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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

Студент гр. 8383 _____ Мололкин К.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы

Исследовать возможность встраивать пользовательский обработчик прерываний в стандартный обработчик клавиатуры.

Ход работы

Был написан и отлажен программный модуль .EXE, который проверяет если параметр /un был передан через командную строку, и записывает соответствующий результат в переменную-флаг IS_UN_FLAG, затем проверяет установлено ли пользовательское прерывание с вектором 09h, если установлено, то проверяется значение IS_UN_FLAG, если оно 1, то прерывание выгружается, если 0, то выводится сообщение о попытке повторно загрузить прерывание, если прерывание не установлено и значение флага 1, то выводится сообщение о попытке выгрузить неустановленное прерывание, если 0, то устанавливает резидентную функцию по обработке прерывания и настраивает вектор прерывания. После нажатия на клавишу 1, выводится символ ‘!’, после нажатия 2, ‘@’, после 3, ‘#’. Пример работы программы представлен на рис. 1.

Рисунок 1 – пример работы программы

На следующем шаге было проверено размещение прерывания в памяти с помощью программы LR3_1.COM, для этого сначала была запущена программа LR5.EXE, а затем LR3_1.COM, результат работы двух программ представлен на рис. 2.

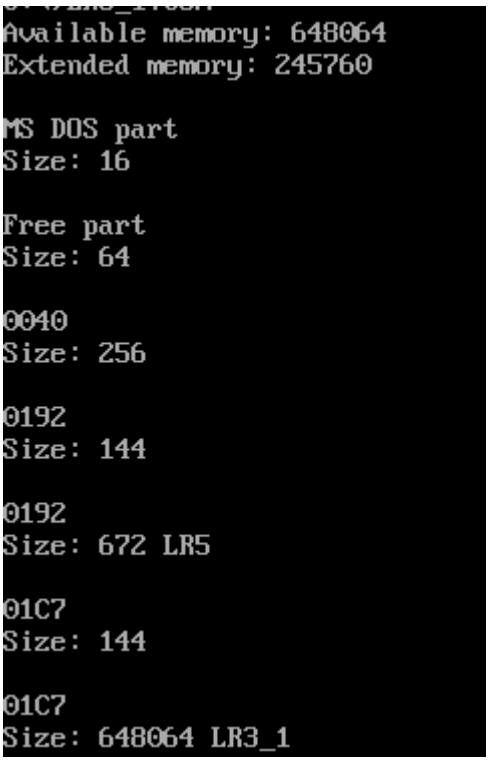

Рисунок 2 – проверка размещения прерывания в памяти

Как видно из рисунка, программа LR4.EXE, размещается в 5 по счету блоке.

Для проверки того, что программа определяет установленный обработчик прерывания, программа LR5.EXE еще раз. Результат повторного запуска представлен на рис. 3.

Рисунок 3 – проверка программы на определение запущенного прерывания

Затем для проверки выгрузки резидентного обработчика прерывания, программа была запущена с ключом выгрузки “/un”. На рис.4 представлен результат запуска программы с ключом, а на рис. 5 представлен результат запуска программы LR3_1.COM, для проверки выгрузки программы из памяти. Код программы LR5.ASM представлен в приложении А.

Рисунок 4 – результат выгрузки программы

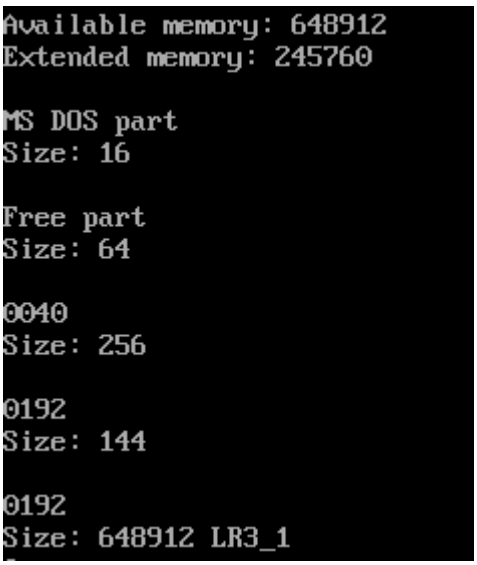

Рисунок 5 – результат выгрузки программы из памяти

Контрольные вопросы

1. Какого типа прерывания использовались в работе?
   В программе используются аппаратное (9Ch) и программные (16h, 21h) прерывания.
2. Чем отличается скан код от кода ASCII?
   Скан-код – однобайтное число, младшие 7 битов которого представляют идентификационный номер, присвоенный каждой клавише. А ASCII код это таблица в которой символам сопоставлены числовые коды.

Вывод

Во время выполнения лабораторной работы были исследована возможность встраивать пользовательский обработчик прерываний в стандартный обработчик клавиатуры.

