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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Операционные системы»
Тема: Обработка стандартных прерываний

Студент гр. 8383 _____ Шишкин И.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Построить обработчик прерываний сигнала таймера. Изучить способы загрузки резидентной программы в память и ее выгрузку

Ход работы.

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

1) Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
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 4Ch 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 4Ch прерывания int 21h.

Код пользовательского прерывания должен выполнять следующие функции:

1) Сохранять значения регистров в стеке при входе и восстановить их при выходе.
2) При выполнении тела процедуры накапливать общее суммарное число прерываний и выводить на экран. Для вывода на экран следует использовать прерывание int 10h, которое позволяет непосредственно выводить информацию на экран.

Программа после запуска выводит количество прерываний в строке 13h и колонке 13h, а также строку: если нет параметра /un и прерывание не было установлено, либо же прерывание просто не было установлено, то выводится “Interrupt not yet set”; если есть параметр /un и прерывание установлено, то выводится “Interrupt already set, but the /un parameter is found”; если нет параметра /un и прерывание установлено, то выводится “Interrupt already set”.

С помощью программы из ЛР3 было выведено состояние памяти: на рис. 1 – после загрузки прерывания, на рис. 2 – после повторной загрузки прерывания, на рис. 3 – после выгрузки прерывания.

Рисунок 1 – После загрузки прерывания

Рисунок 2 – После повторной загрузки прерывания

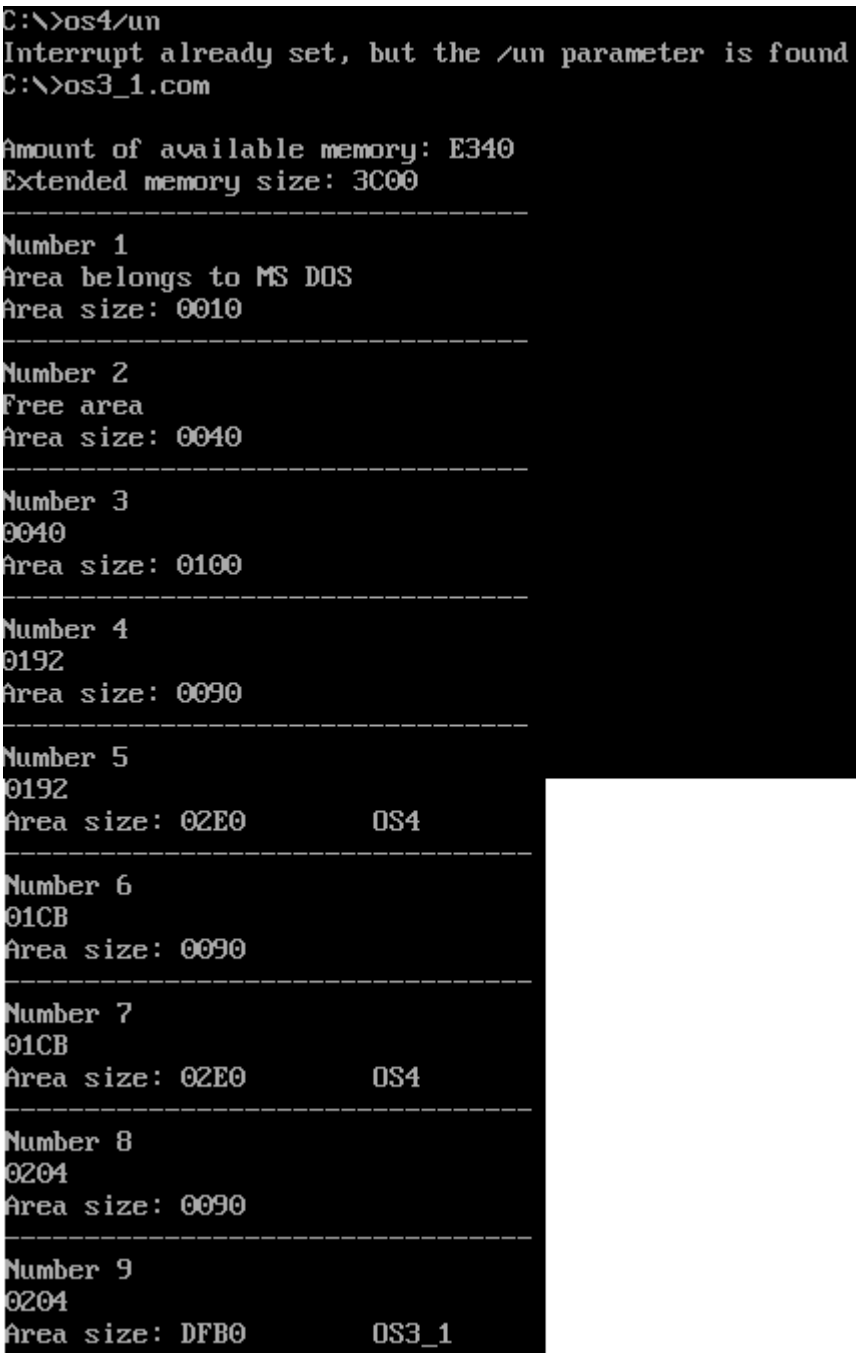

Рисунок 3 – После выгрузки прерывания

Контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Любой компьютер содержит системный таймер. Это устройство вырабатывает прерывание INT 8h приблизительно 18,2 раза в секунду. При инициализации BIOS устанавливает свой обработчик для прерывания таймера. Этот обработчик каждый раз увеличивает на 1 текущее значение четырехбайтовой переменной, располагающейся в области данных BIOS по

адресу 0000:046Ch – счетчик тиков таймера. Если этот счетчик переполняется (прошло более 24 часов с момента запуска таймера), в ячейку 0000:0470h заносится 1. Еще одно действие, которое выполняет обработчик прерывания таймера – вызов прерывания INT 1Ch. После инициализации системы вектор INT 1Ch указывает на команду IRET, т.е. ничего не выполняется. Программа может установить собственный обработчик этого прерывания для того чтобы выполнять какие-либо периодические действия.

Механизм обработки прерывания таймера:

1. Увеличение счетчика, проверка его на переполнение
2. Проверка на возможность обработки прерывания с соответствующим приоритетом
3. Вызов прерывания INT 1Ch
4. Сброс контроллера прерываний

2. Какого типа прерывания использовались в работе?

INT 10h – видео сервис.

Пользовательское прерывание INT 1Ch.

Выводы.

