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

ОТЧЕТ

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

Студентка гр. 8382 _____ Кузина А.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

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

Ход выполнения работы.

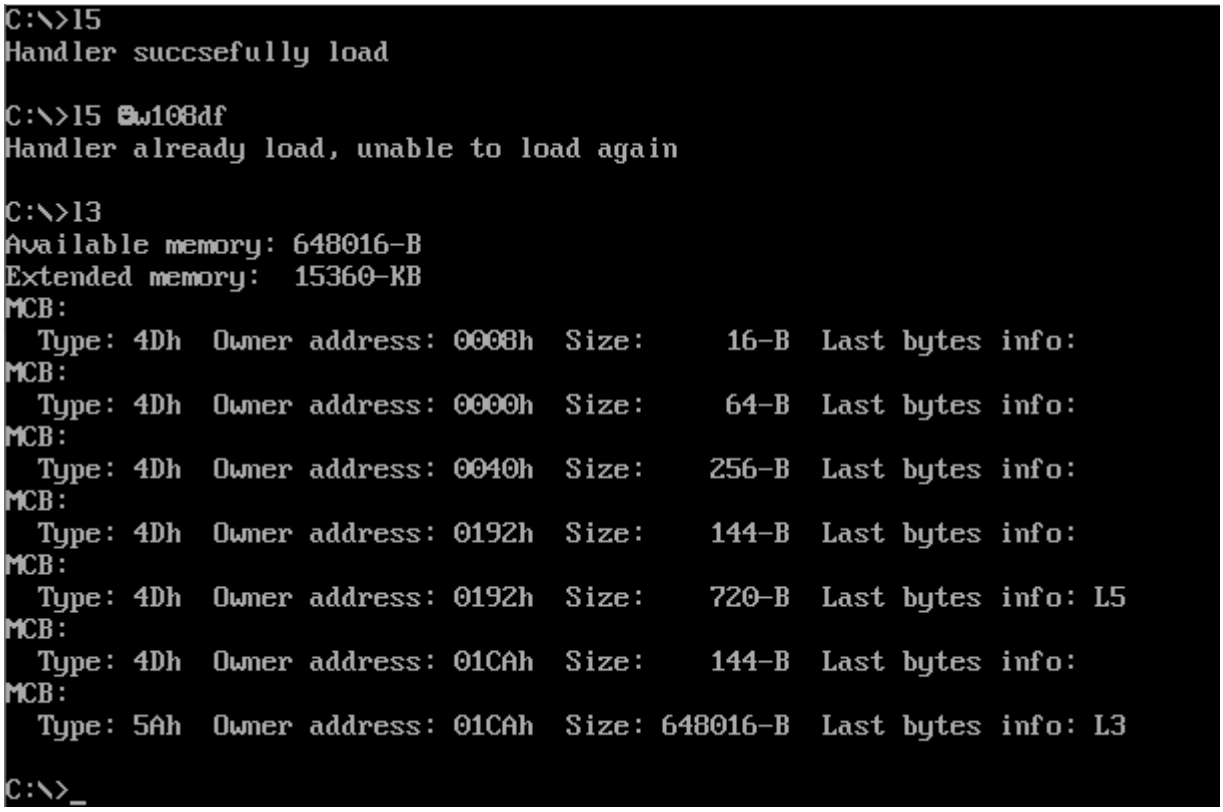
Была написана программа, исходный код которой приведет в приложении А, выполняющая следующие функции:

- 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 int 21h.
- Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход о функции 4Ch прерывания int 21h.
- Выгрузка прерываний по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Осуществляется выход по функции 4Ch прерывания int 21h.

Пользовательский обработчик принимает сигнал нажатия на клавиши клавиатуры. Полученный скан-код анализируется и передается стандартному обработчику, кроме случаев нажатия на кнопки: I, O, S, Q, т.е. пользовательским обработчиком обрабатываются следующие скан-коды соответственно: 17h, 18h, 1Fh, 10h. Результат загрузки обработчика прерывания в память, попытка повторной

загрузки, обработка прерыванием введенной строки: «QwioSdf», а также состояние памяти после этого представлены на рисунке 1.

