**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. 6383 |  | Медведев Г.О. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

В лабораторной работе № 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определенные интервалы времени и, при возникновении такого сигнала, возникает прерывание с определенным значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

**Описание функций и структур данных.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| PRINT | вызывает функцию печати строки |
| setCurs | устанавливает позицию курсора |
| getCurs | возвращает положение курсора |
| ROUT | пользовательский обработчик прерываний, считающий и печатающий количество его вызовов |
| CHECKING | проверяет загруженность обработчика прерываний |
| SET\_INTERRUPT | устанавливает новый обработчик прерывания, запоминая данные для восстановления предыдущего обработчика прерываний |

**Последовательность действий, выполняемых утилитой.**

1. Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
2. Устанавливает обработчик прерываний, если он не установлен, и осуществляется выход.
3. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход.
4. Выгрузка прерывания по соответствующему значению параметра в командной строке /un, восстановления стандартного вектора прерывания.

**Результаты выполнения программ.**

1. Состояние памяти до запуска программы 4.exe представлено на Рис.1, в качестве программы изображена работа файла 3\_1.com.

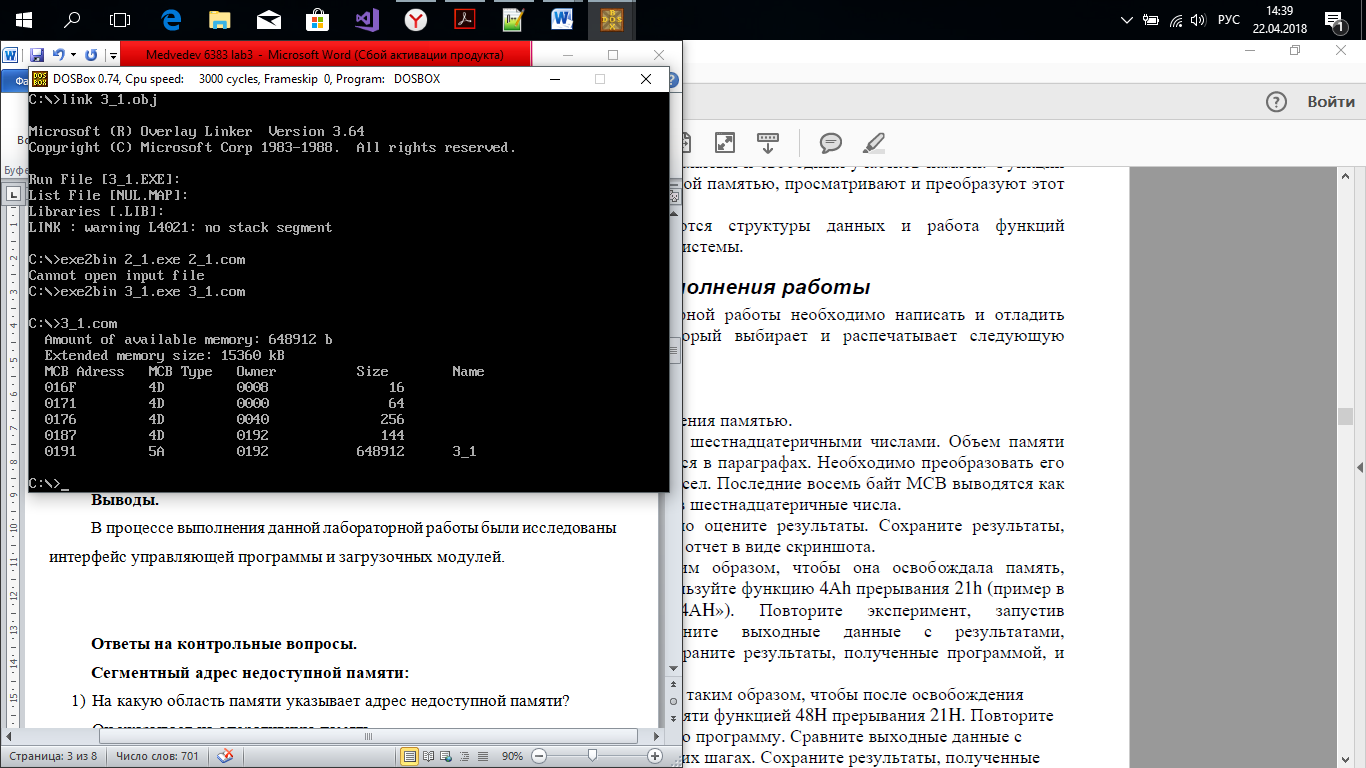


Рисунок 1 – Результат выполнения программы 3\_1.com

1. На Рис. 2 представлен запуск программы 4.exe

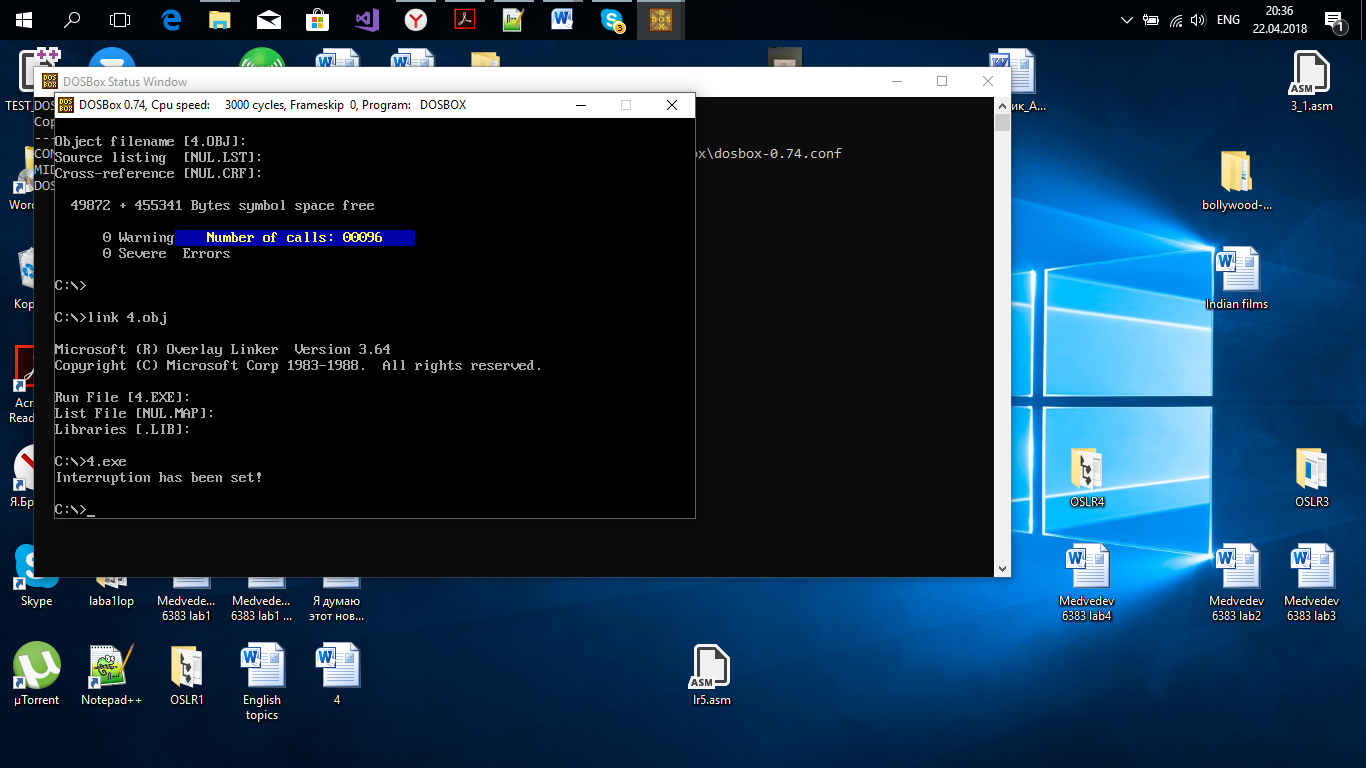


Рисунок 2 – Результат выполнения программы 4.exe

1. Проверим размещение прерывания в памяти с помощью программы 3\_1.com, которая отображает карту памяти в виде списка блоков MCB. Выполнение показано на Рис. 3.

