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. 0382 _____ Морева Е.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определенные вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передает 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 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.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить 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 осуществляется выход по функции 4Ch прерывания int 21h.

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

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.

Для того, чтобы проверить установку прерывания, можно поступить

следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код, который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом, находящимся в резиденте, можно определить, установлен ли резидент. Если значения совпадают, то резидент установлен. Длину кода сигнатуры должна быть достаточной, чтобы сделать случайное совпадение маловероятным.

Программа должна содержать код устанавливаемого прерывания в виде удаленной процедуры. Этот код будет работать после установки при возникновении прерывания. Он должен выполнять следующие функции:

1) Сохраняет стек прерванной программы (регистры SS и SP) в рабочих переменных и восстановить при выходе.

2) Организовать свой стек.

3) Сохранить значения регистров в стеке при входе и восстановить их при выходе.

4) При выполнении тела процедуры накапливать общее суммарное число прерываний и выводить на экран. Для вывода на экран следует использовать прерывание int 10h, которое позволяет непосредственно выводить информацию на экран.

5) Функция прерывания должна содержать только переменные, которые она использует.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания 1Ch установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛР 3, которая отображает карту памяти в виде списка блоков МСВ. Полученные результаты поместите в отчет.

Шаг 3. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

Шаг 4. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Шаг 5. Ответьте на контрольные вопросы.

Выполнение работы.

Шаг 1. Был написан и отлажен программный модуль типа .EXE, который выполняет следующие функции:

1) Проверяет, установлено ли пользовательское прерывание;

2) Устанавливает резидентную функцию для обработки прерывания;

3) Если резидентная функция уже установлена — выводит соответствующее сообщение;

4) Выгружает прерывание по значению параметра в командной строке: /un, восстанавливая стандартный вектор прерывания.

Пользовательское прерывание 1Ch — выводит информацию о количестве сигналов таймера с момента запуска программы.

Функции, реализованные в работе:

- setCurs — Установка курсора;
- getCurs — Возврат положения курсора;
- ROUT — Обработчик прерывания (считает и выводит число его вызовов от таймера);
- CHECK_USER_INT— Проверка, загружено ли пользовательское

прерывание;

- SET_INT — Установка нового обработчика прерывания с запоминанием данных для восстановления предыдущего;
- PRINT — Осуществление вывода

Шаг 2. Загрузка программы.

Запускаем программу с обработкой пользовательского прерывания:

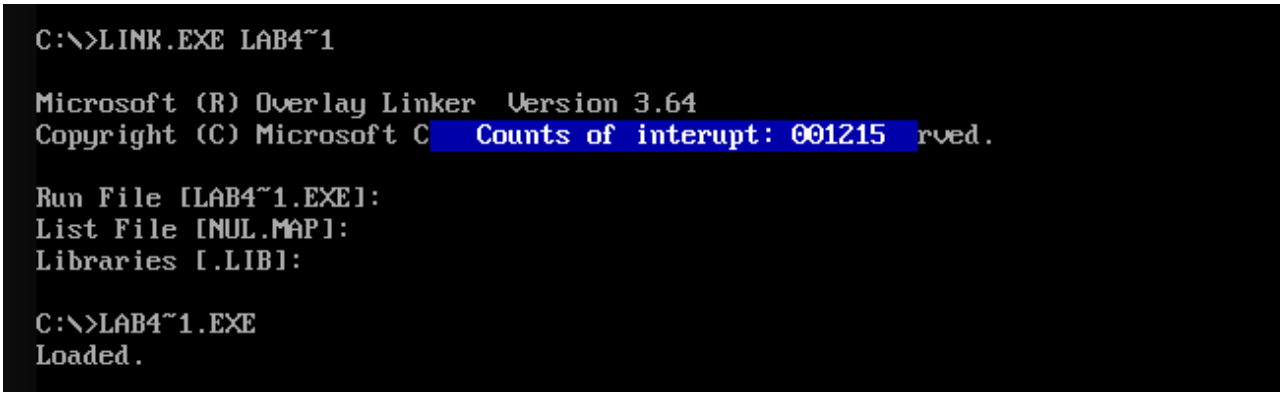

Рисунок 1 — Результат загрузки lab4~1.exe

Запускаем код 3-й лабораторной работы :

