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

ОТЧЕТ

по практической работе № 4

по дисциплине «Операционные системы»

Тема: Обработка стандартных прерываний

Студентка гр. 9383 _____ Лапина А.А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2021

Цель работы.

Исследовать структуры обработчиков стандартных прерываний, построить обработчик прерываний сигналов таймера.

Постановка задачи.

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

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

- Сохранять значения регистров в стеке при входе и восстановить их при выходе.
- 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.

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_STR - 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.

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

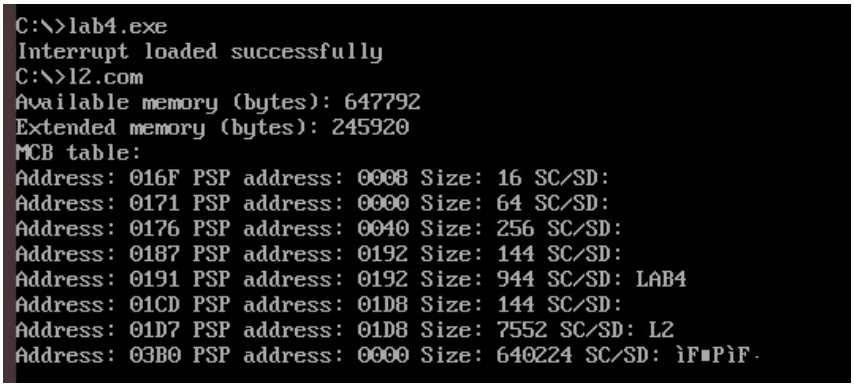

Рисунок 1 – Результат загрузки прерывания в память

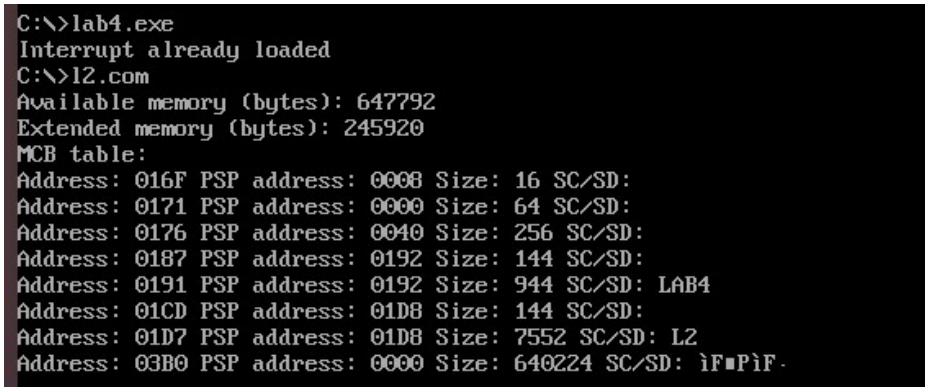

Рисунок 2 – Результат повторной загрузки прерывания в память

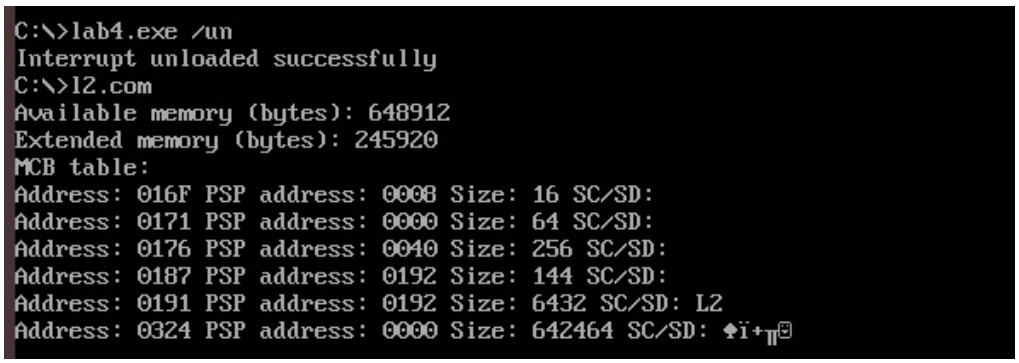

Рисунок 3 – Результат выгрузки выгрузки прерывания из памяти

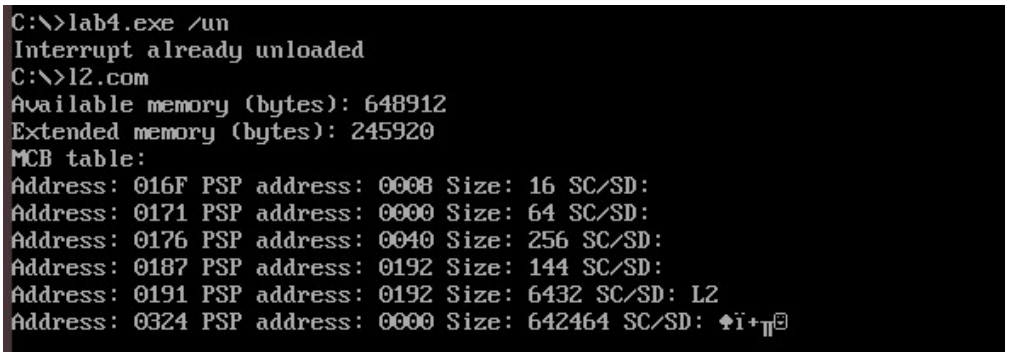

Рисунок 4 – Результат повторной выгрузки прерывания из памяти

Выводы.

