**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 |  | Шишкин И.В. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

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

**Ход работы.**

Был написан программный модуль типа .ЕХЕ, который выполняет следующие функции:

1. Проверяет, установлено ли пользовательское прерывание с вектором 09h.
2. Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.
3. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Сh 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 4Сh прерывания int 21h.

Обработчик прерываний обрабатывает клавиши цифр с 1 до 0 (скан-коды с 02h до 0Ah). Рисунок скан-кодов клавиш изображен на рис. 1. После нажатия на экран выводится сумма предыдущего числа и введенного. Например, если сначала нажать 2, а потом 3, то выведется 5. Если после сложения число оказывается двузначным, то от него отнимается 10. Так, если нажать 5, а потом 8, то выведется 3. Пример работы программы приведен на рис. 2 и рис. 3. Если же была нажата другая клавиша, то управление передается стандартному обработчику прерывания.

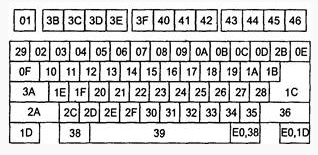


Рисунок 1 – Скан-коды клавиш



Рисунок 2 – После нажатия сначала 3, а потом 4



Рисунок 3 – После нажатия сначала 3, затем 4, а потом 6

С помощью программы из ЛР3 было выведено состояние памяти: на рис. 4 – до загрузки прерывания, на рис. 5 – после загрузки прерывания, на рис. 6 – после выгрузки прерывания.

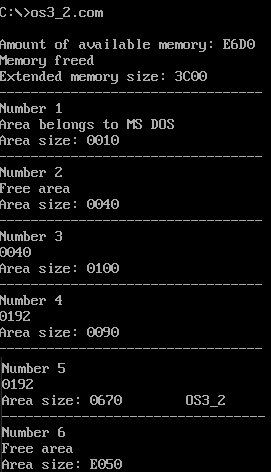


Рисунок 4 – До загрузки прерывания

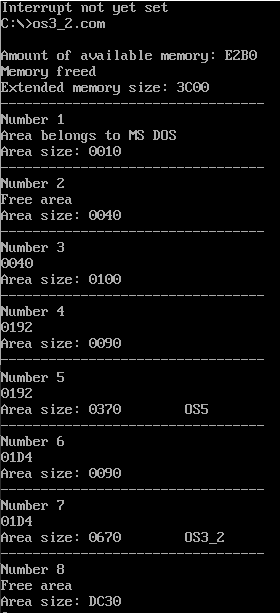


Рисунок 5 – После загрузки прерывания

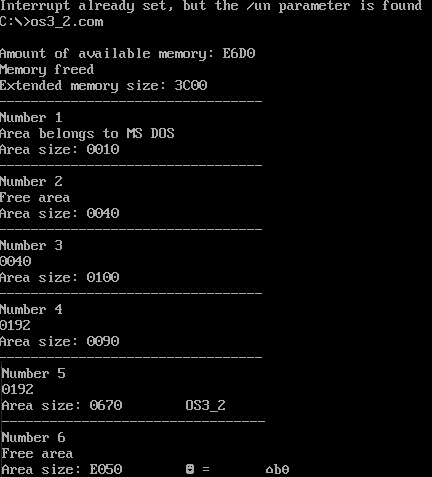


Рисунок 6 – После выгрузки прерывания

Как видно из рисунков, после выгрузки прерывания, блоки, соответствующие программе ЛР5, удаляются.

**Контрольные вопросы.**

1. Какого типа прерывания использовались в работе?

Сервис DOS int 21h.

Прерывание от клавиатуры int 09h. Была написана резидентная программа, которая перехватывает int 09h и проверяет на определенный ключ.

1. Чем отличается скан код от кода ASCII?

Скан-код – код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата. ASCII код же, в свою очередь, является не кодом клавиши, а кодом символа.