Рисунок 1 — Загрузка обработчика прерывания в память и пример работы.



На рисунке 2 представлена выгрузка обработчика прерывания из памяти, попытка повторной выгрузки, а также состояние памяти после этого.

Рисунок 2 — Состояние памяти после выгрузки обработчика.

Контрольные вопросы

- Какого типа прерывания использовались в работе?

  В программе реализован пользовательский обработчик аппаратного прерывания. Также использовалось программное прерывание int 21h.

- Чем отличается скан-код от кода ASCII?

  Скан-код – уникальное число-идентификатор клавиши, используемое драйвером клавиатуры для распознавания нажатой клавиши. ASCII-код – это номер, закрепленный за символом, в таблице кодировок. Скан-код характеризуете клавишу, а код ACSII – символ.

Выводы

В ходе лабораторной работы была исследована возможность встраивания пользовательского обработчика прерываний в стандартное прерывание клавиатуры. Программа загружает или выгружает обработчик прерывания в память, в зависимости от параметра в командной строке.

ПРИЛОЖЕНИЕ А

Исходный код программы l5.asm

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

NewInt PROC FAR
	jmp IntCode

IntData:
	Signature dw 6666h ;сигнатура для проверки загружено ли наше прерывание
	KeepIP  dw 0
	KeepCS  dw 0
	KeepPSP dw 0
	KeepSS dw 0
	KeepSP dw 0
	KeepAX dw 0
	Keyi DB 17h
	Keyo db 18h
	Keys db 1Fh
	Keyq db 10h
	ExtraStack dw 100 dup(0)

IntCode:
	mov KeepSS, ss
	mov KeepSP, sp
	mov KeepAX, ax

	mov ax, seg ExtraStack
	mov ss, ax
	mov sp, offset IntCode

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

	in al, 60h
	cmp al, Keyi
		je KI
	cmp al, Keyo
		je KO
	cmp al, Keys
		je KS
	cmp al, Keyq
		je KQ

DefHandler:
	pushf
	call dword PTR cs:KeepIP
	jmp retern
KI:
	mov cl, '1'
	jmp MyHandler
KO:
	mov cl, '0'
	jmp MyHandler
KS:
	mov cl, '8'
```

```
	jmp MyHandler
KQ:
	mov cl, 02h
	jmp MyHandler

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

	mov ah, 05h
	mov CH, 00h
	int 16h

retern:
	pop es
	pop ds
	pop si
	pop dx
	pop cx
	pop bx

	mov sp, KeepSP
	mov ax, KeepSS
	mov ss, ax
	mov ax, KeepAX ;восстанавливаем регистры

	mov al, 20h ;разрешение прерываний более низкого уровня
	out 20h, al

	iret
NewInt ENDP
LastIntByte:


Residency PROC
	; Установка резидентности
	mov dx, offset LastIntByte; размер в байтах от начала сегмента
	mov cl, 4 ; перевод в параграфы
	shr dx, cl
	add dx, 16h
	inc dx ;размер в параграфах

	mov ah, 31h ;Резидентная программа активизируется каждый раз при возникновении
прерывания,
	int 21h ; вектор которого эта программа изменила на адрес одной из своих процедур.
Residency ENDP


IsTailUnLoad PROC
; провер, есть ли /un, результат в ax
	cmp byte ptr es:[82h], '/'
		jne   TailFalse
	cmp byte ptr es:[83h], 'u'
		jne TailFalse
	cmp byte ptr es:[84h], 'n'
		jne TailFalse
```

```
;проверка, нет ли каких-то еще букв после /un, ищем пробел или перевод строки
	cmp byte ptr es:[85h], 13
		je TailTrue
	cmp byte ptr es:[85h], ' '
		je TailTrue

