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. 9382 _____ Савельев И.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определенные вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания,аппаратура компьютера передает управление по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить 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 настраиваетвектор прерываний, если прерывание не установлено, и осуществляется выход 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.

Для того, чтобы проверить установку прерывания, можно поступить следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код, который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом,находящимся в резиденте, можно определить, установлен ли резидент. Если значения

совпадают, то резидент установлен. Длину кода сигнатуры должна быть достаточной,чтобы сделать случайное совпадение маловероятным.

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

1)Сохраняет стек прерванной программы (регистры SS и SP) в рабочих переменных и восстановить при выходе.

2)Организовать свой стек.

3)Сохранить значения регистров в стеке при входе и восстановить их при выходе.

4)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.

5)Функция прерывания должна содержать только переменные, которые она использует.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания 1Ch установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛР 3, которая отображает карту памяти в виде списка блоков MCB. Полученные Результаты поместите в отчет.

Шаг 3. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

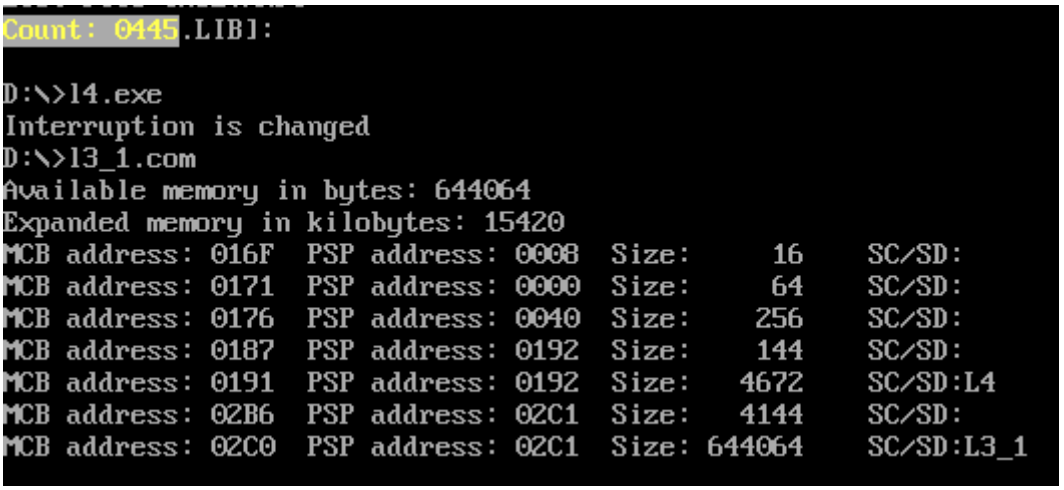
Шаг 4. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Шаг 5. Ответьте на контрольные вопросы.

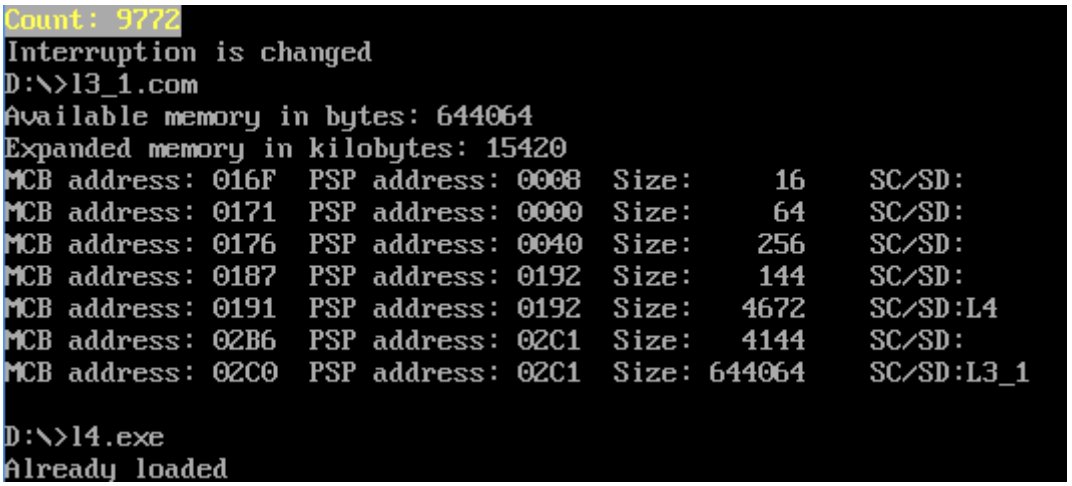
Выполнение работы.

