

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

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

Студент гр. 0382

Корсунов А.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Постановка задачи

Цель работы.

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

Сведения о функциях и структурах данных управляющей программы.

Процедура	Описание
setCurs	Установка курсора в нужную позицию
getCurs	Получение позиции курсора
INTERRUPT	Функция прерывания
CHECK_UN	Проверка наличия параметров командной строки
WRITE_MESSAGE_WORD	Вывод строки на экран
IS_LOADED	Проверка загрузки пользовательского прерывания
LOAD	Загрузка обработчика прерываний
UNLOAD	Выгрузка обработчика прерываний
MAIN	Главная функция программы

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 настраивает вектор прерываний, если прерывание не установлено, и

осуществляется 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.

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 МСВ. 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.

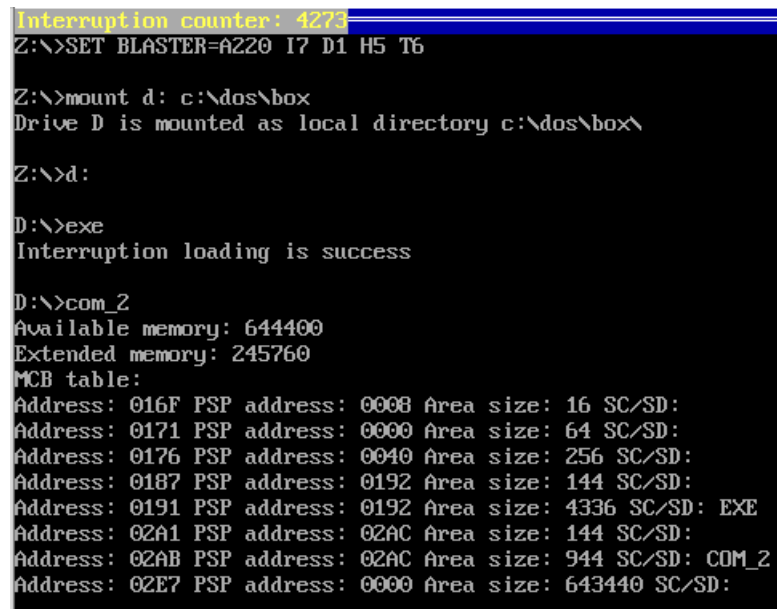
Шаг 4. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Шаг 5. Ответьте на контрольные вопросы.

Выполнение работы

Шаг 1. Для выполнения лабораторной работы был написан и отлажен программный модуль типа .EXE, который выполняет требуемые по заданию функции.

Шаг 2. Был запущена и отлажена программа.



```
Interruption counter: 4273
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount d: c:\dos\box
Drive D is mounted as local directory c:\dos\box\

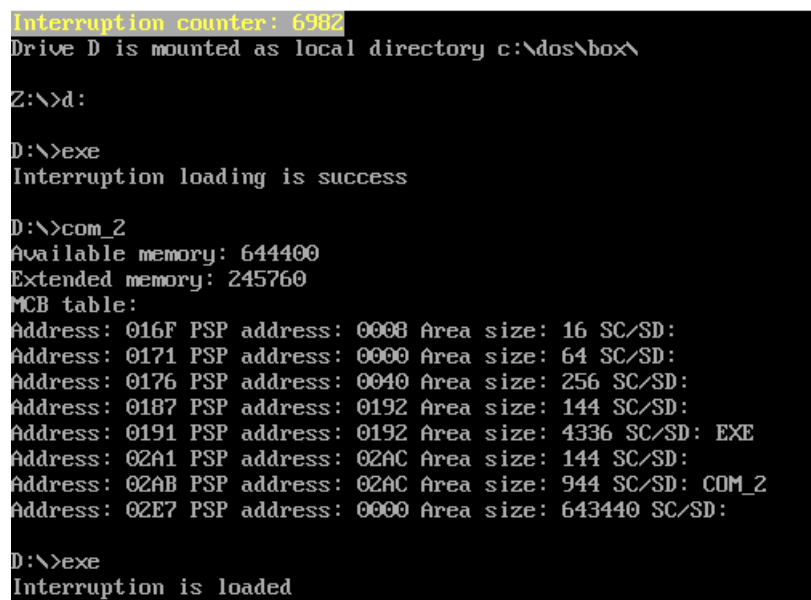
Z:\>d:

D:\>exe
Interruption loading is success

D:\>com_2
Available memory: 644400
Extended memory: 245760
MCB table:
Address: 016F PSP address: 0000 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD:
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 4336 SC/SD: EXE
Address: 02A1 PSP address: 02AC Area size: 144 SC/SD:
Address: 02AB PSP address: 02AC Area size: 944 SC/SD: COM_2
Address: 02E7 PSP address: 0000 Area size: 643440 SC/SD:
```

