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. 7383 _____ Иолшина В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

Цель работы.

Изучить механизм обработки стандартных прерываний, установку пользовательских резидентных обработчиков прерываний и восстановление исходных с выгрузкой резидентных функций из памяти.

Построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определенные интервалы времени и, при возникновении такого сигнала, возникает прерывание с определенным значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Таблица 1 - Описание процедур и структур данных.

Название процедуры	Назначение
PRINT	вызывает функцию печати строки
setCurs	устанавливает позицию курсора
getCurs	возвращает положение курсора
ROUT	пользовательский обработчик прерываний, считающий и печатающий количество его вызовов
CHECKING	проверяет загруженность обработчика прерываний
SET_INTERRUPT	устанавливает новый обработчик прерывания, запоминая данные для восстановления предыдущего обработчика прерываний

Последовательность действий, выполняемых программой.

1. Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
2. Устанавливает обработчик прерываний, если он не установлен, и осуществляется выход.
3. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход.

4. Выгрузка прерывания по соответствующему значению параметра в командной строке /un, восстановления стандартного вектора прерывания.

Результаты работы программы представлены на рис. 1-6.

1. Состояние памяти до запуска программы 4.exe представлено на рис.1, в качестве программы изображена работа файла lr3_1.com.

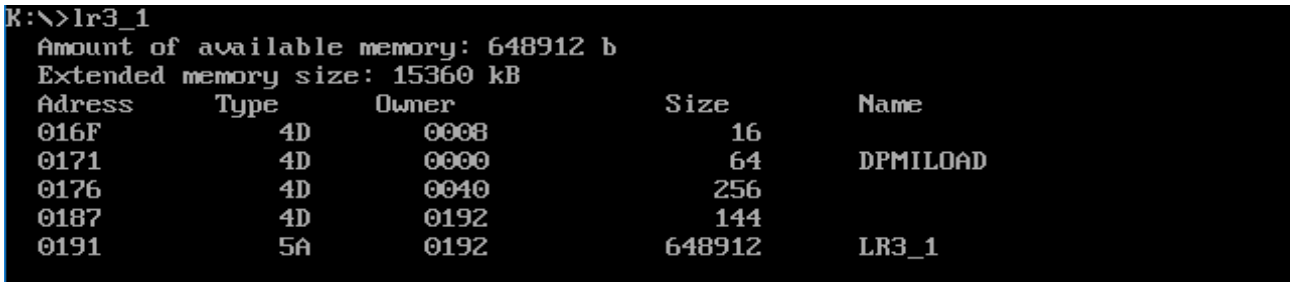

Рисунок 1 – Результат выполнения программы lr3_1.com

2. Резидентный обработчик прерывания 1Ch установлен. На рис. 2 представлен запуск программы 4.exe

Рисунок 2 – Результат выполнения программы 4.exe

3. Проверим размещение прерывания в памяти с помощью программы lr3_1.com, которая отображает карту памяти в виде списка блоков MCB. Выполнение показано на рис. 3.

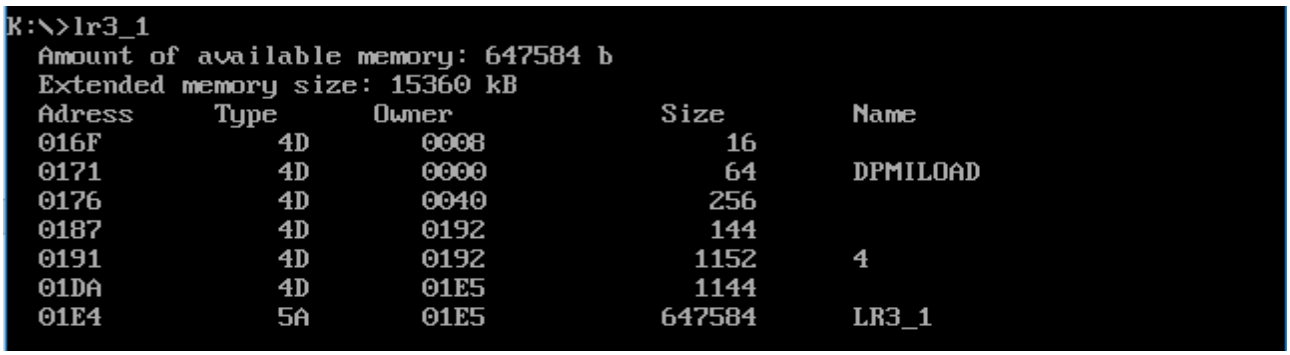

Рисунок 3 – Результат выполнения программы lr3_1.com

4. Повторный запуск программы 4.exe изображён на рис. 4.

