP
LAST_BYTE:
;-------------------------------------
PRINT PROC
      push ax
```

```
      mov ah,09h
      int 21h
      pop ax
      ret
PRINT ENDP
;--------------------------------------
PROV_ROUT PROC
      mov ah,35h
      mov al,1ch
      int 21h
      mov si,offset SIGNATURA
      sub si,offset ROUT
      mov ax,0ABCDh
      cmp ax,ES:[BX+SI]
      je ROUT_EST
            call SET_ROUT
            jmp PROV_KONEC
      ROUT_EST:
            call DEL_ROUT
      PROV_KONEC:
      ret
PROV_ROUT ENDP
;--------------------------------------
SET_ROUT PROC
      mov ax,KEEP_PSP
      mov es,ax
      cmp byte ptr es:[80h],0
            je UST
      cmp byte ptr es:[82h],'/'
            jne UST
      cmp byte ptr es:[83h],'u'
            jne UST
      cmp byte ptr es:[84h],'n'
            jne UST

      mov dx,offset PRER_NE_SET_VIVOD
      call PRINT
      ret

UST:
      call SAVE_STAND
```

```
	mov dx,offset PRER_SET_VIVOD
	call PRINT

	push ds
	mov dx,offset ROUT
	mov ax,seg ROUT
	mov ds,ax

	mov ah,25h
	mov al,1ch
	int 21h
	pop ds

	mov dx,offset LAST_BYTE
	mov cl,4
	shr dx,cl
	add dx,1
	add dx,20h

	xor AL,AL
	mov ah,31h
	int 21h

	xor AL,AL
	mov AH,4Ch
	int 21H
SET_ROUT ENDP
;--------------------------------------
DEL_ROUT PROC
	push dx
	push ax
	push ds
	push es


	mov ax,KEEP_PSP
	mov es,ax
	cmp byte ptr es:[80h],0
		je UDAL_KONEC
	cmp byte ptr es:[82h],'/'
```

```
      jne UDAL_KONEC
cmp byte ptr es:[83h],'u'
      jne UDAL_KONEC
cmp byte ptr es:[84h],'n'
      jne UDAL_KONEC

mov dx,offset PRER_DEL_VIVOD
call PRINT

mov ah,35h
mov al,1ch
int 21h
mov si,offset KEEP_IP
sub si,offset ROUT

mov dx,es:[bx+si]
mov ax,es:[bx+si+2]
mov ds,ax
mov ah,25h
mov al,1ch
int 21h

mov ax,es:[bx+si-2]
mov es,ax
mov ax,es:[2ch]
push es
mov es,ax
mov ah,49h
int 21h
pop es
mov ah,49h
int 21h

jmp UDAL_KONEC2

UDAL_KONEC:
mov dx,offset PRER_UZHE_SET_VIVOD
call PRINT
UDAL_KONEC2:
pop es
pop ds
```

```
      pop ax
      pop dx
      ret
DEL_ROUT ENDP
;---------------------------------------
SAVE_STAND PROC
      push ax
      push bx
      push es
      mov ah,35h
      mov al,1ch
      int 21h
      mov KEEP_CS, ES
      mov KEEP_IP, BX
      pop es
      pop bx
      pop ax
      ret
SAVE_STAND ENDP
;---------------------------------------
BEGIN:
      mov ax,DATA
      mov ds,ax
      mov KEEP_PSP, es
      call PROV_ROUT
      xor AL,AL
      mov AH,4Ch
      int 21H
CODE ENDS

DATA SEGMENT
      PRER_SET_VIVOD db 'Res is load','$'
      PRER_DEL_VIVOD db 'Res unload',0DH,0AH,'$'
      PRER_UZHE_SET_VIVOD db 'Res already load',0DH,0AH,'$'
      PRER_NE_SET_VIVOD db 'Res still unload',0DH,0AH,'$'
      STRENDL db 0DH,0AH,'$'
DATA ENDS
ASTACK SEGMENT STACK
      dw 100h dup (?)
ASTACK ENDS
 END BEGIN
```