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

ЛАБОРАТОРНАЯ РАБОТА № 5

по дисциплине «Операционные системы»

Тема: Сопряжение стандартного и пользовательского обработчиков прерываний.

Студент гр. 9381 _____ Матвеев А. Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерываний получает управление по прерыванию (int 09h) при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет такие же функции, как в программе ЛР 4, а именно:

1) Проверяет, установлено ли пользовательское прерывание с вектором 09h.

2) Если прерывание не установлено то, устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. 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.

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

2) При выполнении тела процедуры анализируется скан-код.

3) Если этот код совпадает с одним из заданных, то требуемый код записывается в буфер клавиатуры.

4) Если этот код не совпадает ни с одним из заданных, то осуществляется передача управления стандартному обработчику прерывания.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания 09h установлен. Работа прерывания проверяется введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.

Шаг 3. 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 4. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

Шаг 5. 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 6. Ответьте на контрольные вопросы

Выполнение работы.

1) Написан и отлажен программный модуль типа .EXE, который выполняет такие же функции, как и в программе лабораторной работы №4, а именно:

• Проверяет, установлено ли пользовательское прерывание с вектором 09h.

• Если прерывание не установлено, то устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. Осуществляется выход по функции 4Ch прерывания int21h.

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

Пользовательское прерывание заменяет символы, вводимые с клавиатуры:

- ‘1’ → ‘I’
- ‘2’ → ‘N’
- ‘3’ → ‘T’

2) Запустил отлаженную программу и убедился, что резидентный обработчик прерывания 09h установлен. Работа прерывания была проверена введением 123.

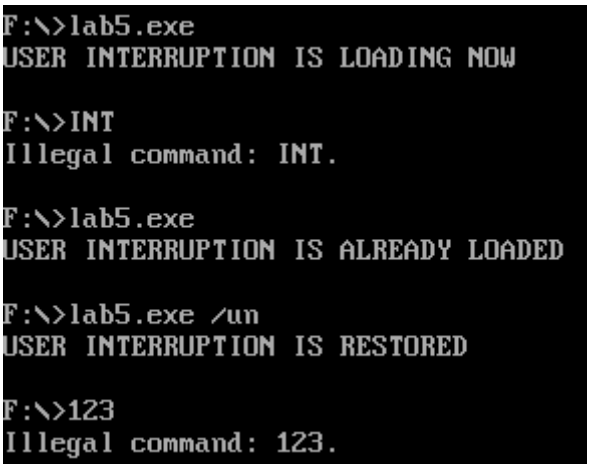

Рис. 1. Ввод ‘123’ с установленным пользовательским прерыванием и с базовым прерыванием.

3) Проверил размещение прерывания в памяти. Для этого запустил программу лабораторной работы №3, которая отображает карту памяти в виде списка блоков MCB.