Рисунок 1 — иллюстрация работы программы (загрузка прерываний в память)

Шаг 3. Была запущена отлаженная программа еще раз.



```
Interruption counter: 6982
Drive D is mounted as local directory c:\dos\box\

Z:\>d:

D:\>exe
Interruption loading is success

D:\>com_2
Available memory: 644400
Extended memory: 245760
MCB table:
Address: 016F PSP address: 0000 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD:
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 4336 SC/SD: EXE
Address: 02A1 PSP address: 02AC Area size: 144 SC/SD:
Address: 02AB PSP address: 02AC Area size: 944 SC/SD: COM_2
Address: 02E7 PSP address: 0000 Area size: 643440 SC/SD:

D:\>exe
Interruption is loaded
```

Рисунок 2 - иллюстрация работы программы (определение установленного обработчика прерываний)

Шаг 4. Была запущена отлаженная программа с ключом выгрузки.

```
D:\>exe /un
Interruption is restored

D:\>com_2
Available memory: 648912
Extended memory: 245760
MCB table:
Address: 016F PSP address: 0008 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD:
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 944 SC/SD: COM_2
Address: 01CD PSP address: 0000 Area size: 647952 SC/SD: Г¶1Y^⬆
```

Рисунок 3 - иллюстрация работы программы (выгрузка резидентного обработчика, остановка сообщений прерываний)

Шаг 5. Ответьте на контрольные вопросы.

1) Как реализован механизм прерывания от часов?

Ответ. Прерывание от часов (int 1Ch) срабатывает примерно 18 раз в секунду, вызывается с помощью аппаратно генерируемого прерывания 08h – системного таймера. При вызове прерывания сохраняются регистры CS, IP для последующего возвращения в программу, а затем определяется адрес (смещение) вызываемого вектора прерывания в таблице прерываний, он помещается в регистры CS и IP. Затем программа сообщает системе, что прерывание от времени закончено, посылая сигнал конец-прерывания контроллеру прерываний, и восстанавливает регистры. После этого управление возвращается прерванной программе.

2) Какого типа прерывания использовались в работе?

Ответ. 1Ch – аппаратное прерывание, 10h и 21h – программные прерывания.

Вывод.

Было произведено исследование структуры обработчиков стандартных прерываний, был построен обработчик прерываний сигналов таймера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

файл exe.asm:

MYSTACK SEGMENT STACK

DW 200 DUP(?)

MYSTACK ENDS

DATA SEGMENT

LOADED db 'Interruption is loaded', 0DH, 0AH, '\$'

LOADED_YES db 'Interruption loading is success', 0DH, 0AH, '\$'

LOADED_NO db 'Interruption loading is not success', 0DH, 0AH, '\$'

LOADED_RESTORED db 'Interruption is restored', 0DH, 0AH, '\$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:MYSTACK

WRITE_MESSAGE_WORD PROC near

push AX

mov AH, 9

int 21h

pop AX

ret

WRITE_MESSAGE_WORD ENDP

setCurs PROC near

mov AH, 02h

```

    mov BH, 0h
    mov DH, 0h
    mov DL, 0h
    int 10h
    ret
setCurs ENDP

```

```

getCurs PROC near
    mov AH, 03h
    mov BH, 0
    int 10h
    ret
getCurs ENDP

```

```

INTERRUPT PROC far ;обработчик прерываний
    jmp begin
    counter db 'Interruption counter: 0000$'
    sign dw 7777h
    keep_IP dw 0 ; для хранения сегмента
    keep_CS dw 0 ; и смещения прерывания
    psp_address dw ?
    keep_SS dw 0
    keep_SP dw 0
    keep_AX dw 0
    my_stack dw 16 dup(?)

