

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

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Организация ЭВМ и систем»
Тема: «Обработка стандартных прерываний»

Студент гр. 8381

Преподаватель

Переверзев Д.Е.

Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Необходимо 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.

Основные теоретические положения.

Резидентные обработчики прерываний - это программные модули, которые вызываются при возникновении прерываний определенного типа (сигнал таймера, нажатие клавиши и т.д.), которым соответствуют определенные вектора прерывания. Когда вызывается прерывание, процессор переключается на выполнение кода обработчика, а затем возвращается на выполнение прерванной программы. Адрес возврата в прерванную программу (CS:IP) запоминается в стеке вместе с регистром флагов. Затем в CS:IP загружается адрес точки входа программы обработки прерывания и начинается выполняться его код. Обработчик прерывания должен заканчиваться инструкцией IRET (возврат из прерывания).

Вектор прерывания имеет длину 4 байта. В первом хранится значение IP, во втором - CS. Младшие 1024 байта памяти содержат 256 векторов. Вектор для прерывания 0 начинается с ячейки 0000:0000, для прерывания 1 - с ячейки 0000:0004 и т.д.

Обработчик прерывания - это отдельная процедура, имеющая следующую структуру:

ROUT PROC FAR

PUSH AX ; сохранение изменяемых регистров

.....

<действия по обработке прерывания>

```
POP AX ; восстановление регистров MOV AL, 20H
OUT 20H,AL
IRET
ROUT ENDP
```

Две последние строки необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное. Для установки написанного прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес.

```
PUSH DS
MOV DX, OFFSET ROUT ; смещение для процедуры в DX
MOV AX, SEG ROUT ; 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
```

Программа, выгружающая обработчик прерываний должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H позволяет восстановить значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. Программа должна содержать следующие инструкции:

```
; -- хранится в обработчике прерываний
KEEP_CS DW 0 ; для хранения сегмента
KEEP_IP DW 0 ; и смещения прерывания
```

; -- в программе при загрузке обработчика прерывания

MOV AH, 35H ; функция получения вектора ; номер

MOV AL, 1CH вектора ШТЕ 21P

MOV KEEP_IP, BX ; запоминание смещения

MOV KEEP_CS, ES ; и сегмента

; -- в программе при выгрузке обработчика прерываний CLI

PUSH DS

MOV DX, KEEP_IP

MOV AX, KEEP_CS

MOV DS, AX

MOV AH, 25H

MOV AL, 1CH

INT 21H ; восстанавливаем вектор

POP DS

STI

Для того, чтобы оставить процедуру прерывания резидентной в памяти, следует воспользоваться функцией DOS 31h прерывания 21h. Эта функция оставляет память, размер которой указывается в качестве параметра, занятой, а остальную память освобождает и осуществляет выход в DOS.

Функция 31h int 21h использует следующие параметры:

AH - номер функции 31h;

AL - код завершения программы;

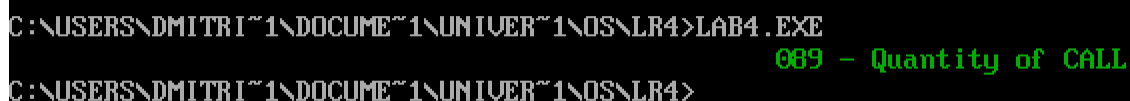
DX - размер памяти в параграфах, требуемый резидентной программой. Пример

обращения к функции:

```
MOV DX, OFFSET LAST_BYTE ; размер в байтах от начала сегмента
MOV CL,4 ; перевод в параграфы
SHR DX,CL
INC DX ; размер в параграфах
MOV AH,31h
INT 21h
```

Выполнение работы.

Программа проверяет, установлено ли пользовательское прерывание с вектором 1Ch. Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход через функцию 2Ch прерывания 21h. Выгружает прерывание по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождения памяти, занимаемой резидентом. Осуществляется выход через функцию 4Ch прерывания 21h. Результат работы программы представлен на рис. 1.



```
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB4.EXE
009 - Quantity of CALL
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>
```

Рисунок 1 – результат работы программы LAB4.EXE.

Для проверки размещения прерывания в памяти была запущена программа из лабораторной работы 3.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB4.EXE
                                052 - Quantity of CALL
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB3.COM
Amount of available memory: 643952 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:    4784 B. Information in last bytes: LAB4
New MCB:
Type: 4Dh. Sector: 02C8h. Size:    4144 B. Information in last bytes:
New MCB:
Type: 5Ah. Sector: 02C8h. Size: 643952 B. Information in last bytes: LAB3
                                101 - Quantity of CALL
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>

```

Рисунок 2 – состояние памяти после загрузки собственного прерывания.

После повторного запуска программы было выведено сообщение о том, что резидентная программа уже загружена. Результат повторного запуска работы представлен на рис. 3.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB4.EXE
Intrruption already loaded
                                209 - Quantity of CALL
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>

```

Рисунок 3 – повторный запуск программы lab4.exe.

Была запущена программа с ключом выгрузки. Для того чтобы проверить, что память, занятая резидентом, освобождена, был выполнен запуск программы лабораторной работы 3.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>
                                213 - Quantity of CALL
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB4.EXE /un
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB4.EXE /un
Interruption was not loadd
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>

```

Рисунок 4 – Результат запуска программы с ключом выгрузки.