Приложение А

```
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:ASTACK, ES:NOTHING


ROUT PROC FAR
     jmp START_ROUT
     INTER_ID dw 7373h
     KEEP_IP dw 0
     KEEP_CS dw 0
     KEEP_SS dw 0
    KEEP_SP dw 0
    KEEP_AX dw 0
     KEEP_PSP dw 0
    ROUT_STACK dw 128 DUP(0)

START_ROUT:
    mov KEEP_SS, ss
    mov KEEP_SP, sp
    mov KEEP_AX, ax
    mov ax, seg ROUT_STACK
    mov ss, ax
    mov ax, offset ROUT_STACK
     add ax, 256
     mov sp, ax

    push ax
    push bx
    push ds
    push dx
    push si
    push cx
    push es

    in al, 60h
    cmp al, 02h
    je ONE
    cmp al, 03h
    je TWO
    cmp al, 04h
    je THREE
    pushf
    call DWORD PTR CS:KEEP_IP
    jmp END_ROUT

ONE:
    mov cl, '!'
    jmp CHOOSEN_BUTTON
TWO:
    mov cl, '@'
```

```
    jmp CHOOSEN_BUTTON
THREE:
    mov cl, '#'

CHOOSEN_BUTTON:
    in al, 61h
    mov ah, al
    or al, 80h
    out 61h, al
    xchg al, al
    out 61h, al
    mov al, 20h
    out 20h, al

PRINT_SYMB:
     mov ah, 05h
     mov ch, 00h
     int 16h
     or al, al
     jz END_ROUT

    mov ax, 0040h
    mov es, ax
    mov ax, es
    mov es:[1ch], ax
    jmp PRINT_SYMB

END_ROUT:
    pop es
    pop cx
    pop si
    pop dx
    pop ds
    pop bx
    pop ax
    mov sp, KEEP_SP
    mov ax, KEEP_AX
    mov ss, KEEP_SS
    mov al, 20h
    out 20h, al
    IRET
ROUT ENDP
ROUT_END:

START_OR_STOP_INTER PROC
    push ax
    push bx
    push si
    push es
     mov ax, KEEP_PSP
     mov es, ax
     cmp byte ptr es:[82h], '/'
     jne IS_LOAD
```

```
     cmp byte ptr es:[83h], 'u'
     jne IS_LOAD
     cmp byte ptr es:[84h], 'n'
    jne IS_LOAD
    mov IS_UN_FLAG, 1

IS_LOAD:
    mov ah, 35h
    mov al, 09h
    int 21h
    mov si, offset INTER_ID
    sub si, offset ROUT
    mov ax, es:[bx+si]
    cmp ax, 7373h
    jne NOT_LOAD
    cmp IS_UN_FLAG, 1
    jne LOAD_AGAIN
    call STOP_INTER
    jmp END_SS

LOAD_AGAIN:
    mov dx, offset TRY_LOAD_AGAIN
    call PRINT_STRING
    jmp END_SS

NOT_LOAD:
    cmp IS_UN_FLAG, 1
    jne BEGIN_ROUT
    mov dx, offset TRY_TO_STOP
    call PRINT_STRING
    jmp END_SS

BEGIN_ROUT:
    call LOAD_INTER

END_SS:

    pop es
    pop si
    pop bx
    pop ax
    ret
START_OR_STOP_INTER ENDP

LOAD_INTER PROC
    push ax
     push bx
     push cx
     push dx
     push ds
     push es

     mov ah, 35h
```

```
     mov al, 09h
     int 21h
     mov KEEP_CS, es
     mov KEEP_IP, bx
     push ds
     mov dx, offset ROUT
     mov ax, SEG ROUT
     mov ds, ax
     mov ah, 25h
     mov al, 09h
     int 21h
     pop ds
     mov dx, offset ROUT_END
     add dx, 10Fh
     mov cl, 4h
     shr dx, cl
     inc dx
     xor ax, ax
     mov ah, 31h
     int 21h

     pop es
     pop ds
     pop dx
     pop cx
     pop bx
     pop ax
     ret
LOAD_INTER ENDP

STOP_INTER PROC
    CLI
     push ax
     push bx
     push dx
     push ds
     push es
     push si

     mov ah, 35h
     mov al, 09h
     int 21h
     mov si, offset KEEP_IP
     sub si, offset ROUT
     mov dx, es:[bx + si]
     mov ax, es:[bx + si + 2]
     push ds
     mov ds, ax
     mov ah, 25h
     mov al, 09h
     int 21h
     pop ds
     mov ax, es:[bx + si + 4]
```

```
     mov es, ax
     push es
     mov ax, es:[2Ch]
     mov es, ax
     mov ah, 49h
     int 21h
     pop es
     mov ah, 49h
     int 21h

     pop si
     pop es
     pop ds
     pop dx
     pop bx
     pop ax
     STI
     ret
STOP_INTER ENDP

PRINT_STRING PROC near
    push AX
    mov ah, 09h
     int 21h
    pop AX
    ret
PRINT_STRING ENDP

MAIN PROC
    xor ax, ax
    mov ax, DATA
    mov ds, ax
    mov KEEP_PSP, es
    call START_OR_STOP_INTER
    xor al, al
     mov ah, 4Ch
     int 21h
MAIN ENDP

CODE ENDS

ASTACK SEGMENT STACK
    dw 128 DUP(0)
ASTACK ENDS

DATA SEGMENT
    IS_UN_FLAG db 0
    TRY_TO_STOP db "Trying to stop not loaded interruption$"
    TRY_LOAD_AGAIN db "Trying to load interruption again$"
DATA ENDS
END MAIN
```