

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»

Студентка гр. 6383

Михеева Е. Е.

Преподаватель

Губкин А. Ф.

Санкт-Петербург

2018

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт управление и выполняет соответствующие действия.

В лабораторной работе номер 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Алгоритм работы программы.

1. Проверяется, установлено ли пользовательское прерывание с вектором 1Ch. Проверка происходит путём сравнения значения сигнатуры, расположенной на определённом известном смещении в теле резидента, с реальным кодом, находящимся в резиденте.
2. Если пользовательское прерывание не установлено (код не совпадает с сигнатурой), то устанавливается резидентная функция для обработки прерывания, настраивается вектор прерываний и осуществляется выход по функции 4Ch прерывания int 21h.
3. Если пользовательское прерывание установлено (код совпадает с сигнатурой), то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
4. Если пользовательское прерывание установлено и программа запущена с ключом /up, то прерывание выгружается из памяти: происходит восстановление стандартного вектора прерываний и освобождение памяти,

занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Сведения о функциях и структурах данных управляющей программы.

| Название процедуры | Описание |
|--------------------|--|
| PRINT | Выводит на экран строку |
| ROUT | Пользовательский обработчик прерывания |
| CHECK_INT | Проверяет, установлено ли пользовательское прерывание |
| SET_INT | Устанавливает пользовательское прерывание в поле векторов прерываний |
| outputAL | Вывод символа с кодом AL в текущее положение курсора |
| DELETE_INT | Выход из пользовательского прерывания |
| EXIT_FROM_INTER | Проверка наличия /up |

Таблица 1. Описание функций.

| Название переменной | Тип | Описание |
|---------------------|-----|---|
| INT_IS_LOADED | db | Строка-сообщение для вывода информации о том, что пользовательское прерывание уже установлено |
| UNLOADED | db | Строка-сообщение для вывода информации о том, что пользовательское прерывание выгружено |
| INT_LOAD | db | Строка-сообщение для вывода информации о том, что пользовательское прерывание установлено |
| STRN | db | Строка, которая используется для определения: было ли установлено пользовательское прерывание или нет |

| | | |
|----------|----|--|
| KEEP_CS | dw | Переменная для сохранения значения в регистре CS |
| KEEP_IP | dw | Переменная для сохранения значения в регистре IP |
| KEEP_PSP | dw | Переменная для сохранения сегментного адреса PSP |

Таблица 2. Описание переменных.

Ход работы.

Шаг 1.

```

C:\>4
INTERRUPTION JUST LOADED

C:\>31
Size of available memory: 648016 B
Size of expanded memory: 15360 KB
ADDR TYPE      SIZE OWNER NAME
016F  4D      00016 0008
0171  4D      00064 0000 DPMILOAD
0176  4D      00256 0040
0187  4D      00144 0192
0191  4D      00720 0192 4
01BF  4D      00144 01CA
01C9  5A     648016 01CA 31
C:\>

```

Рис. 1. Результат работы программы 4.exe.

Шаг 2.

```

C:\>41
INTERRUPTION JUST LOADED

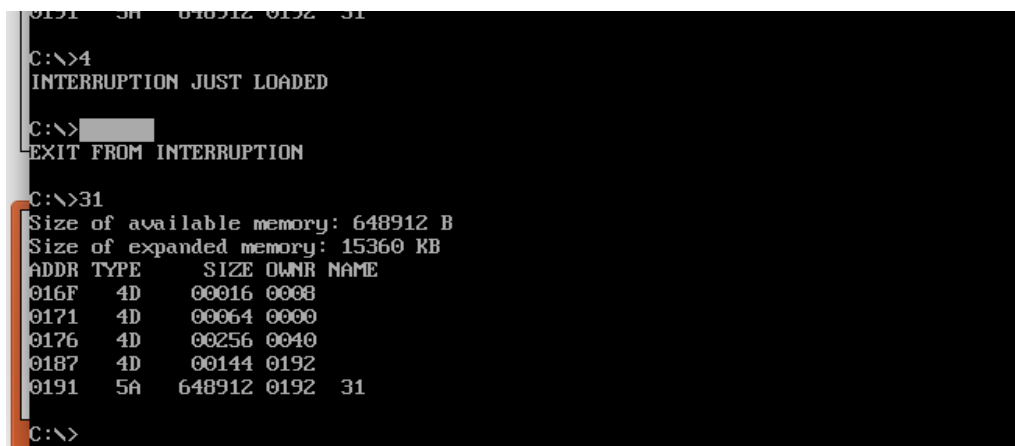
C:\>41
INTERRUPTION IS ALREADY LOADED ???

C:\>

```

Рис. 2. Проверка повторной загрузки прерывания.

Шаг 3.



```
C:\>4
INTERRUPTION JUST LOADED

C:\>
EXIT FROM INTERRUPTION

C:\>31
Size of available memory: 648912 B
Size of expanded memory: 15360 KB
ADDR TYPE      SIZE OWNER NAME
016F  4D      00016 0000
0171  4D      00064 0000
0176  4D      00256 0040
0187  4D      00144 0192
0191  5A     648912 0192  31

C:\>
```

Рис. 3. Проверка выгрузки резидентного обработчика прерывания.

Ответы на контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Ответ: Аппаратное прерывание 1Ch системного таймера вызывается автоматически при каждом тике системного таймера.

