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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Операционные системы»

Тема: Обработка стандартных прерываний

Студентка гр. 7383 Маркова А.В.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

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 определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора.

Описание процедур, используемых в работе:

- LINE_OUTPUT – выводит сообщение на экран;
- SET_CURS – установка курсора в заданную позицию, где DH,DL – строка, колонка соответственно;
- GET_CURS – определяет текущую позицию курсора. После выполнения, возвращает DH – текущая строка и DL – текущая колонка курсора;
- ROUT – обработчик прерываний сигналов таймера. Выводит на экран счётчик – суммарное число прерываний 1Ch. При значении check = 1 восстанавливает стандартный вектор прерывания и выгружает из памяти пользовательское прерывание;
- CHECK_INT – проверка, установлено ли пользовательское прерывание с вектором 1Ch. Также смотрит есть ли в хвосте «/un», если да, то присваивает значению check единицу;
- MY_INT – установка написанного прерывания в поле векторов прерываний.

Таблица 1 – Описание структуры данных управляющей программы:

Название	**Тип**	**Назначение**
interrupt_already_loaded	db	Строка – сообщение для вывода информации о том, что пользовательское прерывание уже было установлено
interrupt_was_unloaded	db	Строка – сообщение для вывода информации о том, что пользовательское прерывание выгружено
interrupt_was_loaded	db	Строка – сообщение для вывода информации о том, что пользовательское прерывание установлено

Таблица 2 – Описание структуры данных собственного прерывания:

Название	**Тип**	**Назначение**
signature	db	Сигнатура пользовательского прерывания
keep_psp	dw	Переменная для сохранения сегментного адреса PSP
keep_ss	dw	Переменная для сохранения сегментного адреса стека
keep_ax	dw	Переменная для сохранения в AX
keep_sp	dw	Переменная для сохранения указателя стека
keep_cs	dw	Переменная для сохранения в CS
keep_ip	dw	Переменная для сохранения в IP
check	dw	Флаг для определения необходимости выгружать прерывание
timer	db	Число вызовов прерывания

Ход работы.

1. Был написан программный модуль .EXE, который выполняет следующие функции:
   - Проверяет установлено ли пользовательское прерывание с вектором 1Ch;
   - Устанавливает резидентную функцию для обработки прерывания, настраивает вектор прерывания, если прерывание не установлено;
   - Если прерывание установлено, то выводит соответствующее сообщение на экран;
   - Выгрузка прерывания происходит при получении параметра «/un» в командной строке при вызове программы.
2. Смонтирован виртуальный диск k с каталогом tasm.
3. Было произведено транслирование программы с помощью строки tasm «имя файла», в последствии чего создался объектный файл.
4. Линковка загрузочного модуля с помощью команды tlink.
5. Получен LR4.EXE модуль из исходного кода для .EXE модуля.
6. Была дана оценка состояния памяти до запуска LR4.EXE, используя программу из предыдущей лабораторной работы. Результат представлен на рис. 1.

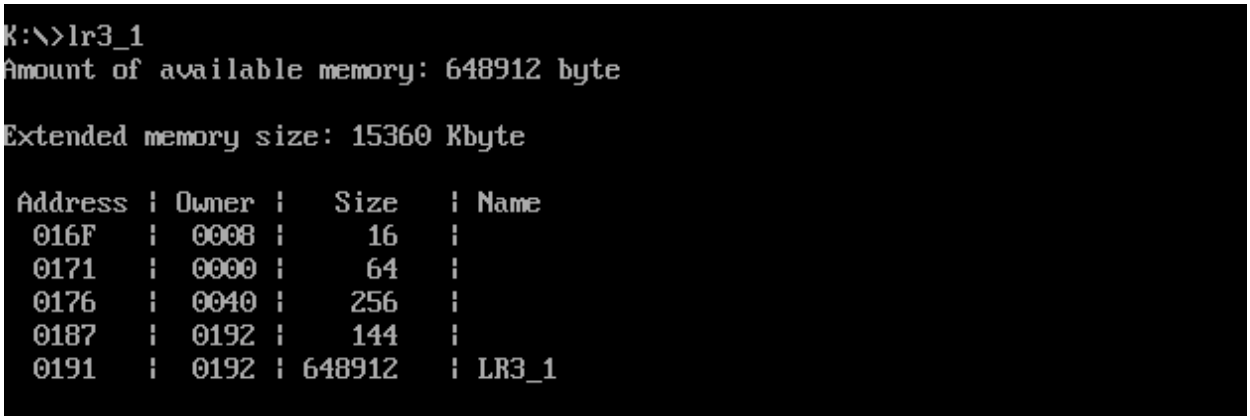

Рисунок 1 – Результат работы программы lr3_1.com

7. Запущена отлаженная программа, чтобы удостовериться, что резидентный обработчик прерывания 1Ch установлен. Работа прерывания отображается на экране, что показано на рис. 2.