**Выводы.**

В ходе выполнения лабораторной работы была реализована программа, загружающая и выгружающая пользовательское прерывание от клавиатуры.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack, ES:NOTHING

ROUT PROC FAR

jmp INTERRUPT\_BEGIN

ADDITION\_RESULT dw 0

CHAR db 0

INTERRUPT\_ID dw 9888h

KEEP\_AX dw 0

KEEP\_SS dw 0

KEEP\_SP dw 0

KEEP\_IP dw 0

KEEP\_CS dw 0

KEEP\_PSP DW 0

INTERRUPTION\_STACK dw 128 dup(0)

INTERRUPT\_BEGIN:

mov KEEP\_SS, SS

mov KEEP\_SP, SP

mov KEEP\_AX, AX

mov AX, SEG INTERRUPTION\_STACK

mov SS, AX

mov AX, offset INTERRUPTION\_STACK

add AX, 256 ; на конец стека

mov SP, AX

push BX

push CX

push DX

push SI

push DS

push ES

mov AX, SEG CHAR

mov DS, AX

in AL, 60h ;читать ключ

cmp AL, 02h

jne CONTINUE\_FROM\_1

mov AX, ADDITION\_RESULT ;1

add AX, 1

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_1:

cmp AL, 03h

jne CONTINUE\_FROM\_2

mov AX, ADDITION\_RESULT ;2

add AX, 2

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_2:

cmp AL, 04h

jne CONTINUE\_FROM\_3

mov AX, ADDITION\_RESULT ;3

add AX, 3

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_3:

cmp AL, 05h

jne CONTINUE\_FROM\_4

mov AX, ADDITION\_RESULT ;4

add AX, 4

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_4:

cmp AL, 06h

jne CONTINUE\_FROM\_5

mov AX, ADDITION\_RESULT ;5

add AX, 5

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_5:

cmp AL, 07h

jne CONTINUE\_FROM\_6

mov AX, ADDITION\_RESULT ;6

add AX, 6

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_6:

cmp AL, 08h

jne CONTINUE\_FROM\_7

mov AX, ADDITION\_RESULT ;7

add AX, 7

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

IF\_ADD\_RES\_GREATER\_9: ;если в результате хранится двузначное число

add AX, -10

jmp DO\_REQ

CONTINUE\_FROM\_7:

cmp AL, 09h

jne CONTINUE\_FROM\_8

mov AX, ADDITION\_RESULT ;8

add AX, 8

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_8:

cmp AL, 0Ah

jne CONTINUE\_FROM\_9

mov AX, ADDITION\_RESULT

add AX, 9

cmp AX, 9

jg IF\_ADD\_RES\_GREATER\_9

jmp DO\_REQ

CONTINUE\_FROM\_9:

cmp AL, 0Bh

jne IF\_NO\_CORRECT\_SYM

mov AX, ADDITION\_RESULT

jmp DO\_REQ

IF\_NO\_CORRECT\_SYM:

pushf ;уйти на исходный обработчик

call DWORD PTR CS:KEEP\_IP

jmp END\_OF\_INT

DO\_REQ: ;следующий код необходим для обработки аппаратного прерывания

mov ADDITION\_RESULT, AX

add AL, 30h ;перевести число в символ

mov CHAR, AL

;int 29h ;вывести его на экран!

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 AL, AL ;извлечь исходное значение порта

out 61h, AL ;и записать его обратно

mov AL, 20h ;послать сигнал "конец прерывания"

out 20h, AL ;контроллеру прерываний 8259

WRITE\_ANS:

mov AH, 05h

mov CL, CHAR

mov CH, 00h

int 16h

or AL, AL

jz END\_OF\_INT

mov AX, 0040h

mov ES, AX

mov AX, ES:[1Ah]

mov ES:[1Ch], AX

jmp WRITE\_ANS

END\_OF\_INT:

pop ES

pop DS

pop SI

pop DX

pop CX

pop BX

mov SP, KEEP\_SP

mov AX, KEEP\_SS

mov SS, AX

mov AX, KEEP\_AX

mov AL, 20h

OUT 20h, AL

IRET

ROUT ENDP

LAST\_BYTE:

;--------------------------------------------------

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

;----------------------------------------------

SET\_INTERRUPT PROC near

push AX

push BX

push CX

push DX

push DS

push ES

mov AH, 35H ; функция получения вектора

mov AL, 09H ; номер вектора

int 21H

mov KEEP\_IP, BX ; запоминание смещения

mov KEEP\_CS, ES ; и сегмента

CLI

push DS

mov DX, offset ROUT

mov AX, seg ROUT

mov DS, AX

mov AH, 25H

mov AL, 09H

int 21H ; восстанавливаем вектор

pop DS

STI

mov DX, offset LAST\_BYTE

add DX, 10Fh

mov CL, 4h ; перевод в параграфы

shr DX, CL

inc DX ; размер в параграфах

xor AX, AX

mov AH, 31h

int 21h

pop ES

pop DS

pop DX

pop CX

pop BX

pop AX

ret

SET\_INTERRUPT ENDP

;----------------------------------------------

INTERRUPT\_UPLOAD PROC near

push AX

push BX

push DX

push DS

push ES

push SI

CLI

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

STI

pop SI

pop ES

pop DS

pop DX

pop BX

pop AX

ret

INTERRUPT\_UPLOAD ENDP

;----------------------------------------------

CHECK\_PARAMETER PROC near

push AX

push ES

mov AX, KEEP\_PSP

mov ES, AX

cmp byte ptr ES:[81h+1], '/'

jne END\_OF\_PARAMETER

cmp byte ptr ES:[81h+2], 'u'

jne END\_OF\_PARAMETER

cmp byte ptr ES:[81h+3], 'n'

jne END\_OF\_PARAMETER

mov PARAMETER, 1

END\_OF\_PARAMETER:

pop ES

pop AX

ret

CHECK\_PARAMETER ENDP

;----------------------------------------------

CHECK\_09H PROC near

push AX

push BX

push SI

mov AH, 35h

mov AL, 09h

int 21h

mov SI, offset INTERRUPT\_ID

sub SI, offset ROUT

mov AX, ES:[BX+SI]

cmp AX, 9888h

jne END\_OF\_CHECK

mov IS\_INTERRUPT\_LOADED, 1

END\_OF\_CHECK:

pop SI

pop BX

pop AX

ret

CHECK\_09H ENDP

;----------------------------------------------

BEGIN PROC FAR

mov AX, DATA

mov DS, AX

mov KEEP\_PSP, ES

call CHECK\_09H

call CHECK\_PARAMETER

mov AL, PARAMETER

cmp AL, 1

je IF\_UN

mov AL, IS\_INTERRUPT\_LOADED

cmp AL, 1

jne IF\_NEED\_TO\_SET\_INTERRUPT

mov DX, offset IF\_INTERRUPT\_SET

call PRINT

jmp ENDD

IF\_NEED\_TO\_SET\_INTERRUPT:

mov DX, offset IF\_INTERRUPT\_NOTSET

call PRINT

call SET\_INTERRUPT

jmp ENDD

IF\_UN:

mov AL, IS\_INTERRUPT\_LOADED

cmp AL, 1

jne IF\_1CH\_NOT\_SET

mov DX, offset STR\_UN

call PRINT

call INTERRUPT\_UPLOAD

jmp ENDD

IF\_1CH\_NOT\_SET:

mov DX, offset IF\_INTERRUPT\_NOTSET

call PRINT

ENDD:

xor AL, AL

mov AH, 4Ch

int 21h

BEGIN ENDP

CODE ENDS

AStack SEGMENT STACK

dw 128 dup(0)

Astack ENDS

DATA SEGMENT

IS\_INTERRUPT\_LOADED db 0

PARAMETER db 0

IF\_INTERRUPT\_SET db 'Interrupt already set $'

IF\_INTERRUPT\_NOTSET db 'Interrupt not yet set $'

STR\_UN db 'Interrupt already set, but the /un parameter is found $'

DATA ENDS

END BEGIN