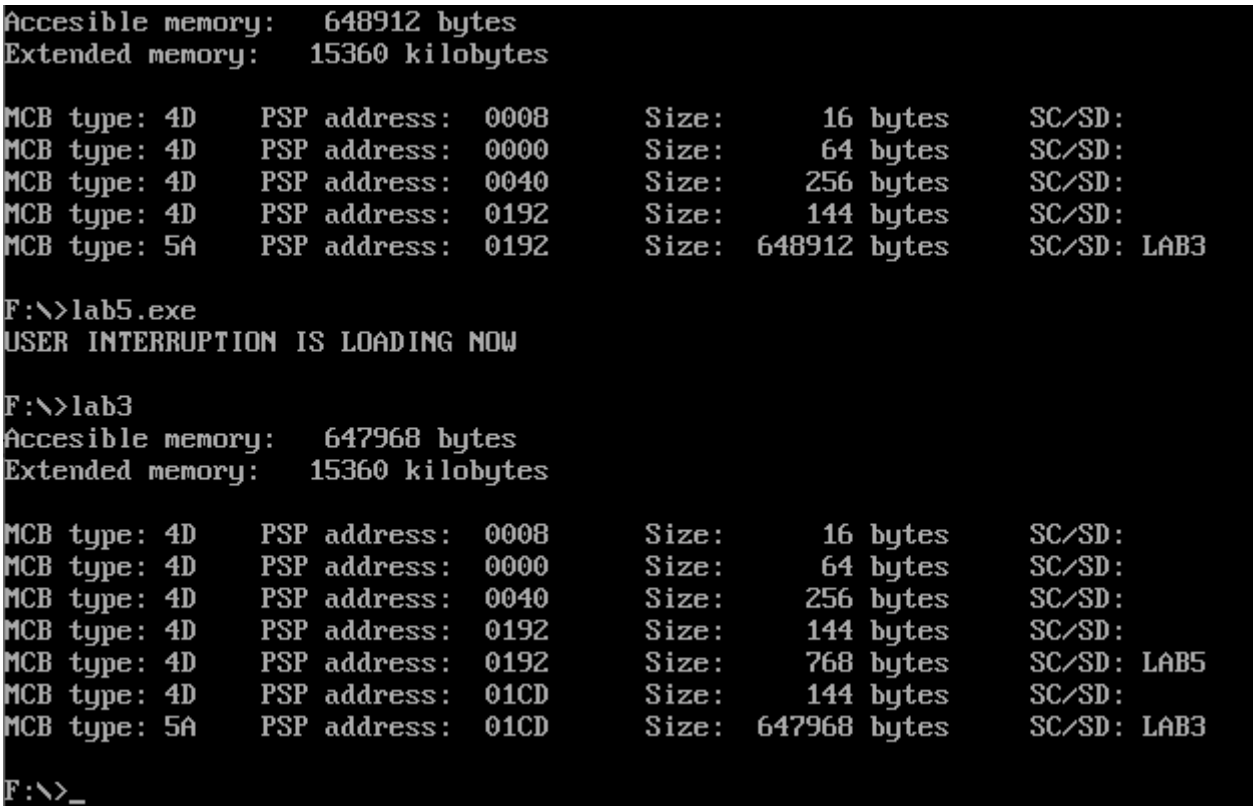

Рис. 2. Запуск программы лабораторной №3 до и после загрузки пользовательского прерывания.

Резидент находится в памяти и используется.

4) Запустил отлаженную программу с ключом выгрузки и убедился, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а занятая резидентом память освобождена. Для этого ещё раз была запущена программа из лабораторной №3. Как видно из рис. 3-4, резидент был успешно выгружен из памяти.