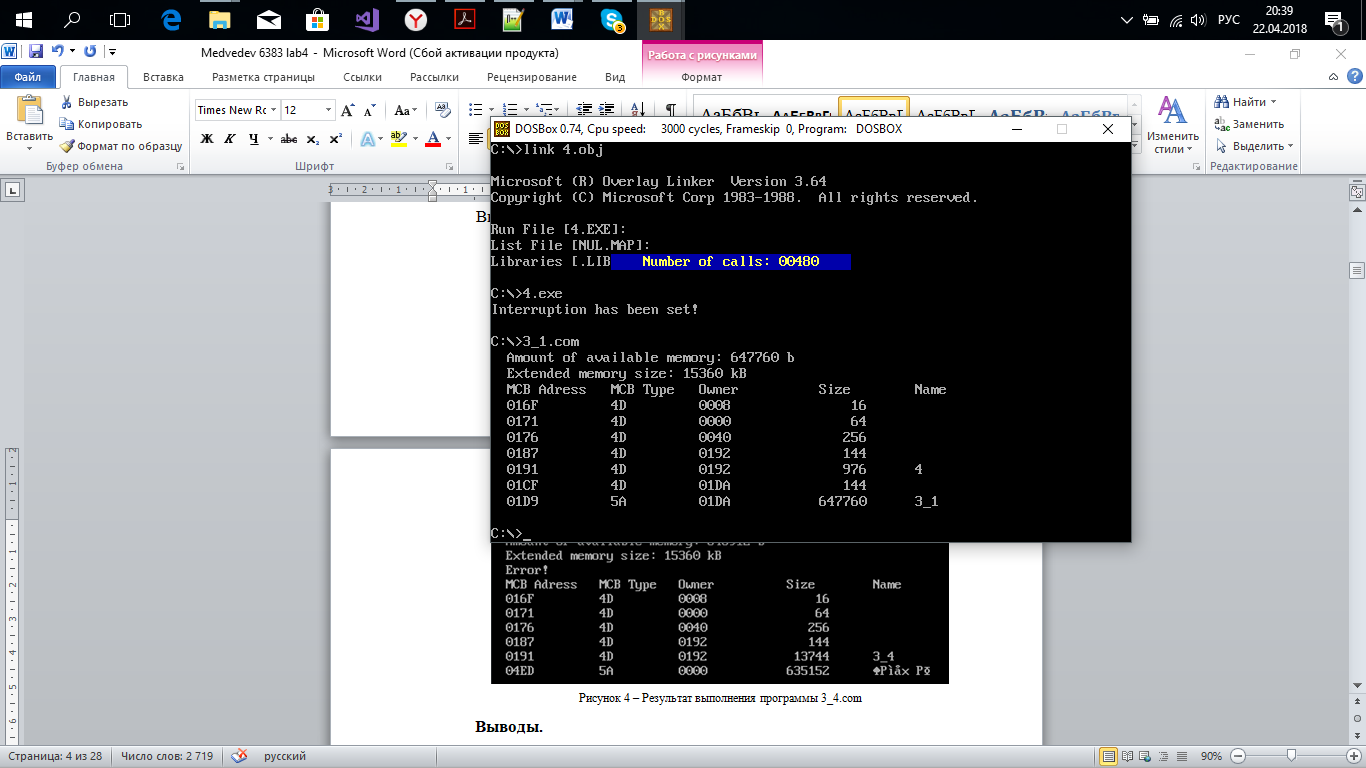


Рисунок 3 – Результат выполнения программы 3\_1.com

1. Повторный запуск программы 4.exe изображён на Рис. 4.

