

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

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

Студентка гр. 9383

Лапина А.А.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2021

Цель работы.

Исследовать возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

Постановка задачи.

Необходимо написать и отладить программный модуль типа .EXE, который выполняет следующие функции:

1. проверяет, установлено ли пользовательское прерывание с вектором 09h;
2. устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерывания, если прерывание не установлено;
3. если прерывание установлено, то выводится соответствующее сообщение;
4. выгружает прерывания по соответствующему значению параметра в командные строки '/un'.

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

1. сохранять значения регистров в стеке при входе и восстанавливать их при выходе;
2. анализировать скан-код;
3. записывать требуемый код в буфер клавиатуры, если этот код совпадает с одним из заданных;
4. осуществлять передачу управления стандартному обработчику прерываний, если этот код не совпадает ни с одним из заданных.

Запустить отлаженную программу и убедиться, что резидентный обработчик прерывания 09h установлен. Работа прерывания проверяется введением различных символов, обрабатываемых установленным и стандартными обработчиками. Проверить размещение прерывания в памяти.

Запустить отлаженную программу ещё раз и убедиться, что программа определяет установленный обработчик прерываний. Запустить отлаженную программу с ключом выгрузки и убедиться, что резидентный обработчик прерывания выгружен

Выполнение работы.

Для выполнения данной работы были реализованы следующие функции:

inter – код резидентного обработчика прерывания 1Ch;

load - для загрузки обработчика прерываний;

unload - для выгрузки обработчика прерываний;

isParam - для проверки наличия ключа выгрузки;

isLoad - для проверки установки обработчика прерываний;

print - для вывода строки, адрес которой лежит в регистре DX;

main - для выполнения поставленной в данной лабораторной работе задачи.

Для вывода информации на экран были созданы следующие строки:

ls, хранящая в себе строку 'Interrupt loaded successfully\$';

us, хранящая в себе строку 'Interrupt unloaded successfully\$';

ial, хранящая в себе строку 'Interrupt already loaded\$';

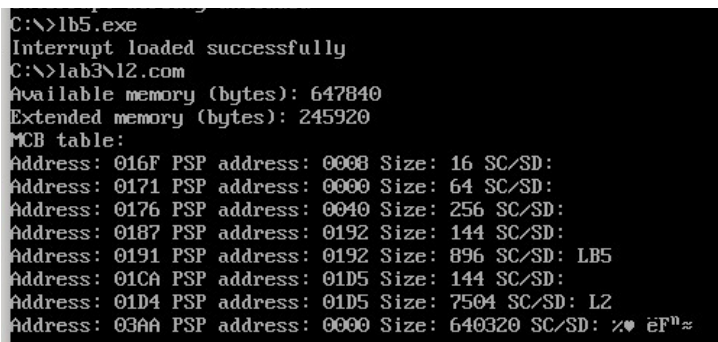
iau, хранящая в себе строку 'Interrupt already unloaded\$'.

При запуске программы с помощью процедуры isParam проверяется наличие ключа выгрузки '/un'. В случае, если его нет, переменная flag остаётся равной 0, в противном случае – становится равной 1. В зависимости от значения этой переменной будет происходить загрузка или выгрузка резидентной функции. В случае, если ключа нет, но программа уже загружена, что проверяется так же с помощью переменной flag после вызова процедуры isLoad, выводится сообщение о том, что обработчик уже загружен, и программа

завершается. Если же обработчик не был загружен, то вызывается процедура load, которая устанавливает резидентную функцию и настраивает вектор прерываний. В случае попытки выгрузки вновь вызывается процедура isLoad и проверяется переменная flag. Если резидентная программа не была установлена, выводится соответствующее сообщение, в противном случае – вызывается процедура unload, которая восстанавливает необходимые регистры, вектор прерываний и освобождает память.