В ходе лабораторной работы была исследована обработка стандартных прерываний, а также построен обработчик прерываний сигналов таймера, которые генерируются аппаратурой через определённые интервалы времени. Программа загружает и выгружает резидент, а также производится проверка флагов и загрузки прерывание в память.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как реализован механизм прерывания от часов?

Каждый такт из таймера вычитается определённое значение. Когда значение достигает 0, возникает прерывание от таймера. При возникновении прерывания процессор запоминает в стеке адрес возврата (CS:IP) и регистр флагов. Затем в CS:IP загружается адрес обработчика прерывания и выполняется его код. В конце регистры восстанавливаются, и процессор возвращается на выполнение прерванной программы.

2. Какие прерывания использовались в работе?

В данной работе использовались аппаратное прерывание 21h с вектором 1Ch, а также пользовательские прерывания 10h и 21h.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4.asm

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

data    segment
flag    db 0
ls      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     begint

ID      dw 0FFFFh
PSP     dw ?
keepCS  dw 0
keepIP  dw 0
keepSS  dw 0
keepSP  dw 0
keepAX  dw 0
intstr  db '00000 interrupts'
lenstr = $ - intstr
intstk  db 128 dup (?)
endstk:

begint: mov     keepSS, SS
        mov     keepSP, SP
        mov     keepAX, AX
        mov     AX, CS
        mov     SS, AX
        mov     SP, offset endstk
```

```
        push    BX
        push    CX
        push    DX
        push    DS
        push    ES
        push    SI
        push    DI
        push    BP
; ----------------------
        mov     AH, 03h     ;получение курсора
        mov     BH, 0
        int     10h
        push    DX
; ----------------------
        mov     AH, 02h     ;установка курсора
        mov     BH, 0
        mov     DX, 0
        int     10h
;----------------------
        push    BP
        push    DS
        push    SI
        mov     DX, seg intstr
        mov     DS, DX
        mov     SI, offset intstr
        mov     CX, 5
incr:   mov     BP, CX
        dec     BP
        mov     AL, byte ptr [SI+BP]
        inc     AL
        mov     [SI+BP], AL
        cmp     AL, 3Ah
        jne     good
        mov     AL, 30h
        mov     byte ptr [SI+BP], AL
        loop    incr
good:   pop     SI
        pop     DS


        push    ES
        mov     DX, seg intstr
        mov     ES, DX
```

```
        mov     BP, offset intstr
        mov     AH, 13h
        mov     AL, 1
        mov     BH, 0
        mov     CX, lenstr
        mov     DX, 0
        int     10h
        pop     ES
        pop     BP
; ----------------------
        mov     AH, 02h     ;возвращаем курсор
        mov     BH, 0
        pop     DX
        int     10h

        pop     BP
        pop     DI
        pop     SI
        pop     ES
        pop     DS
        pop     DX
        pop     CX
        pop     BX
        mov     AX, keepSS
        mov     SS, AX
        mov     SP, keepSP
        mov     AX, keepAX
        mov     AL, 20h
        out     20h, AL
        iret
endint:
inter   endp

load    proc
        push    AX
        push    CX
        push    DX
; ------------------------
        mov     AH, 35h   ;в программе при загрузке обработчика прерывания
        mov     AL, 1Ch
        int     21h
        mov     keepIP, BX
```

```
        mov     keepCS, ES
; ------------------------
        push    DS        ;Настройка прерывания
        mov     DX, offset inter
        mov     AX, seg inter
        mov     DS, AX
        mov     AH, 25h
        mov     AL, 1Ch
        int     21h
        pop     DS

        mov     DX, offset endint
        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

        cli             ;в программе при выгрузке обработчика прерывания
        push    DS
        mov     AH, 35h
        mov     AL, 1Ch
        int     21h
        mov     SI, offset keepCS
        sub     SI, offset inter
        mov     DX, ES:[BX+SI+2]
        mov     AX, ES:[BX+SI]
```

```
        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
        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, 1Ch
        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_STR   proc
        push    AX
        mov     AH, 09h
        int     21h
        pop     AX
        ret
PRINT_STR   endp

main    proc far
        mov     AX, data
        mov     DS, AX
        mov     PSP, ES
        mov     flag, 0
        call    isParam
        cmp     flag, 1
        je      un

        call    isLoad      ;Loading
        cmp     flag, 0
        je      notld
        mov     DX, offset ial
        call    PRINT_STR
        jmp     fin
notld:  mov     DX, offset ls
        call    PRINT_STR
```

```
        call    load
        jmp     fin

un:     call    isLoad      ;Unloading
        cmp     flag, 0
        jne     alrld
        mov     DX, offset iau
        call    PRINT_STR
        jmp     fin
alrld:  call    unload
        mov     DX, offset us
        call    PRINT_STR

fin:    mov     AX, 4C00h     ;завершение
        int     21h
main    endp
code    ends
        end     main
```