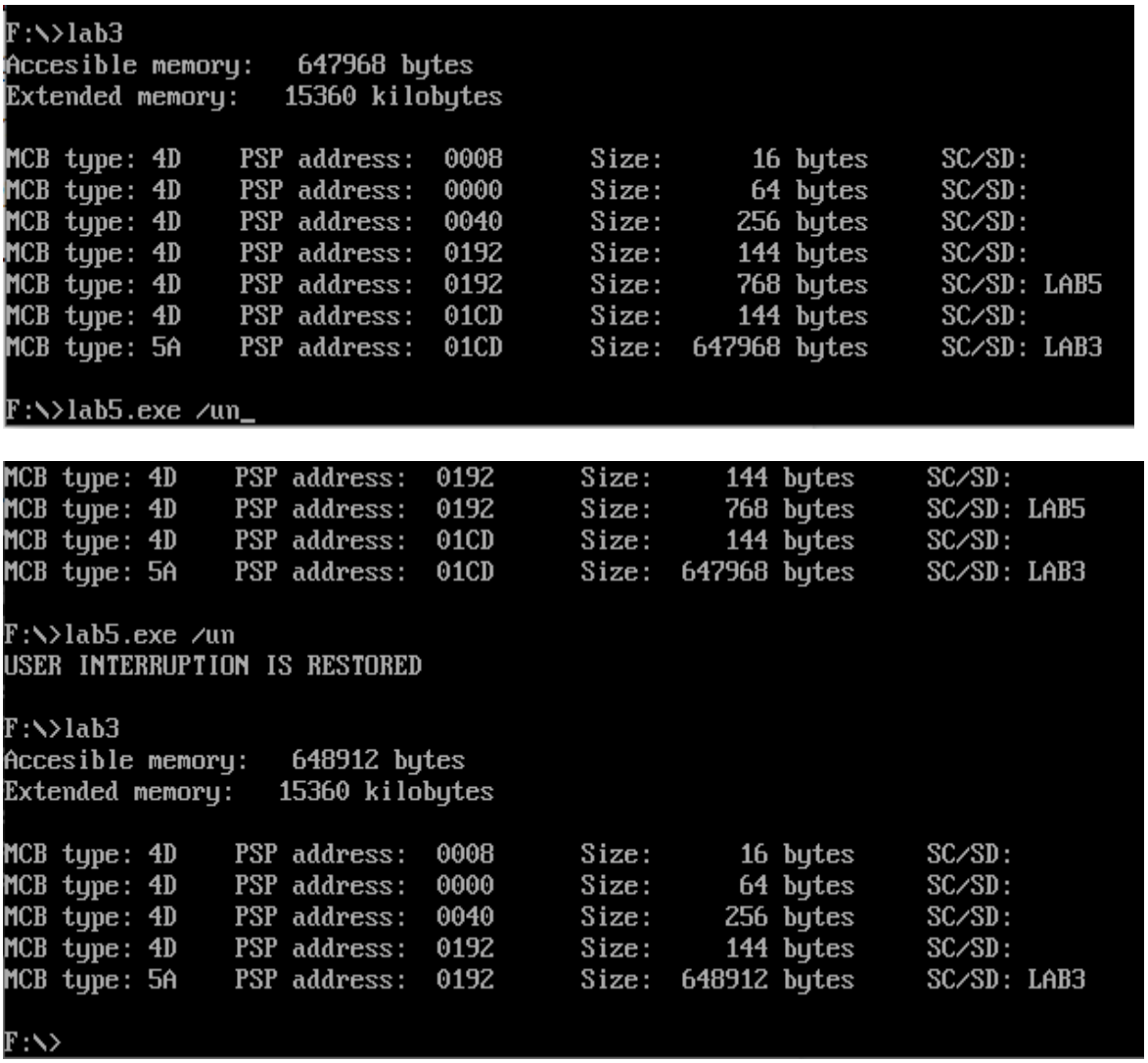

Рис. 3-4. Нахождение резидента в памяти до и после введения ключа выгрузки.

Вывод

В результате выполнения лабораторной работы были изучены возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры и разработано пользовательское прерывание от клавиатуры, которое обрабатывает скан-код, выполняет вывод сообщения результата нажатия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

Результаты исследования проблем:

1. Какого типа прерывания использовались в работе?

Аппаратное прерывание int 09h, генерируемое при каждом нажатии и отпускании клавиши (прерывание функций BIOS).

Программное прерывание int 21h – предназначено для предоставления программисту различных услуг со стороны DOS (прерывание функций DOS).

Прерывание int 16h – интерфейс прикладного уровня с клавиатурой.

Нажатия клавиш на самом деле обрабатываются асинхронно на заднем плане. Когда клавиша получена от клавиатуры, она обрабатывается прерыванием INT 09H и помещается в циклическую очередь. (прерывание функций BIOS).

2. Чем отличается скан код от кода ASCII?

Скан код – код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата.

Код ASCII – код, сопоставляемый печатным и непечатным символам (из таблицы ASCII).

Отличие: скан-код - номер клавиши, а ASCII-код - код, соответствующий обозначению на этой клавише.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ.

Файл lab5.asm.