```

```

begin:
    mov keep_SP, SP
    mov keep_AX, AX
    mov keep_ss, SS

```


mov SP, offset begin
mov AX, seg my_stack
mov SS, AX

push AX ;сохранение изменяемых регистров
push CX
push DX

call getCurs ;получение курсора
push DX

call setCurs ;установка курсора
push SI
push CX
push DS
push BP

mov AX, seg counter
mov DS, AX
mov SI, offset counter
add SI, 21
mov CX, 4

loop_count:
mov BP, CX
mov AH, [SI+BP]
inc AH
mov [SI+BP], AH
cmp AH, 3ah

```
jne print  
mov AH, 30h  
mov [SI+BP], AH  
loop loop_count
```

print:

```
pop BP ; восстановление регистров  
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 CX, 26  
mov BH,0  
int 10h
```

```
pop BP  
pop ES  
pop DX
```

```
mov AH,02h ; возвращение курсора  
mov BH,0h  
int 10h
```

pop DX

pop CX

pop AX

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

INTERRUPT_last:

INTERRUPT ENDP

CHECK_UN PROC near

push AX

push BP

mov CL, 0h

mov BP, 81h

mov AL, ES:[BP + 1]

cmp AL, '/'

jne finish

mov AL, ES:[BP + 2]

cmp AL, 'u'

jne finish

mov AL, ES:[BP + 3]

cmp AL, 'n'

jne finish

mov CL, 1h

finish:

pop BP

pop AX

ret

CHECK_UN ENDP

IS_LOADED PROC near

push AX

push DX

push ES

push SI

mov CL, 0h

mov AH, 35h

mov AL, 1ch

int 21h

mov SI, offset sign

sub SI, offset INTERRUPT

mov DX, ES:[BX+SI]

cmp DX, sign

jne if_end

mov CL, 1h

```
if_end:
    pop SI
    pop ES
    pop DX
    pop AX
ret
IS_LOADED ENDP
```

LOAD PROC near

```
push AX
push CX
push DX
```

```
call IS_LOADED
cmp CL, 1h
je already_load
```

```
mov psp_address, ES
;загрузка обработчика прерывания
mov AH, 35h ; функция получения вектора
mov AL, 1ch ; номер вектора
int 21h
```

```
mov keep_CS, ES ; запоминание сегмента
mov keep_IP, BX ; и смещения
```

```
push ES
push BX
push DS
```

;настройка прерывания

lea DX, INTERRUPT ; смещение для процедуры в DX

mov AX, seg INTERRUPT ; сегмент процедуры

mov DS, AX ; помещаем в DS

mov AH, 25h ; функция установки вектора

mov AL, 1ch ; номер вектора

int 21h ;меняем прерывание

pop DS

pop BX

pop ES

mov DX, offset LOADED_YES

call WRITE_MESSAGE_WORD

lea DX, INTERRUPT_last

mov CL, 4h

shr DX, CL

inc DX

add DX, 100h

xor AX,AX

mov AH, 31h

int 21h

jmp end_load

already_load:

mov DX, offset LOADED

call WRITE_MESSAGE_WORD

end_load:

pop DX
pop CX
pop AX

ret

LOAD ENDP

UNLOAD PROC near

push AX
push SI

call IS_LOADED
cmp CL, 1h
jne not_load

;при выгрузке обработчика прерывания
cli

push DS
push ES
mov AH, 35h
mov AL, 1ch
int 21h ; восстанавливаем вектор

mov SI, offset keep_IP
sub SI, offset INTERRUPT
mov DX, ES:[BX+SI]
mov AX, ES:[BX+SI+2]
mov DS, AX
mov AH, 25h

mov AL, 1ch

int 21h

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

pop ES

pop DS

sti

mov DX, offset LOADED_RESTORED

call WRITE_MESSAGE_WORD

jmp end_unload

not_load:

mov DX, offset LOADED_NO

call WRITE_MESSAGE_WORD

end_unload:

pop SI


```

        pop AX
ret
UNLOAD ENDP

MAIN PROC far
        sub AX, AX
        mov AX, DATA
        mov DS, AX

        call CHECK_UN ;проверка на /un
        cmp CL, 0h
        jne un
        call LOAD ;загрузить
        jmp end_main

un:
        call UNLOAD ;выгрузить

end_main:
        xor AL, AL
        mov AH, 4ch
        int 21h
MAIN ENDP

CODE ENDS

END MAIN

```