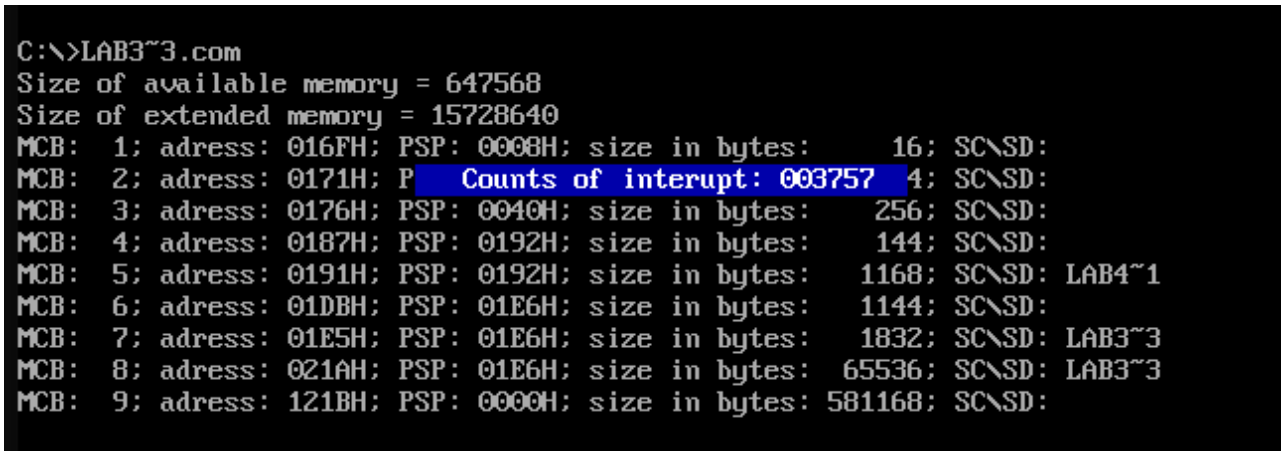

Рисунок 2 — Результат запуска lab3~3.com после установки прерывания

Мы видим, что в памяти остались блоки MCB обработчика прерывания), значит пользовательское прерывание успешно загружено в память.

Шаг 3. Повторный запуск программы.