```

C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>LAB3.COM
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 5Ah. Sector: 0192h. Size: 648912 B. Information in last bytes: LAB3
C:\USERS\DMITRI~1\DOCUME~1\UNIVER~1\OS\LR4>

```

Рисунок 5 – Состояние памяти после выгрузки резидентной программы.

Контрольные вопросы

1. Как реализован механизм прерывания от часов?

Прерывание по таймеру вызывается каждые 55 мс – 18 раз в секунду. После вызова сохраняется содержимое регистров и определяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерываний. Полученный адрес сохраняется в регистр CS:IP. После этого управление передаётся по этому адресу, т. е. выполняется запуск обработчика прерываний и происходит его выполнение. После выполнения происходит возврат управления прерванной программе.

2. Какого типа прерывания использовались в работе?

- int 1Ch -пользовательское прерывание по таймеру. Аппаратное прерывание
- int 10h - видео сервис. Программное прерывание
- int 21h - сервис DOS. Программное прерывание

Вывод

В ходе выполнения лабораторной работы были построен и изкчен обработчик прерываний сигналов таймера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. lab4.asm

SSTACK SEGMENT STACK

DW 128

SSTACK ENDS

DATA SEGMENT

LOADED db 'Intrruption already loaded', 0DH, 0AH, '\$'

NOTLOAD db 'Interruption was not loadd', 0DH, 0AH, '\$'

LOAD db 0

UN db 0

DATA ENDS

CODE SEGMENT

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

ROUT PROC FAR

jmp INTERRUPT

COUNTER db '000 - Quantity of CALL '

SIGN dw 1000h

KEEP_IP dw 0

KEEP_CS dw 0


```
KEEP_PSP dw 0
KEEP_SS dw 0
KEEP_SP dw 0
KEEP_AX dw 0
INT_STACK dw 128 dup(0)
```

INTERRUPT:

```
mov KEEP_AX, AX
mov KEEP_SP, SP
mov KEEP_SS, SS
mov AX, SEG INT_STACK
mov SS, AX
mov AX, offset INT_STACK
add AX, 256
mov SP, AX
push AX
push BX
push CX
push DX
push SI
push ES
push DS
mov AX, seg COUNTER
mov DS, AX
mov AH, 03h
```

```
mov bh, 0h
int 10h
push DX
mov AH, 02h
mov DX, 1730h
mov bh, 0h
int 10h
mov AX, seg COUNTER
push DS
mov DS, AX
mov SI, offset COUNTER
add SI, 2
mov CX, 3
```

loopa:

```
mov AH, [SI]
inc AH
mov [SI], AH
cmp AH, ':'
jnz END_loopa
mov AH, '0'
mov [SI], AH
dec SI
loop loopa
```

END_loopa:

```
pop DS
push ES
push BP
mov AX, seg COUNTER
mov ES, AX
mov BP, offset COUNTER
mov AH, 13h
mov AL, 01h
mov bh, 00h
mov BL, 02h
mov CX, 25
int 10h
pop BP
pop ES
pop DX
mov AH, 02h
mov bh, 00h
int 10h
pop DS
pop ES
pop SI
pop DX
pop CX
pop BX
mov AX, KEEP_SS
```

```
mov SS, AX
mov AX, KEEP_AX
mov SP, KEEP_SP
mov AL, 20h
out 20h, AL
IRET
```

ROUT ENDP

END_ROUT:

CHECK PROC

```
push AX
push BX
push SI
mov AH, 35h
mov AL, 1ch
int 21h
mov SI, offset SIGN
sub SI, offset ROUT
mov AX, ES:[BX+SI]
cmp AX, SIGN
jnz check_end
mov LOAD, 1
```

check_end:

```
pop SI
```

pop BX

pop AX

ret

CHECK ENDP

CHECK_UN PROC

push AX

push ES

mov AX, KEEP_PSP

mov ES, AX

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

jnz CHECK_UN_END

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

jnz CHECK_UN_END

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

jnz CHECK_UN_END

mov UN, 1

CHECK_UN_END:

pop ES

pop AX

ret

CHECK_UN ENDP

LOAD_I PROC

push AX

push BX

push CX

push DX

push ES

push DS

mov AH, 35h

mov AL, 1ch

int 21h

mov KEEP_IP, BX

mov KEEP_CS, ES

mov AX, seg ROUT

mov DX, offset ROUT

mov DS, AX

mov AH, 25h

mov AL, 1ch

int 21h

pop DS

mov DX, offset END_ROUT

mov cl, 4h

shr DX, cl

add DX, 10fh

inc DX

```
xor AX, AX
mov AH, 31h
int 21h
pop ES
pop DX
pop CX
pop BX
pop AX
ret
```

```
LOAD_I ENDP
```

```
LOAD_UN PROC
```

```
CLI
push AX
push BX
push CX
push DX
push DS
push ES
push SI
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 CX
```


pop BX

pop AX

ret

LOAD_UN ENDP

MAIN PROC FAR

push DS

xor AX,AX

push AX

mov AX, DATA

mov DS, AX

mov KEEP_PSP, ES

cALI CHECK

cALI CHECK_UN

cmp UN, 1

je UNLOAD

cmp LOAD, 1

jnz LOAD_

mov DX, offset LOADED

push AX

mov AH, 09h

int 21h

pop AX

jmp MEND

LOAD_:

 cALI LOAD_I

 jmp MEND

UNLOAD:

 cmp LOAD, 1

 jnz NOT_LOAD

 cALI LOAD_UN

 jmp MEND

NOT_LOAD:

 mov DX, offset NOTLOAD

 push AX

 mov AH, 09h

 int 21h

 pop AX

MEND:

 xor AL, AL

 mov AH, 4ch

 int 21h

MAIN ENDP

CODE ENDS

END MAIN