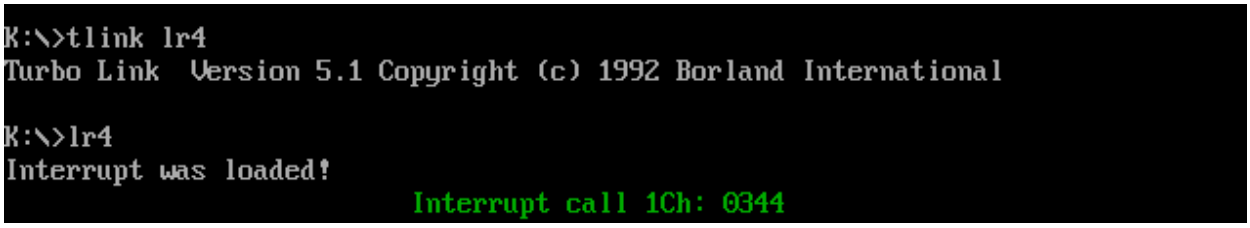

Рисунок 2 – Результат работы программы lr4.exe

8. Была запущена отлаженная программа ещё один раз, чтобы убедиться, что LR4.EXE определяет установленный обработчик прерывания. Результат представлен на рис. 3.

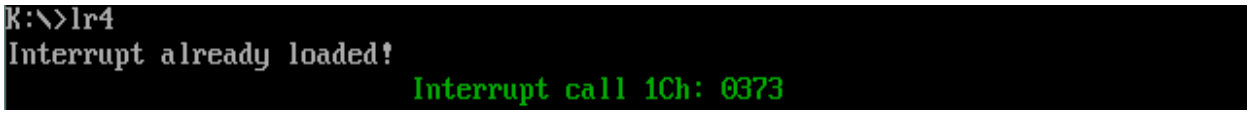

Рисунок 3 – Повторный запуск программы

9. Проверено размещение прерывания в памяти с помощью LR3_1.COM, которая отображает карту памяти в виде списка блоков MCB. Вывод на консоль после запуска программы показан на рис.4.

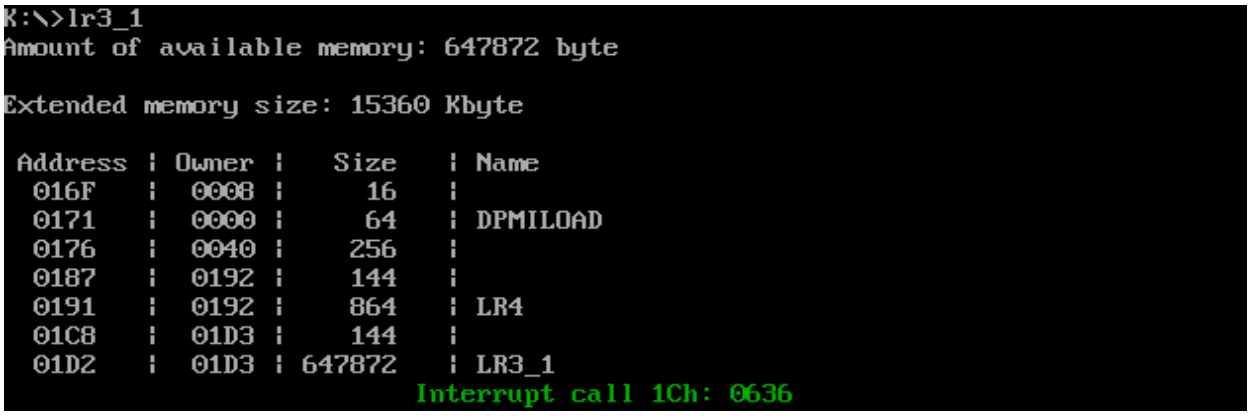

Рисунок 4 – Размещение прерывания в памяти

10. Запущена отлаженная программа с ключом выгрузки «/un», по результатам, которые представлены на рис. 5 – 6, видно, что

резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память занятая резидентом освобождена.