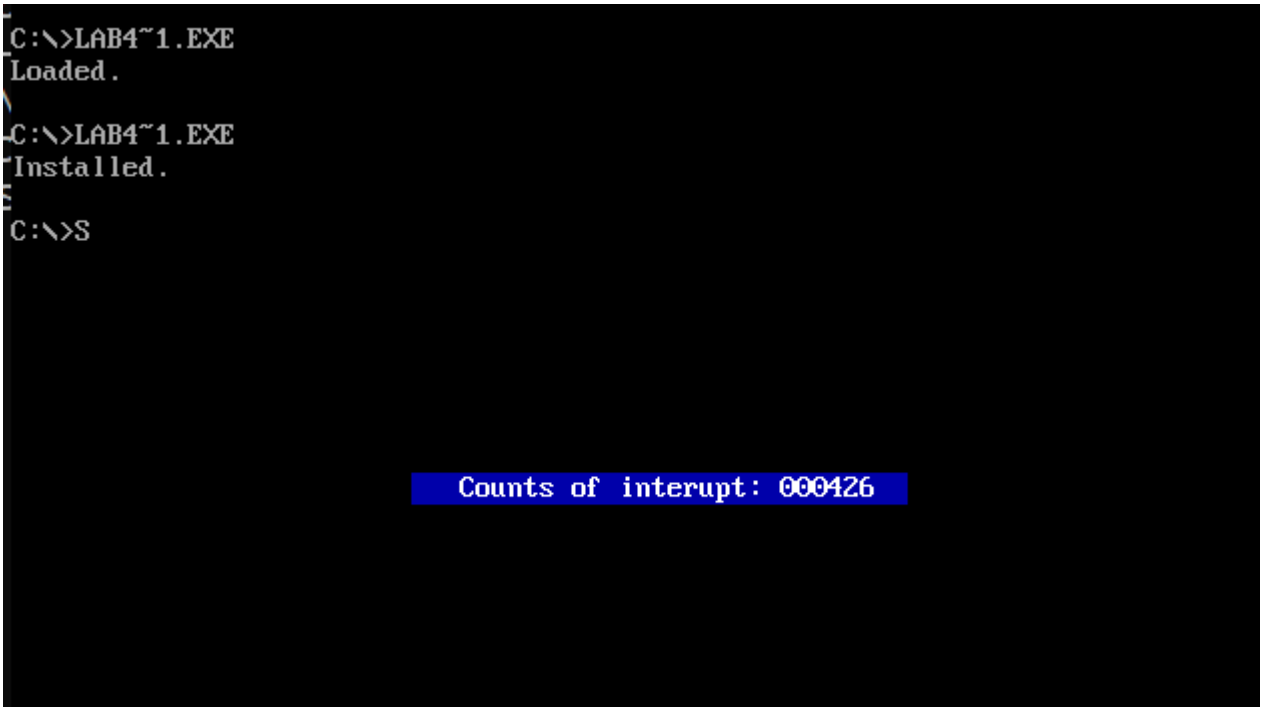

Рисунок 3 — Результат повторной загрузки lab4~1.exe

Выведено сообщение что данное прерывание уже установлено и его нет необходимости устанавливать повторно.

Шаг 4. Восстановление прерывания по умолчанию.

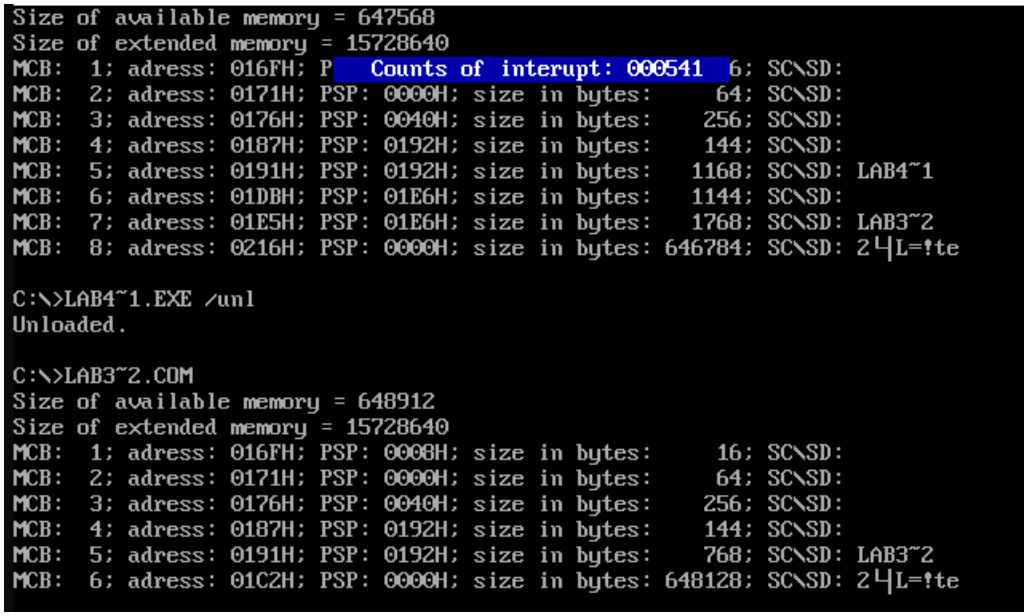

Рисунок 4 — Результат запуска lab4~1.exe с ключом выгрузки

Мы видим,что блоки памяти, в которых хранилось пользовательское прерывание удалены, значит прерывание успешно выгружено.

Исходный код программ см. в приложении А

Контрольные вопросы.

1) Как реализован механизм прерывания от часов?

Сигналы генерируются аппаратурой примерно 18 раз в секунду. При возникновении такого сигнала возникает прерывание от часов (вызывается с помощью аппаратного прерывания 1CH). При замене вектора прерывания на пользовательский, при генерации прерывания будет вызываться пользовательская функция.

2) Какого типа прерывания использовались в работе?

Программные прерывания: int 10h, int 21h

Аппаратные : int 1Ch

Выводы.

В ходе выполнения был изучен механизм обработки прерываний в DOS.