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

ASTACK SEGMENT STACK
     DW 64 DUP(?)
ASTACK ENDS

CODE SEGMENT
;-----------------------------
WRITE PROC NEAR ;Вывод на экран сообщения
          push ax
          mov  ah, 09h
         int  21h
         pop     ax
         ret
WRITE ENDP
;-----------------------------
USR_INTER PROC FAR
     jmp START_CODE
     ADDR_PSP1   dw 0 ;offset 3
     ADDR_PSP2   dw 0 ;offset 5
     KEEP_IP    dw 0 ;offset 7
     KEEP_CS    dw 0 ;offset 9
     SIGN  dw 0ABCDh ;offset 11
     REQ_KEY_1  db 02h
     REQ_KEY_2  db 03h
     REQ_KEY_3  db 04h
     INT_STACK  dw 64 dup (?)
     KEEP_SS         dw 0
     KEEP_AX         dw 0
     KEEP_SP         dw 0

START_CODE:
     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

mov ax,0040h
mov es,ax
mov al,es:[17h]
and al,00000010b
jnz stand_set

in al,60h ;Считать ключ
cmp al, REQ_KEY_1
je   CHG_1_I

cmp al, REQ_KEY_2
je   CHG_N_T

cmp al, REQ_KEY_3
je   CHG_3_T

mov ss, KEEP_SS
mov sp, KEEP_SP

stand_set:
     pop es
     pop ds
     pop dx
     mov ax, CS:KEEP_AX
     mov sp, CS:KEEP_SP
     mov ss, CS:KEEP_SS
     jmp dword ptr cs:[KEEP_IP]
CHG_1_I:
     mov cl, 'I'
     jmp do_req
CHG_N_T:
     mov cl, 'N'
     jmp do_req
```

```
CHG_3_T:
        mov cl, 'T'
        jmp do_req

do_req:
        in al,61h  ;Взять значение порта управления клавиатурой
        mov ah,al  ;Сохранить его
        or al,80h  ;Установить бит разрешения для клавиатуры
        out 61h,al ;И вывести его в управляющий порт
        xchg ah, al     ;Извлечь исходное значение порта
        out 61h,al ;И записать его обратно
        mov al,20h ;Послать сигнал конца прерывания контроллеру
прерываний 8259
        out 20h,al

        push bx
        push cx
        push dx

        mov ah, 05h ;функция, позволяющая записать символ в буфер
клавиатуры
        mov ch, 00h ;символ в CL уже занесён ранее, осталось
обнулить CH
        int 16h
        or   al, al     ;проверка переполнения буфера
        jnz SKIP   ;если переполнен - идём в skip
        jmp END_OF_USR_INTER ;иначе выходим

SKIP:                ;очищаем буфер
        push es
        push si
        mov ax, 0040h
        mov es, ax
        mov si, 001ah
        mov ax, es:[si]
        mov si, 001ch
        mov es:[si], ax
        pop si
```

```
            pop es

      END_OF_USR_INTER:
            pop dx
            pop cx
            pop bx
            mov ax, KEEP_SS
            mov ss, ax
            mov ax, KEEP_AX
            mov sp, KEEP_SP
            iret
USR_INTER ENDP
;-----------------------------
last_byte:
CHECK_INTSET PROC NEAR     ;Проверка установки прерывания
      push bx
      push dx
      push es

      mov ah, 35h     ;Получение вектора прерываний
      mov al, 09h     ;Функция выдает значение сегмента в ES, смещение в
BX
      int 21h

      mov dx, es:[bx + 11]
      cmp dx, 0ABCDh ;Проверка на совпадение кода прерывания
      je INSTALLED
      mov al, 00h
      jmp END_CHECK_INTSET

INSTALLED: ; процедура вернёт 1 если прерывание установлено
      mov al, 01h
      jmp END_CHECK_INTSET

END_CHECK_INTSET:
      pop es
      pop dx
      pop bx
```

```
     ret
CHECK_INTSET ENDP
;-----------------------------
UN_CHECK PROC NEAR ;Проверка на то, не ввёл ли пользователь /un
     push es
     mov ax, ADDR_PSP1
     mov es, ax

     cmp byte ptr es:[82h], '/'
     jne END_UN_CHECK
     cmp byte ptr es:[83h], 'u'
     jne END_UN_CHECK
     cmp byte ptr es:[84h], 'n'
     jne END_UN_CHECK
     mov al, 1h