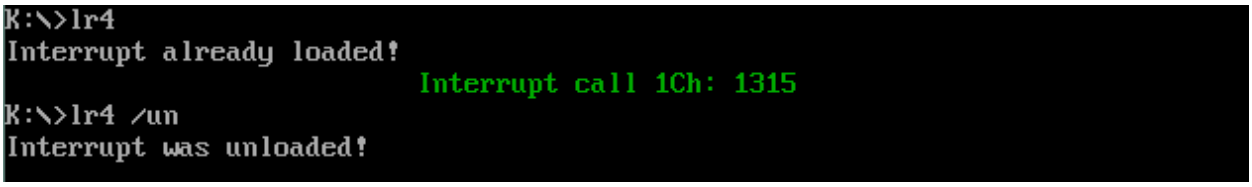

Рисунок 5 – Результат запуска программы с ключом «/un»

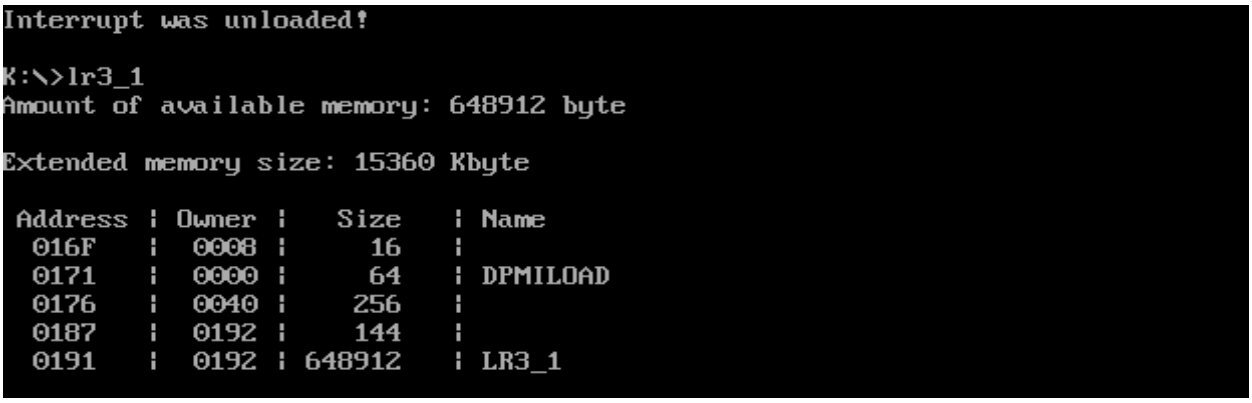

Рисунок 6 – Результат работы программы lr3_1.com

Выводы.

В ходе данной лабораторной работы был изучен механизм обработки стандартных прерываний. Также был создан 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. Код программы представлен в приложении А.

Ответы на контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Аппаратное прерывание 1Ch вызывается автоматически при каждом такте системного таймера (каждые 55 мс). После вызова, в стеке сохраняются значения регистров, которые будут использоваться в обработчике. Происходит сохранение положения курсора и его перемещение в позицию для вывода сообщения о том какой раз было выполнено прерывание. Затем определяется источник прерывания, по номеру которого формируется смещение в таблице векторов прерываний, устанавливаются CS и IP. Выполняется процедура обработчика прерываний. Затем курсор возвращается в исходную позицию, все регистры восстанавливаются. Работа обработчика завершается и управление передаётся в программу, которая вызывала обработчик. Возвращение произойдёт в ту точку, где она была прервана. Для этого из стека берутся исходные данные сегментов IP и CS.

2. Какого типа прерывания использовались в работе?

В данной лабораторной работе использовались аппаратные и программные прерывания.

Аппаратное прерывание:

- int 1Ch – пользовательское прерывание по таймеру. Берёт по каждому тику аппаратных часов (каждые 55 мс). Первоначально указывает на IRET.

Программные прерывания:

- int 21h – используется для большинства функций DOS;
- int 10h – видео сервис. Используется для вывода информации на экран средствами BIOS. Возможен вывод в любую точку экрана.