TailFalse:
	mov ax, 0
	ret

TailTrue:
	mov ax, 1
	ret

IsTailUnLoad ENDP



IsIntLoad PROC
	push bx
	push si
	push es
	push dx
	mov si, offset Signature
	sub si, offset NewInt ;смещение до сигнатуры нашего обработчика

	mov ah, 35h
	mov al, 09h
	int 21h ;es:bx - адрес обработчика прерывания
	mov ax, es:[bx+si] ;сигнатура из установленного сейчас обработчика
	mov dx, Signature  ; сигнатура нашего обработчика
	cmp ax, dx ;если совпали, значит загружен наш обработчик
		je IntIsLoad

IntIsntLoad:
	mov ax, 0
		jmp reter

IntIsLoad:
	mov ax, 1

reter:
	;ax = 1 - сигнатуры совпали, иначе ax = 0
	pop dx
	pop es
	pop si
	pop bx

	ret
IsIntLoad ENDP



LoadInt PROC
	push ax
	push dx
	push bx
	push es

	mov ah, 35h ; получение вектора предыдущего прерывания
	mov al, 09h
```

```
	int 21h   ;сохранение адреса обработчика прерывания
	mov KeepIP, bx ; запоминаем смещение
	mov KeepCS, es ; и сегмент

	push ds
	mov dx, offset NewInt ;ds:dx - адрес нашего обработчика прерывания
	mov ax, seg NewInt
	mov ds, ax
	mov ah, 25h ; установка нашего прерывания
	mov al, 09h
	int 21h
	pop ds

	pop es
	pop bx
	pop dx
	pop ax
	ret
LoadInt ENDP



UnLoadInt PROC
	push ax
	push bx
	push dx
	push si
	push es
	push ds
	; Взятие смещения до сохраненных данных
	mov si, offset KeepIP
	sub si, offset NewInt ;смещение

	mov ah, 35h ;получение, сохранение адреса обработчика текущего прерывания
	mov al, 09h
	int 21h

	cli
	mov dx, es:[bx+si]  ;KeepIP
	mov ax, es:[bx+si+2]; KeepCS
	mov ds, ax
	mov ah, 25h
	mov al, 09h
	int 21h ; воостанавливаем исходный вектор прерывания
	sti

	pop ds
	mov ax, es:[bx+si+4] ;KeepPSP
	mov es, ax
	push es
	mov ax, es:[2Ch]
	mov es, ax
	mov ah, 49h ;освобождаем память, занятую обработчиком, в es сег.адрес осв. блока
памяти
	int 21h


	pop es
	mov ah, 49h
	int 21h

	pop es
```

```
	pop si
	pop dx
	pop bx
	pop ax
	ret
UnLoadInt ENDP


MAIN PROC
	mov ax, DATA
	mov ds, ax
	mov KeepPSP, es
	call IsTailUnLoad ; проверка есть ли /un
	cmp ax, 1
		je UnLoad

Load: ;ключа нет - или загрузить обработчик или вывести сообщение
	call IsIntLoad
	cmp ax, 1
		je CantLoad

	CanLoad:
		call LoadInt
		mov dx, offset InLoad
		mov ah, 09h
		int 21h
		call Residency

			jmp ext

	CantLoad:
		mov dx, offset HLoad
		mov ah, 09h
		int 21h
			jmp ext

UnLoad: ;ключ есть - или выгрузить обработчик или вывести сообщение
	call IsIntLoad
	cmp ax, 0
		jne CanUnLoad

	CantUnLoad:
		mov dx, offset HnLoad
		mov ah, 09h
		int 21h
			jmp ext

	CanUnLoad:
		call UnLoadInt
		mov dx, offset OutLoad
		mov ah, 09h
		int 21h
			jmp ext

ext:
	mov ah, 4Ch
	int 21h

	MAIN ENDP
CODE ENDS

;важно, что сегммент стека и данных после сегмента кода
```

```
;тк иначе обработчик займет очень много памяти
ASTACK SEGMENT  STACK
       dw 200 DUP(0)
ASTACK ENDS

DATA SEGMENT
       HLoad db 'Handler already load, unable to load again', 13, 10, '$'
       InLoad db 'Handler succsefully load', 13, 10, '$'
       OutLoad db 'Handler succsefully unload', 13, 10, '$'
       HnLoad db 'Handler not load yet, unable to unload', 13, 10, '$'
DATA ENDS



END MAIN
```