Резидентная функция в начале своего выполнения сохраняет в переменных регистры SS, SP и AX (последний используется в создании собственного стека, поэтому значение сохраняется в переменную), создаёт свой стек и сохраняет все регистры в него. Затем считывает скан-код и сравнивает его с требуемым (цифры от 0 до 9). В случае, если коды разные, вызывается стандартный обработчик прерывания, в противном случае – посылается сигнал подтверждения микропроцессору клавиатуры. Введенная цифра выводится на 1 меньше. В конце работы резидентной функции все регистры восстанавливаются

Результаты работы программы представлены на рисунках 1-4. Для проверки установки обработчика прерываний используется программа L2.COM из третьей лабораторной работы.



```
C:\>lb5.exe
Interrupt loaded successfully
C:\>lab3\l2.com
Available memory (bytes): 647840
Extended memory (bytes): 245920
MCB table:
Address: 016F PSP address: 0008 Size: 16 SC/SD:
Address: 0171 PSP address: 0000 Size: 64 SC/SD:
Address: 0176 PSP address: 0040 Size: 256 SC/SD:
Address: 0187 PSP address: 0192 Size: 144 SC/SD:
Address: 0191 PSP address: 0192 Size: 896 SC/SD: LB5
Address: 01CA PSP address: 01D5 Size: 144 SC/SD:
Address: 01D4 PSP address: 01D5 Size: 7504 SC/SD: L2
Address: 03AA PSP address: 0000 Size: 640320 SC/SD: %♥ ēFⁿ≈
```

Рисунок 1 – Проверка размещения программы в памяти при загрузке

```

C:\>lb5.exe /un
Interrupt unloaded successfully
C:\>lab3\l2.com
Available memory (bytes): 648912
Extended memory (bytes): 245920
MCB table:
Address: 016F PSP address: 0008 Size: 16 SC/SD:
Address: 0171 PSP address: 0000 Size: 64 SC/SD:
Address: 0176 PSP address: 0040 Size: 256 SC/SD:
Address: 0187 PSP address: 0192 Size: 144 SC/SD:
Address: 0191 PSP address: 0192 Size: 6432 SC/SD: L2
Address: 0324 PSP address: 0000 Size: 642464 SC/SD: 2U-^n&

```

Рисунок 2 –

Проверка очистки памяти при выгрузке программы

```

C:\>lb5.exe
Interrupt loaded successfully
C:\>0123456789

```

Рисунок 3 – Результат обработки установленным обработчиком прерываний — числа на 1 меньше (вводилось 1234567890)

```

C:\>lb5.exe
Interrupt already loaded
C:\>qwerty_

```

Рисунок 4 – Результат обработки стандартным обработчиком прерываний - буквы не меняет (вводилось qwerty)

Выводы.

В ходе работы были исследованы возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

В ходе выполнения работы были даны ответы на следующие контрольные вопросы.

1. Какого типа прерывания использовались в работе?

В данной работе использовались аппаратное прерывание клавиатуры 09h и прерывание DOS 21h.

2. Чем отличается скан-код от кода ASCII?

Скан-код – это код, присвоенный каждой клавише, с помощью которого определяется, какая клавиша была нажата. ASCII-код – это код печатного символа.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5.asm

```
stack segment stack
    db 256 dup (?)
stack ends

data segment
flag db 0

Is db 'Interrupt loaded successfully$'      ;успешно загружено
us db 'Interrupt unloaded successfully$'    ;успешно выгружено
ial db 'Interrupt already loaded$'
iau db 'Interrupt already unloaded$'

data ends

code segment
assume CS:code, DS:data

inter proc far
    jmp ibegin
PSP dw ?
keepIP dw 0
keepCS dw 0
ID dw 0FFFFh
keepSS dw ?
keepSP dw ?
keepAX dw ?
vector dd 0
istk dw 32 dup (?)
istkend dw ?
; --- Begin of the resident program ---
ibegin: mov keepSS, SS
        mov keepSP, SP
        mov keepAX, AX
        mov AX, CS
        mov SS, AX
        mov SP, offset istkend
        push BX
        push CX
```

```

    push    DX
    push    DS
    push    ES
    push    SI
    push    DI
    push    BP
; --- Getting the scan code ---
    in      AL, 60h
    cmp     AL, 02h
    jl      stdrt
    cmp     AL, 0Bh
    jle     doreq
stdrt: mov     AX, keepIP
      mov     word ptr vector, AX
      mov     AX, keepCS
      mov     word ptr vector + 2, AX
      pop     BP
      pop     DI
      pop     SI
      pop     ES
      pop     DS
      pop     DX
      pop     CX
      pop     BX
      mov     AX, keepSS
      mov     SS, AX
      mov     AX, keepAX
      mov     SP, keepSP
      jmp     CS:[vector]
doreq: push    AX
      in      AL, 61h
      mov     AH, AL
      or      AL, 80h
      out     61h, AL
      xchg    AH, AL
      out     61h, AL
      mov     AL, 20h
      out     20h, AL
; --- The processing ---
      pop     AX
      xor     CX, CX
      mov     CL, AL
      add     CL, 2Eh