В ходе выполнения лабораторной работы была реализована программа, загружающая и выгружающая пользовательское прерывание от системного таймера в память.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

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

ROUT PROC FAR
	jmp INTERRUPT_BEGIN
	STR_FOR_INT db 'Number of interruptions: 0000$'
	INTERRUPT_ID dw 0804h
	KEEP_PSP DW 0
	KEEP_IP dw 0
	KEEP_CS dw 0
	KEEP_AX dw 0
	KEEP_SS dw 0
	KEEP_SP 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 BP
		push ES

		;getCurs
		mov AH, 03h
		mov BH, 00h
		int 10h
		push DX
		;setCurs
		mov ah,09h ; писать символ с текущей позиции курсора
		mov bh,0 ; номер видео страницы
```

```
        mov cx,0 ; число экземпляров символа для записи
        int 10h ; выполнить функцию

        mov ah,02h
        mov bh,0
        mov dh,13h ; DH,DL = строка, колонка (считая от 0)
        mov dl,13h
        int 10h ; выполнение.

        mov AX, SEG STR_FOR_INT
        push DS
        push BP
        mov DS, AX
        mov SI, offset STR_FOR_INT
        add SI, 24
        mov CX, 4

CYCLE:
        mov BP, CX
        mov AH, [SI+BP]
        inc AH
        mov [SI+BP], AH
        cmp AH, ':'
        jne END_OF_CYCLE
        mov AH, '0'
        mov [SI+BP], AH
        loop CYCLE

END_OF_CYCLE:
        pop BP
        pop DS

        push ES
        push BP
        mov AX, SEG STR_FOR_INT
        mov ES, AX
        mov BP, offset STR_FOR_INT
        call outputBP
        pop BP
        pop ES

        pop DX
        mov AH, 02h ; вернуть курсор
        mov BH, 0
```

```
                int 10h

                pop ES
                pop BP
                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
                ret
ROUT ENDP
LAST_BYTE:
;-----------------------------------------------
; функция вывода строки по адресу ES:BP на экран
outputBP proc
        mov ah,13h ; функция
        mov al,1 ; sub function code
        ; 1 = use attribute in BL; leave cursor at end of string
        mov bl, 1h
        mov cx, 29
        mov bh,0 ; видео страницы
        int 10h
        ret
outputBP endp
;-----------------------------------------------
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, 1CH ; 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, 1CH
        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, 1Ch
        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, 1Ch
        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_1CH PROC near
        push AX
        push BX
        push SI

        mov AH, 35h
        mov AL, 1Ch
        int 21h
        mov SI, offset INTERRUPT_ID
        sub SI, offset ROUT
        mov AX, ES:[BX+SI]
        cmp AX, 0804h
        jne END_OF_CHECK
        mov IS_INTERRUPT_LOADED, 1

        END_OF_CHECK:
                pop SI
                pop BX
                pop AX
                ret
CHECK_1CH ENDP
;-----------------------------------------
BEGIN PROC FAR
        push DS
        xor AX, AX
        push AX
        mov AX, DATA
        mov DS, AX
        mov KEEP_PSP, ES
```

```
        call CHECK_1CH
        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
```