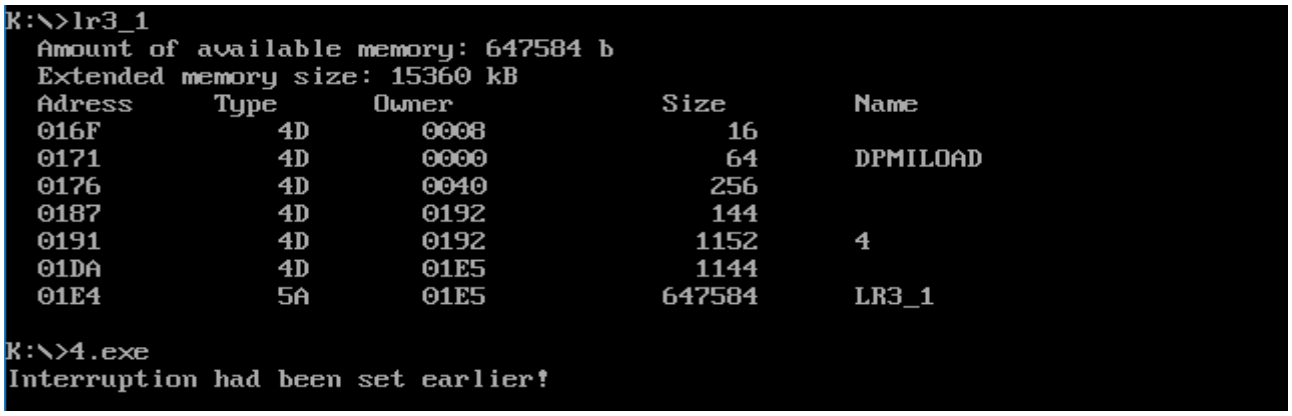

Рисунок 4 – Результат повторного выполнения программы 4.exe

5. Запустим программу 4.exe ещё раз, но с ключом выгрузки /un (см. Рис.5):

Рисунок 5 – Результат выполнения программы 4.exe с ключом /un

6. Запустим программу lr3_1.com, чтобы убедиться в том, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память занятая резидентом освобождена.

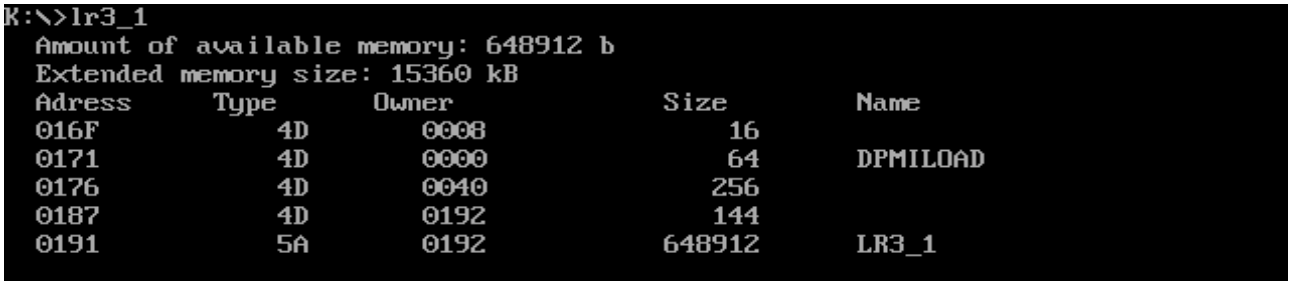

Рисунок 6 – Результат повторного выполнения программы lr3_1.com

Выводы.

В процессе выполнения данной лабораторной работы был изучен механизм обработки стандартных прерываний, а также построен обработчик прерываний сигналов таймера, который проверяет, установлено ли пользовательское прерывание с вектором 1Ch (вызывается системой каждые 55 мс), устанавливает резидентную функцию для обработки прерывания, выгружает пользовательское прерывание по соответствующему значению

параметра командной строки «/un», а также выводит на экран информацию об общем суммарном числе прерываний

Ответы на контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Аппаратное прерывание 1Ch вызывается автоматически при каждом такте системного таймера (каждые 55 мс) и реализовано следующим образом:

1) сохранение содержимого регистров в стеке;

2) сохранение положения курсора и его перемещение в позицию для вывода сообщения о количестве выполнения прерывания;

3) определение источника прерывания, по номеру которого определяется смещение в таблице векторов прерывания;

4) сохранение в CS и IP;

5) передача управления по адресу CS:IP;

6) выполнение обработчика прерывания;

7) курсор возвращается в исходную позицию, все регистры восстанавливаются;

8) завершение работы обработчика и возврат управления прерванной программе.

2. Какого типа прерывания использовались в работе?

В данной лабораторной работе использовались аппаратные (int 1Ch - пользовательское прерывание по таймеру) и программные (int 21h и int 10h - для вывода информации на экран средствами BIOS) прерывания.