```



```

write: mov    AH, 05h
       mov    CH, 00h
       int    16h
       or     AL, AL
       jnz    skip
       jmp    ifin
skip:  mov    AH, 0Ch
       mov    AL, 0
       int    21h
       jmp    write
; --- End of the resident program ---
ifin:  pop     BP
       pop     DI
       pop     SI
       pop     ES
       pop     DS
       pop     DX
       pop     CX
       pop     BX
       mov     AX, keepSS
       mov     SS, AX
       mov     AX, keepAX
       mov     SP, keepSP
       iret
iend:
inter  endp

load  proc
       push    AX
       push    CX
       push    DX
; --- Storing offset and segment ---
       mov     AH, 35h
       mov     AL, 09h
       int     21h
       mov     keepIP, BX
       mov     keepCS, ES
; --- Interrupt setting ---
       push    DS
       mov     DX, offset inter
       mov     AX, seg inter
       mov     DS, AX
       mov     AH, 25h

```

```

        mov     AL, 09h
        int     21h
        pop     DS
; --- Resident program preservation ---
        mov     DX, offset iend
        mov     CL, 4
        shr     DX, CL
        inc     DX
        mov     AX, CS
        sub     AX, PSP
        add     DX, AX
        xor     AX, AX
        mov     AH, 31h
        int     21h
        pop     DX
        pop     CX
        pop     AX
        ret
load    endp

```

```

unload proc
        push    AX
        push    DX
        push    SI
        push    ES
; --- Recovery offset and segment ---
        cli
        push    DS
        mov     AH, 35h
        mov     AL, 09h
        int     21h
        mov     SI, offset keepIP
        sub     SI, offset inter
        mov     DX, ES:[BX+SI]
        mov     AX, ES:[BX+SI+2]
        mov     DS, AX
        mov     AH, 25h
        mov     AL, 09h
        int     21h
        pop     DS
        mov     AX, ES:[BX+SI-2]
        mov     ES, AX
        push    ES

```

```

    mov     AX, ES:[2Ch]
    mov     ES, AX
    mov     AH, 49h
    int     21h
    pop     ES
    mov     AH, 49h
    int     21h
    sti
    pop     ES
    pop     SI
    pop     DX
    pop     AX
    ret
unload endp

```

```

isParam proc
    push    AX
    mov     AL, ES:[82h]
    cmp     AL, '/'
    jne     nparam
    mov     AL, ES:[83h]
    cmp     AL, 'u'
    jne     nparam
    mov     AL, ES:[84h]
    cmp     AL, 'n'
    jne     nparam
    mov     flag, 1
nparam: pop    AX
    ret
isParam endp

```

```

isLoad proc
    push    AX
    push    DX
    push    SI
    mov     flag, 1
    mov     AH, 35h
    mov     AL, 09h
    int     21h
    mov     SI, offset ID
    sub     SI, offset inter
    mov     DX, ES:[BX+SI]
    cmp     DX, 0FFFFh

```

```

        je    ld
        mov   flag, 0
ld:     pop   SI
        pop   DX
        pop   AX
        ret
isLoad endp

print proc
        push  AX
        mov   AH, 09h
        int   21h
        pop   AX
        ret
print endp

main    proc far
        mov   AX, data
        mov   DS, AX
        mov   PSP, ES
        mov   flag, 0
        call  isParam
        cmp   flag, 1
        je    un
; --- Loading ---
        call  isLoad
        cmp   flag, 0
        je    notld
        mov   DX, offset ial
        call  print
        jmp   fin
notld:  mov   DX, offset ls
        call  print
        call  load
        jmp   fin
; --- Unloading ---
un:     call  isLoad
        cmp   flag, 0
        jne   alrld
        mov   DX, offset iau
        call  print
        jmp   fin
alrld:  call  unload

```

```
        mov    DX, offset us
        call   print
; --- End ---
fin:    mov    AX, 4C00h
        int    21h
main    endp
code    ends
        end    main
```