Разработана программа, загружающая и выгружающая пользовательское прерывание от системного таймера в память.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lb.asm

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

DATA SEGMENT
LOADED db 'Loaded.' , 0DH, 0AH, '$'
INSTALLED db 'Installed.', 0DH, 0AH, '$'
UNLOADED db 'Unloaded.' , 0DH, 0AH, '$'
DATA ENDS

CODE SEGMENT
   ASSUME CS:CODE, DS:DATA, ES:DATA, SS:AStack

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 DX
        push CX
        mov AH,02h
        mov BH,0
        int 10h
        pop CX
        pop DX
        pop BX
        pop AX
        ret
setCurs ENDP

getCurs PROC
; чтение позиции и размера курсора
;  вход: BH = видео страница
; выход: DH,DL = текущие строка, колонка курсора
;       CH,CL = текущие начальная, конечная строки курсора
        push AX
        push BX
        push CX
        mov AH,03h
        mov BH,0
        int 10h
        pop CX
        pop BX
        pop AX
  ret
getCurs ENDP

ROUT PROC FAR
  jmp _ROUT

  SIGN db '0000'
       KEEP_CS dw 0    ; для хранения сегмента
  KEEP_IP dw 0    ; и смещения прерывания
       KEEP_PSP dw 0
       STOP_VAL db 0
       PROC_STACK   dw 100 dup (0)
  KEEP_SS dw 0
  KEEP_AX     dw 0
  KEEP_SP dw 0
```

```
        COUNT db '   Counts of interupt: 00000    ','$'

_ROUT:
        mov KEEP_SS, SS
        mov KEEP_AX, AX
        mov KEEP_SP, SP
        mov AX, seg PROC_STACK
   mov SS, AX
   xor SP, SP
   mov AX, KEEP_AX
   push AX
   push DX
   push DS
   push ES
   cmp STOP_VAL, 1
   je RES
   call getCurs
   push DX
   mov DH, 15 ; DH,DL = строка, колонка (считая от 0)
   mov DL, 25
   call setCurs
; подсчет общего числа прерываний
   push SI
   push CX
   push DS
        push AX
   mov AX, seg COUNT
   mov DS, AX
   mov BX, offset COUNT
   add BX, 24
   mov SI, 4
_LOOP:
   mov AH,[BX+SI]
        inc AH
   cmp AH, 3AH
   jne ROUT_NEXT
   mov AH, 30H
   mov [BX+SI], AH
   dec SI
   cmp SI, 0
   jne _LOOP
ROUT_NEXT:
   mov [BX+SI], AH
   pop DS
   pop SI
   pop BX
   pop AX
   push ES
        push BP
   mov AX, seg COUNT
   mov ES, AX
```

```
   mov AX, offset COUNT
   mov BP, AX
   mov AH, 13h
   mov AL, 1
   mov CX, 31  ; число экземпляров символа для записи
   mov BH, 0   ; номер видео страницы
   int 10h
   pop BP
   pop ES
   pop DX
   call setCurs
   jmp EXIT
RES:
   cli
   mov DX, KEEP_IP
   mov AX, KEEP_CS
   mov DS, AX
   mov AH, 25h
        mov AL, 1CH
        int 21H     ; восстанавливаем вектор
   mov ES, KEEP_PSP
   mov ES, ES:[2Ch]
   mov AH, 49h
   int 21h
   mov ES, KEEP_PSP
   mov AH, 49h
   int 21h
   sti
EXIT:
   pop ES
   pop DS
   pop DX
   pop AX
        mov AX, KEEP_SS
   mov SS, AX
   mov SP, KEEP_SP
   mov AX, KEEP_AX

        mov AL, 20H
        out 20H,AL

   iret
ROUT ENDP

CHECK_USER_INT PROC
   mov AH, 35h       ; функция получения вектора
   mov AL, 1Ch       ; номер вектора
   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_INT
    mov DX, offset FINAL  ; размер в байтах от начала сегмента
    mov CL, 4             ; перевод в параграфы
    shr DX, CL
    inc DX                ; размер в параграфах
    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 INST
    cmp byte ptr ES:[83h],'u'
    jne INST
    cmp byte ptr ES:[84h],'n'
    je UNL
INST:
    pop AX
    pop ES
    mov DX, offset INSTALLED
    call PRINT
    ret
UNL:
    pop AX
    pop ES
    mov byte ptr ES:[BX+SI+10], 1
    mov DX, offset UNLOADED
    call PRINT
    ret
CHECK_USER_INT ENDP

SET_INT PROC
        push AX
        push BX
        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 LOADED
    call PRINT
    pop ES
        pop DX
        pop BX
        pop AX
    ret
SET_INT ENDP

MAIN PROC FAR
    mov AX, DATA
    mov DS, AX
        mov KEEP_PSP, ES
    call CHECK_USER_INT
    xor AL, AL
    mov AH, 4Ch
    int 21h
MAIN ENDP

        FINAL:
CODE ENDS
    END MAIN
```