В результате выполнения лабораторной работы была написана программа, которая устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний.

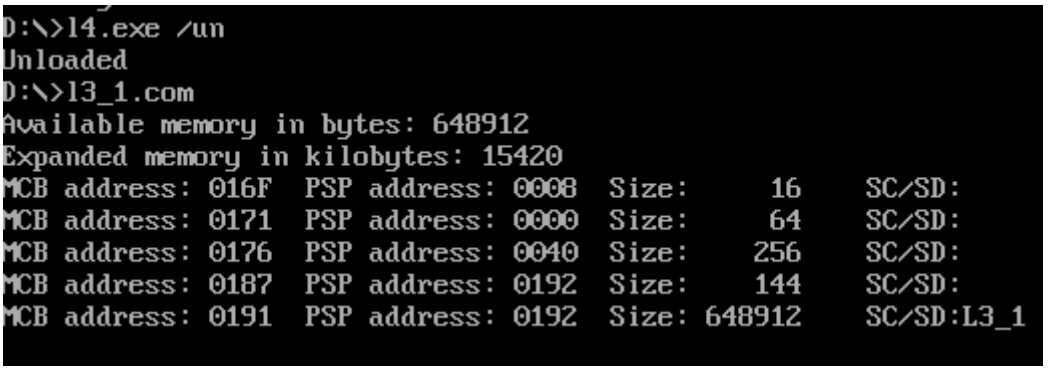
Результат запуска программ



Проверка, что программа определяет установленный обработчик прерываний.



Результат запуска программы с ключом /un



Вывод.

В процессе выполнения данной лабораторной работы были изучены основы работы со стандартными обработчиками прерываний.

Приложение А. Ответы на контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Приблизительно каждые 54 мс приходит сигнал прерывания, он принимается, сохраняются значения регистров, с помощью номера источника прерывания в таблице векторов устанавливается смещение, сохраняются адреса 2 байта в CS и 2 байта в IP, по сохраненному адресу выполняется прерывание, после чего восстанавливаются данные прерванного процесса и управление возвращается прерванной программе.

2. Какого типа прерывания использовались в работе?

Программные прерывания функций BIOS 10h и функций DOS 21h. Аппаратные прерывания 1Ch.