ПРИЛОЖЕНИЕ А

LR4.ASM

```
EOFLine EQU '$'                                 ; определение символьной константы
                                                ; $ - "конец строки"

MY_STACK SEGMENT STACK
    DW 64 DUP (?)
MY_STACK ENDS

STACK SEGMENT STACK
    DW 64 DUP (?)
STACK ENDS
;ДАННЫЕ
;-------------------------------------------------------------------------------------------
DATA SEGMENT
    interrupt_already_loaded DB 'Interrupt already loaded!',  0DH,0AH,EOFLine
    interrupt_was_unloaded   DB 'Interrupt was unloaded!',    0DH,0AH,EOFLine
    interrupt_was_loaded     DB 'Interrupt was loaded!',      0DH,0AH,EOFLine
DATA ENDS
;-------------------------------------------------------------------------------------------
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK
START: JMP BEGINNING                            ;переход на метку
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------------------------
LINE_OUTPUT PROC NEAR                           ;вывод строки
    mov AH, 0009h
    int 21h
    ret
LINE_OUTPUT ENDP

SET_CURS PROC                                   ;установка позиции курсора; установка на строку 25 делает курсор невидимым
    push AX
    push BX
    push CX
    mov  AH, 0002h
    mov  BH, 0000h
    int  0010h                                  ;выполнение функции 0002h
    pop  CX                                     ;Вход: BH = видео страница
    pop  BX                                     ;      DH,DL = строка, колонка (считая от 0)
    pop  AX
    ret
SET_CURS ENDP

GET_CURS PROC                                   ;функция, определяющая позицию и размер курсора
    push AX
    push BX
    push CX
    mov  AH, 0003h                              ;0003h читать позицию и размер курсора
    mov  BH, 0000h                              ;Вход: BH = видео страница
    int  10h                                    ;выполнение функции 0003h
    pop  CX                                     ;Выход: DH, DL = текущая строка, колонка курсора
    pop  BX                                     ;       CH, CL = текущая начальная, конечная строки курсора
    pop  AX
    ret
GET_CURS ENDP
;-------------------------------------------------------------------------------------------
ROUT PROC FAR                                   ;обработчик прерывания
```

```
    jmp ROUT_BEGINNING
ДАННЫЕ
-------------------------------------------------------------------------------------------
    signature db '0000'                                 ;некоторый код, который идентифицирует резидента
    keep_cs   dw 0                                      ;для хранения сегмента кода
    keep_ip   dw 0                                      ;для хранения смещения прерывания
    keep_psp  dw 0                                      ;для хранения PSP
    check     dw 0                                      ;проверяем надо выгружать прерывание или нет
    keep_ss   dw 0                                      ;для хранения сегмента стека
    keep_ax   dw 0                                      ;для хранения регистра AX
    keep_sp   dw 0                                      ;для хранения регистра SP
    timer     db 'Interrupt call 1Ch: 0000 $'           ;счётчик
-------------------------------------------------------------------------------------------
    ROUT_BEGINNING:
        mov  keep_ax, AX                                ;запоминаем ax
        mov  keep_ss, SS                                ;запоминаем начало стека
        mov  keep_sp, SP                                ;запоминаем регистр SP
        mov  AX, MY_STACK                               ;устанавливаем собственный стек
        mov  SS, AX
        mov  SP, 64h
        mov  AX, keep_ax
        push DX                                         ;сохраняем все изменяемые регистры
        push DS
        push ES
        cmp  check, 1
        je   ROUT_REC
        call GET_CURS                                   ;получаем текущее положение курсора
        push DX                                         ;сохраняем положения курсора в стеке
        mov  DH, 17h                                    ;DH, DL - строка, колонка (считая от 0)
        mov  DL, 1Ah                                    ;определяем местоположение надписи
        call SET_CURS                                   ;устанавливаем курсор

    ROUT_COUNT:                                         ;счётчик количества прерываний
        push SI                                         ;сохраняем все изменяемые регистры
        push CX
        push DS
        mov  AX, seg timer
        mov  DS, AX
        mov  SI, offset timer
        add  SI, 0017h                                  ;смещение на последнюю цифру (23 знак)

        count:
            mov  AH, [SI]                               ;получаем цифру
            inc  AH                                     ;увеличиваем её на 1
            mov  [SI], AH                               ;возвращаем
            cmp  AH, 3Ah                                ;если не равно 9
            jne  END_COUNT                              ;завершение и вывод результата