ПРИЛОЖЕНИЕ А

4.asm

```
STACK SEGMENT STACK
	DW 100 DUP (?)
STACK ENDS


DATA SEGMENT
	wasloaded DB 'Interruption had been set earlier!',0DH,0AH,'$'
	unloaded DB 'Interruption has been unloaded!',0DH,0AH,'$'
	loading DB 'Interruption has been set!',0DH,0AH,'$'
DATA ENDS


CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK
START: JMP BEGIN


PRINT PROC NEAR
	push ax
	mov ah, 09h
	int 21h
	pop ax
	ret
PRINT ENDP


setCurs PROC ; установка позиции курсора
	push AX
	push BX
	push CX
	mov AH,02h
	mov BH,00h
	int 10h
	pop CX
	pop BX
	pop AX
	ret
setCurs ENDP


getCurs PROC ; определение позиции и размера курсора
	push AX
```

```
        push BX
        push CX
        mov AH,03h
        mov BH,00h
        int 10h
        pop CX
        pop BX
        pop AX
        ret
getCurs ENDP


ROUT PROC FAR ; обработчик прерываний
        jmp ROUT_

; Данные
_DATA:
        SIGN DB '0000'
        KEEP_CS DW 0
        KEEP_IP DW 0
        KEEP_PSP DW 0
        VALUE DB 0
        COUNTER DB '    Number of calls: 00000    $' ; счетчик
        STACK_ DW       64 dup (?)
        KEEP_SS DW 0
        KEEP_AX         DW      ?
  KEEP_SP DW 0

        ROUT_:
        mov KEEP_SS, SS
        mov KEEP_AX, AX
        mov KEEP_SP, SP
        mov AX, seg STACK_
        mov SS, AX
        mov SP, 0
        mov AX, KEEP_AX

        push AX
        push DX
        push DS
        push ES
        cmp VALUE, 1
        je ROUT_RES
        call getCurs
        push DX
        mov DH,7
```

```
        mov DL,15
        call setCurs

ROUT_SUM: ; счетчик количества прерываний
        push SI
        push CX
        push DS
        push AX
        mov AX,SEG COUNTER
        mov DS,AX
        mov bx, offset COUNTER
        add bx, 22
        mov si, 3
next_:
        mov ah, [bx+si]
        inc ah
        cmp ah, 58
        jne ROUT_NEXT
        mov ah, 48
        mov [bx+si], ah
        dec si
        cmp si, 0
        jne next_
ROUT_NEXT:
        mov [bx+si],ah
        pop ds
        pop si
        pop bx
        pop ax
        push es
        push bp
        mov ax,SEG COUNTER
        mov es,ax
        mov ax,offset COUNTER
        mov bp,ax
        mov ah,13h
        mov al,0
        mov cx,30
        mov bh,0
        int 10h
        pop bp
        pop es
        pop dx
        call setCurs
        jmp ROUT_END
```

```
ROUT_RES: ; восстановление вектора прерывания
        CLI
        mov DX,KEEP_IP
        mov AX,KEEP_CS
        mov DS,AX
        mov AH,25h
        mov AL,1Ch
        int 21h
        mov ES, KEEP_PSP
        mov ES, ES:[2Ch]
        mov AH, 49h
        int 21h
        mov ES, KEEP_PSP
        mov AH, 49h
        int 21h
        STI

ROUT_END: ; восстановление регистров
        pop ES
        pop DS
        pop DX
        pop AX

        mov AX, KEEP_SS
        mov SS, AX
        mov SP,KEEP_SP
        mov AX,KEEP_AX
        iret
ROUT ENDP


CHECKING PROC  ; проверка пользовательского прерывания
        mov AH,35h
        mov AL,1Ch
        int 21h
        mov SI, offset SIGN
        sub SI, offset ROUT
        mov AX,'00'
        cmp AX,ES:[BX+SI]
        jne UNLOAD
        cmp AX,ES:[BX+SI+2]
        je LOAD

UNLOAD:
        call SET_INTERRUPT
        mov DX,offset LAST_BYTE
```

```
        mov CL,4
        shr DX,CL
        inc DX
        add DX,CODE
        sub DX,KEEP_PSP
        xor AL,AL
        mov AH,31h
        int 21h
LOAD:
        push ES
        push AX
        mov AX,KEEP_PSP
        mov ES,AX
        cmp byte ptr ES:[82h],'/'
        jne BACK
        cmp byte ptr ES:[83h],'u'
        jne BACK
        cmp byte ptr ES:[84h],'n'
        je UNLOAD_
BACK:
        pop AX
        pop ES
        mov dx,offset wasloaded
        call PRINT
        ret
UNLOAD_:
        pop AX
        pop ES
        mov byte ptr ES:[BX+SI+10],1
        mov dx,offset unloaded
        call PRINT
        ret
CHECKING ENDP


SET_INTERRUPT PROC ; добавление нового прерывания
        push DX
        push DS
        mov AH,35h
        mov AL,1Ch
        int 21h
        mov KEEP_IP,BX
        mov KEEP_CS,ES
        mov DX,offset ROUT
        mov AX,seg ROUT
        mov DS,AX
```

```
        mov AH,25h
        mov AL,1Ch
        int 21h
        pop DS
        mov DX,offset loading
        call PRINT
        pop DX
        ret
SET_INTERRUPT ENDP


BEGIN:
        mov AX,DATA
        mov DS,AX
        mov KEEP_PSP,ES
        call CHECKING
        xor AL,AL
        mov AH,4Ch
        int 21H
LAST_BYTE:
        CODE ENDS
        END START
```