Приложение Б. Исходный код программы.

```
Stack SEGMENT  STACK
           DW 64 DUP(?)
Stack ENDS

CODE SEGMENT
       ASSUME CS:CODE, DS:DATA, SS:Stack

INTERRURTT proc far
    jmp START
    KEEP_PSP dw 0
    KEEP_IP dw 0
    KEEP_CS dw 0
    KEEP_SS DW 0
KEEP_SP DW 0
KEEP_AX DW 0

    INDEX dw 1388h
    COUNTER db 'Count: 0000'
    SecondStack DW 64 DUP(?)

START:
    mov KEEP_SP, sp
    mov KEEP_AX, ax
    mov ax, ss
    mov KEEP_SS, ss

    mov ax, KEEP_AX

    mov sp, offset START

    mov ax, seg SecondStack
    mov ss, ax

    push bx
       push cx
       push dx

; Считываем курсор
    mov ah,03h
mov bh,00h
int 10h
    push dx

    push si
    push cx
    push ds
    push ax
    push bp
```

```
    mov ax, SEG COUNTER
    mov ds,ax
    mov si, offset COUNTER

    add si, 6
    mov cx, 4

INCREASE:
    mov bp, cx
    mov ah, [si+bp]
    inc ah
    cmp ah, 3ah
    jl LEAVE_LOOP
    mov ah, 30h
    mov [si+bp], ah
    loop INCREASE

LEAVE_LOOP:
    mov [si+bp], ah

    pop bp
    pop ax
    pop ds
    pop cx
    pop si

    push es
    push bp

mov ax, SEG COUNTER
    mov es,ax
    mov ax, offset COUNTER

    mov bp,ax
    mov ah,13h
    mov al,00h
mov dh,9h
       mov dl,0h

    mov cx,11
    mov bh,0
    int 10h

    pop bp
    pop es

; Востанавливаем курсор
    pop dx
    mov ah,02h
    mov bh,0h
    int 10h
```

```
    pop dx
    pop cx
    pop bx

    mov KEEP_AX, ax
    mov sp, KEEP_SP
    mov ax, KEEP_SS
    mov ss, ax
    mov ax, KEEP_AX

    mov al, 20H
    out 20H, al

    iret

INTERRURTT endp

MYPRINT proc near
    push ax
    mov ah, 9h
    int 21h
    pop ax
    ret
MYPRINT endp

CHEK_LOADING proc near
       push ax
       push si

    push es
    push dx

; Получаем вектор и его номер
       mov ah,35h
       mov al,1ch
       int 21h

       mov si, offset INDEX
       sub si, offset INTERRURTT
       mov dx,es:[bx+si]
       cmp dx, INDEX
       jne end_CHEK_LOADING
       mov ch,1h

end_CHEK_LOADING:
    pop dx
    pop es
       pop si
       pop ax
       ret
```

```
CHEK_LOADING ENDP

CHECK_UN proc near
       push ax
push es
       mov al, es:[82h]
       cmp al, '/'
       jne LEAVEE
       mov al, es:[83h]
       cmp al, 'u'
       jne LEAVEE
       mov al, es:[84h]
       cmp al, 'n'
       jne LEAVEE
mov cl, 1h
LEAVEE:
    pop es
    pop ax
    ret
CHECK_UN endp

UNLOADD PROC near
       push ax
       push si

; Замена на старое прерывание
cli
       push ds
; Получаем вектор и его номер
       mov ah,35h
    mov al,1ch
    int 21h
    mov si,offset KEEP_IP
    sub si,offset INTERRURTT
    mov dx,es:[bx+si]
mov ax,es:[bx+si+2]
    mov ds,ax
    mov ah,25h
    mov al,1ch
    int 21h
    pop ds
    mov ax,es:[bx+si-2]
    mov es,ax
    push es
    mov ax,es:[2ch]
    mov es,ax

; освобождаем память среды
    mov ah,49h
    int 21h
```

```
; освобождаем память сегмента
    pop es
    mov ah,49h
    int 21h
    sti
    pop si
    pop ax
    ret
UNLOADD endp

LOADD PROC near
       push ax
       push dx
mov KEEP_PSP, es
; Получаем вектор и его номер
       mov ah,35h
    mov al,1ch
    int 21h
    mov KEEP_IP, bx
    mov KEEP_CS, es
       push ds
       lea dx, INTERRURTT
       mov ax, SEG INTERRURTT
       mov ds,ax
       mov ah,25h
       mov al,1ch
       int 21h
       pop ds
       lea dx, LASTT
       mov cl,4h
       shr dx,cl
       inc dx
       add dx,100h
xor ax, ax
       mov ah,31h
       int 21h
       pop dx
       pop ax
       ret
LOADD endp

MAIN proc far
    push  DS
    push  AX
    mov   AX,DATA
    mov   DS,AX
; Проверяем на un
    call CHECK_UN
    cmp cl, 1h
    je UNLOADDD
; Проверяем на загрузку
```

```
    call CHEK_LOADING
    cmp ch, 1h
    je ALREADYLOAD
; Если не загружен
    mov dx, offset LOADINGMSG
    call MYPRINT
    call LOADD
    jmp EXIT

UNLOADDD:
; Проверяем на загрузку
    call CHEK_LOADING
    cmp ch, 1h
    jne CANTUNLOAD
; Удаляем
    call UNLOADD
    mov dx, offset UNLOADEDMSG
    call MYPRINT
    jmp EXIT

; Обработчик не установлен
CANTUNLOAD:
    mov dx, offset NOTLOADEDMSG
    call MYPRINT
    jmp EXIT
; Уже установлен
ALREADYLOAD:
    mov dx, offset LOADEDMSG
    call MYPRINT
    jmp EXIT

EXIT:
    mov ah, 4ch
    int 21h
LASTT:
MAIN endp
CODE ends

DATA  SEGMENT
    LOADEDMSG db "Already loaded$"
    LOADINGMSG db "Interruption is changed$"
    NOTLOADEDMSG db "Not loaded$"
    UNLOADEDMSG db "Unloaded$"
DATA  ENDS

END Main
```