            mov  AH, 30h                                ;обнуляем
            mov  [SI], AH
            dec  SI
        loop     count

    END_COUNT:                                          ;печать счётчика-строки на экран
        pop  DS
        pop  CX
        pop  SI
        push ES
```

```
    push BP
    mov  AX, seg timer
    mov  ES, AX
    mov  AX, offset timer
    mov  BP, AX                         ;Вход: ES:BP выводимая строка
    mov  AH, 00013h                     ;функция 13h прерывания 10h, выдаёт строку в позиции курсора
    mov  AL, 0000h                      ;режим вывода
    mov  CX, 0019h                      ;длина строки = 25 символов
    mov  BH, 0000h                      ;видео страница, её номер
    mov  BL, 0002h                      ;установка атрибута
    int  10h
    pop  BP
    pop  ES
    pop  DX                             ;возвращение курсора
    call SET_CURS
    jmp  ROUT_END

ROUT_REC:                               ;восстановление вектора прерывания
    CLI                                 ;запрещение прерывания, путём сбрасывания флага IF
    mov DX, keep_ip
    mov AX, keep_cs
    mov DS, AX                          ;DS:DX = вектор прерывания: адрес программы обработки прерывания
    mov AH, 25h                         ;функция 25h прерывания 21h, устанавливает вектор прерывания
    mov AL, 1Ch                         ;Вход: AL = номер вектора прерывания
    int 21h
    mov ES, keep_psp
    mov ES, ES:[2Ch]                    ;ES = сегментный адрес (параграф) освобождаемого блока памяти
    mov AH, 49h                         ;функция 49h прерывания 21h, освободить распределённый блок памяти
    int 21h
    mov ES, keep_psp
    mov AH, 49h
    int 21h
    STI                                 ;разрешение прерывания

ROUT_END:
    pop ES                              ;восстановление регистров
    pop DS
    pop DX
    mov SS, keep_ss
    mov SP, keep_sp
    mov AX, keep_ax
    iret

LAST_BYTE:
    ROUT ENDP
--------------------------------------------------------------------------------
HECK_INT PROC                           ;проверка, установлено ли пользовательское прерывание с вектором 1Ch
    mov AH, 35h                         ;функция 35h прерывания 21h, даёт вектор прерывания
    mov AL, 1Ch                         ;AL = номер прерывания
    int 21h                             ;Выход: ES:BX = адрес обработчика прерывания
    mov SI, offset signature
    sub SI, offset ROUT                 ;SI = смещение signature относительно начала функции прерывания
    mov AX, '00'                        ;сравнив известное значение сигнатуры
    cmp AX, ES:[BX+SI]                  ;с реальным кодом, находящимся в резиденте
    jne NOT_LOADED                      ;если значения не совпадают, то резидент не установлен
    cmp AX, ES:[BX+SI+2]
    jne NOT_LOADED
```

```
    jmp LOADED

    NOT_LOADED:
        call MY_INT                                 ;установка пользовательского прерывания
        mov  DX, offset LAST_BYTE                   ;размер в байтах от начала
        mov  CL, 4                                  ;перевод в параграфы
        shr  DX, CL                                 ;сдвиг на 4 разряда вправо
        inc  DX
        add  DX, CODE                               ;прибавляем адрес сегмента CODE
        sub  DX, keep_psp
        xor  AL, AL
        mov  AH, 31h                                ;номер функции 31h прерывания 21h, оставляем нужное количество памяти
        int  21h

    LOADED:                                         ;смотрим, есть ли в хвосте /un , тогда нужно выгружать
        push ES
        push AX
        mov AX, keep_psp
        mov ES, AX
        cmp byte ptr ES:[0082h], '/'
        jne NOT_UNLOAD
        cmp byte ptr ES:[0083h], 'u'
        jne NOT_UNLOAD
        cmp byte ptr ES:[0084h], 'n'
        je UNLOAD

    NOT_UNLOAD:
        pop  AX
        pop  ES
        mov  dx, offset interrupt_already_loaded
        call LINE_OUTPUT
        ret

    UNLOAD:
        pop  AX
        pop  ES
        mov  byte ptr ES:[BX+SI+10], 1              ;check = 1
        mov  dx, offset interrupt_was_unloaded
        call LINE_OUTPUT
        ret
CHECK_INT ENDP

MY_INT PROC                                         ;установка написанного прерывания в поле векторов прерываний
    push DX
    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
    mov  DX, offset ROUT                            ;смещение для процедуры в DX
    mov  AX, seg ROUT
    mov  DS, AX
    mov  AH, 25h                                    ;функция установки вектора
    mov  AL, 1Ch                                    ;номер вестора
    int  21h
    pop  DS
    mov  DX, offset interrupt_was_loaded
    call LINE_OUTPUT
    pop  DX
    ret
MY_INT ENDP
;-----------------------------------------------------------------------------------------
BEGINNING:
    mov  AX, DATA
    mov  DS, AX
    mov  keep_psp, ES
    call CHECK_INT
    xor  AL, AL
    mov  AH, 4Ch                                    ;выход
    int  21H
    CODE ENDS
    END  START
```