2. Какого типа прерывания использовались в работе?

Ответ: В данной лабораторной работе использовались: аппаратные прерывания (int 1Ch) и программные прерывания (int 21h, int 10h).

Вывод.

В ходе данной лабораторной работы была создан обработчик прерываний сигналов таймера, который проверяет, установлено ли пользовательское прерывание с вектором 1Ch, устанавливает резидентную функцию для обработки прерывания, выгружает пользовательское прерывание по соответствующему значению параметра командной строки /un.

Приложение А. Код программы

.286

ASTACK SEGMENT stack

db 100h dup(?)

ASTACK ENDS

DATA SEGMENT

INT_LOAD DB 'INTERRUPTION JUST LOADED', 0AH, 0DH, '\$'

INT_IS_LOADED DB 'INTERRUPTION IS ALREADY
LOADED !!!', 0AH, 0DH, '\$'

EXIT_FROM_INTERR DB 'EXIT FROM
INTERRUPTION', 0AH, 0DH, '\$'

INT_NOT_LOADED DB 'INTERRUPTION NOT
LOADED!!!', 0AH, 0DH, '\$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:CODE, ES:CODE, SS:ASTACK

PRINT PROC NEAR

pusha

push ds

mov ax, DATA

mov ds, ax

mov ah, 09h

int 21h

pop ds

popa

ret

PRINT ENDP

outputAL PROC NEAR

mov ah, 09h

mov cx, 1

mov bh, 0

int 10h

ret

outputAL ENDP

ROUT PROC FAR

jmp entry

STRN DB 'ABC'

entry:

pusha

push es

push ds

;getCurs

mov ah, 03h

mov bh, 0

int 10h

push dx

push cx

call outputAL

; setCurs

pop cx

pop dx

mov ah, 02h

int 10h

inc cs:COUNTER

; end of code

pop ds

pop es

popa

mov al, 20h

out 20h, al

iret

KEEP_CS DW 0

KEEP_IP DW 0

KEEP_PSP DW 0h

COUNTER DW 0h

last_byte:

ROUT ENDP

SET_INT PROC NEAR

pusha

push ds

push es

mov ah, 35h

mov al, 1Ch

int 21h

mov cs:KEEP_IP, bx

mov cs:KEEP_CS, es

mov ax, SEG ROUT

mov ds, ax

mov dx, offset ROUT

mov ah, 25h

mov al, 1Ch

int 21h

;end

mov dx, offset last_byte

shr dx, 4

inc dx

add dx, CODE

sub dx, cs:KEEP_PSP

mov ah, 31h

int 21h

pop es

pop ds

popa

ret

SET_INT ENDP

CHECK_INT PROC NEAR ; if set then al=1, else al=0

cli

pusha

push ds

push es

mov ah, 35h

mov al, 1Ch

int 21h

mov dx, es:KEEP_IP

mov ax, es:KEEP_CS

cmp word ptr es:STRN+0, 'BA'

jne no

cmp byte ptr es:STRN+2, 'C'

jne no

yes:

pop es

pop ds

popa

mov al, 1

sti

jmp end_ch

no:

pop es

pop ds

popa

mov al, 0

sti

end_ch:

ret

CHECK_INT ENDP

DELETE_INT PROC NEAR

cli

pusha

push ds

push es

mov ah, 35h

mov al, 1Ch

int 21h

push es

mov dx, es:KEEP_IP

mov ax, es:KEEP_CS

mov ds, ax

mov ah, 25h

mov al, 1Ch

int 21h

pop es

mov es, es:KEEP_PSP

push es

mov es, es:[2Ch]

mov ah, 49h

int 21h

pop es

mov ah, 49h

int 21h

pop es

pop ds

popa

sti

ret

DELETE_INT ENDP

EXIT_FROM_INTER PROC NEAR

pusha

cmp byte ptr ES:[82H], '/'

jne exit_from_int

cmp byte ptr es:[83H], 'u'

jne exit_from_int

cmp byte ptr es:[84H], 'n'

jne exit_from_int

popa

mov al, 1

jmp end_ex

exit_from_int:

popa

mov al, 0

end_ex:

ret

EXIT_FROM_INTER ENDP

BEGIN:

push DS

sub AX, AX

push AX

```
mov cs:KEEP_PSP, ds
```

```
;checking /un to exit or not
```

```
call EXIT_FROM_INTER
```

```
cmp al, 1
```

```
je end_int
```

```
call CHECK_INT
```

```
cmp ax, 0
```

```
jne loading
```

```
mov dx, offset INT_LOAD
```

```
call PRINT
```

```
call SET_INT
```

```
jmp exit
```

```
;already loaded
```

```
loading:
```

```
mov dx, offset INT_IS_LOADED
```

```
call PRINT
```

```
jmp exit
```

```
call CHECK_INT
```

```
cmp al, 1
```

```
jne already_loaded
```

```
;exit from int
```

```
end_int:
```

```
call DELETE_INT
```

```
mov dx, offset EXIT_FROM_INTERR
```

```
call PRINT
```

```
jmp exit
```

```
already_loaded:
```

```
mov dx, offset INT_LOAD
```

call PRINT

exit:

xor AL,AL

mov AH,4Ch

int 21H

CODE ENDS

END BEGIN