END_UN_CHECK:
     pop es
     ret
UN_CHECK ENDP
;-----------------------------
REDEF_INT PROC NEAR ;Сохранение стандартного обработчика прерываний и
загрузка пользовательской версии
     push ax
     push bx
     push dx
     push es
     ; получаем адрес обработчика прерывания (старого) для того чтобы
сохранить
     mov ah, 35h ;функция получения вектора
     mov al, 09h ; номер вектора
     int 21h
     ; на выходе в ES:BX = адрес обработчика прерывания
     ;возвращает значение вектора прерывания для INT (AL);
     mov KEEP_IP, bx ;Запоминаем смещение и сегмент
     mov KEEP_CS, es

     push ds
```

```
	lea dx, USR_INTER
	mov ax, seg USR_INTER
	mov ds, ax
	mov ah, 25h ; функция установки вектора
	mov al, 09h ; номер вектора
	int 21h     ; меняем прерывание
	pop ds

	lea dx, LOAD_MES
	call WRITE

	pop es
	pop dx
	pop bx
	pop ax

	ret
REDEF_INT ENDP
;-----------------------------
RESTORE_INT PROC NEAR;Выгрузка обработчика прерывания (восстановленение старого)
	push ax
	push bx
	push dx
	push es

	mov ah, 35h
	mov al, 09h
	int 21h

	cli;сбрасывает флаг прерывания в регистре флагов.
	;Когда этот флаг сброшен, процессор игнорирует все прерывания (кроме NMI)от внешних устройств
	push ds
	mov dx, es:[bx + 7]
	mov ax, es:[bx + 9]
	mov ds, ax
	mov ah, 25h
```

```
      mov al, 09h
      int 21h
      pop ds
      sti

      lea dx, UNLOAD_MES
      call WRITE

      push es ;Удаление MCB
      mov cx,es:[bx+3]
      mov es,cx
      mov ah,49h ; Освободить распределенный блок памяти
      int 21h

      pop es
      mov cx,es:[bx+5]
      mov es,cx ; es - сегментный адрес (параграф) освобождаемого блока
памяти
      int 21h

      pop es
      pop dx
      pop bx
      pop ax

      mov ah, 4Ch     ;Выход из программы через функцию 4C
      int 21h
      ret
RESTORE_INT ENDP
;---------------------------
MAIN  PROC FAR
    mov bx,2Ch
      mov ax,[bx]
      mov ADDR_PSP2,ax
      mov ADDR_PSP1,ds  ;сохранение PSP
      mov dx, ds
      xor ax,ax
      xor bx,bx
```

```
      mov ax,data
      mov ds,ax
      xor dx, dx

      call UN_CHECK ;Проверка на введение /un
      cmp al, 01h
      je TRY_TO_UNLOAD


      call CHECK_INTSET  ;Проверка не является ли программа резидентной
      cmp al, 01h
      jne NEED_TO_REDEF

ALREADY_INSTALLED:
      lea dx, ALR_LOADED_MES ;Программа уже загружена
      call WRITE
      jmp END_OF_MAIN

;Загрузка пользовательского прерывания
NEED_TO_REDEF:
      call REDEF_INT
      lea dx, last_byte
      mov cl, 04h
      shr dx, cl
      add dx, 1Bh
      mov ax, 3100h
      int 21h

;Выгрузка  пользовательского прерывания
TRY_TO_UNLOAD:
      call CHECK_INTSET
      cmp al, 1h
      jne NOT_LOADED
      call RESTORE_INT
      jmp END_OF_MAIN

;Прерывание выгружено
NOT_LOADED:
```

```
      lea dx, NOT_LOADED_MES
      call WRITE

END_OF_MAIN:
      mov ah, 4Ch
      int 21h
MAIN       ENDP
CODE  ENDS

DATA SEGMENT
      LOAD_MES   db 'USER INTERRUPTION IS LOADING NOW', 0dh, 0ah, '$'
    NOT_LOADED_MES db 'USER INTERRUPTION IS NOT LOADED', 0dh, 0ah, '$'
      ALR_LOADED_MES db 'USER INTERRUPTION IS ALREADY LOADED', 0dh, 0ah,
'$'
      UNLOAD_MES      db 'USER INTERRUPTION IS RESTORED', 0dh, 0ah, '$'
DATA ENDS

END Main
```