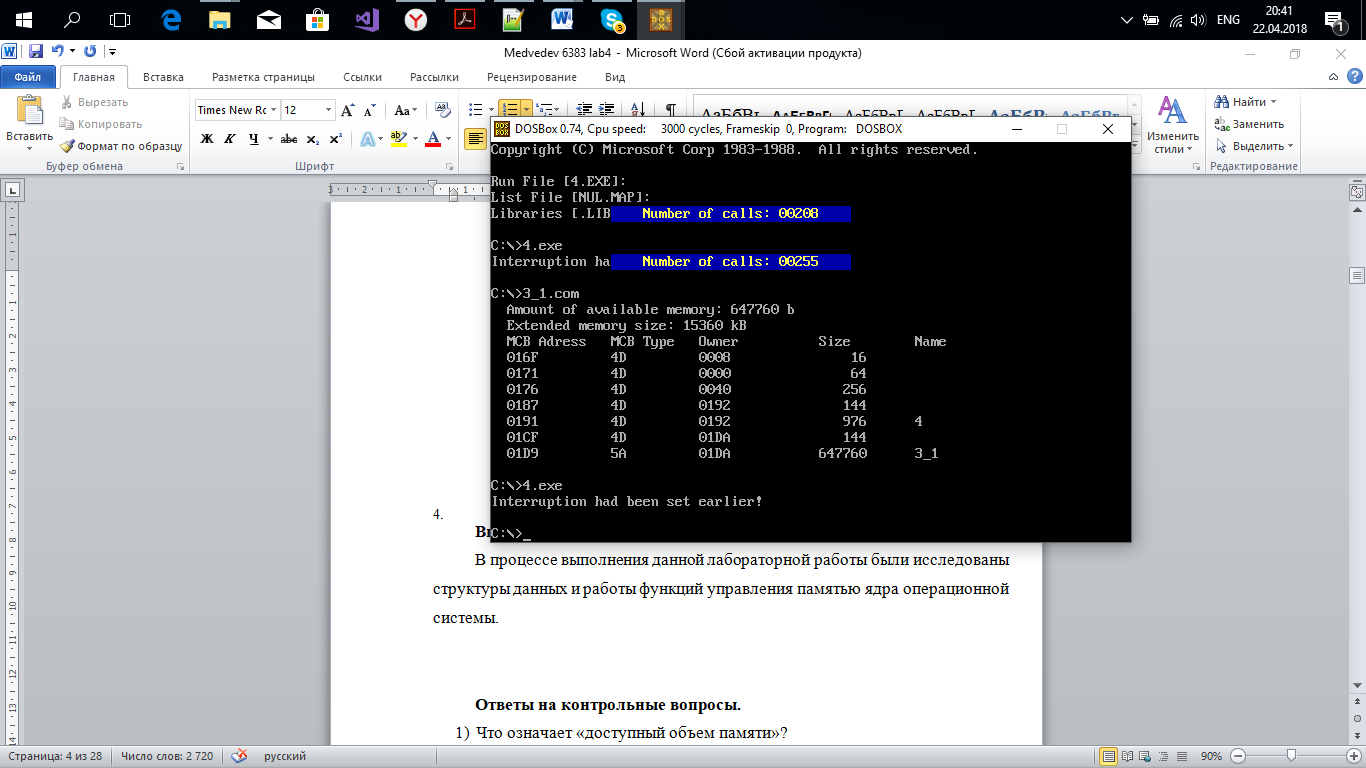


Рисунок 4 – Результат повторного выполнения программы 4.exe

1. Запустим программу 4.exe ещё раз, но с ключом выгрузки /un (см. Рис.5):

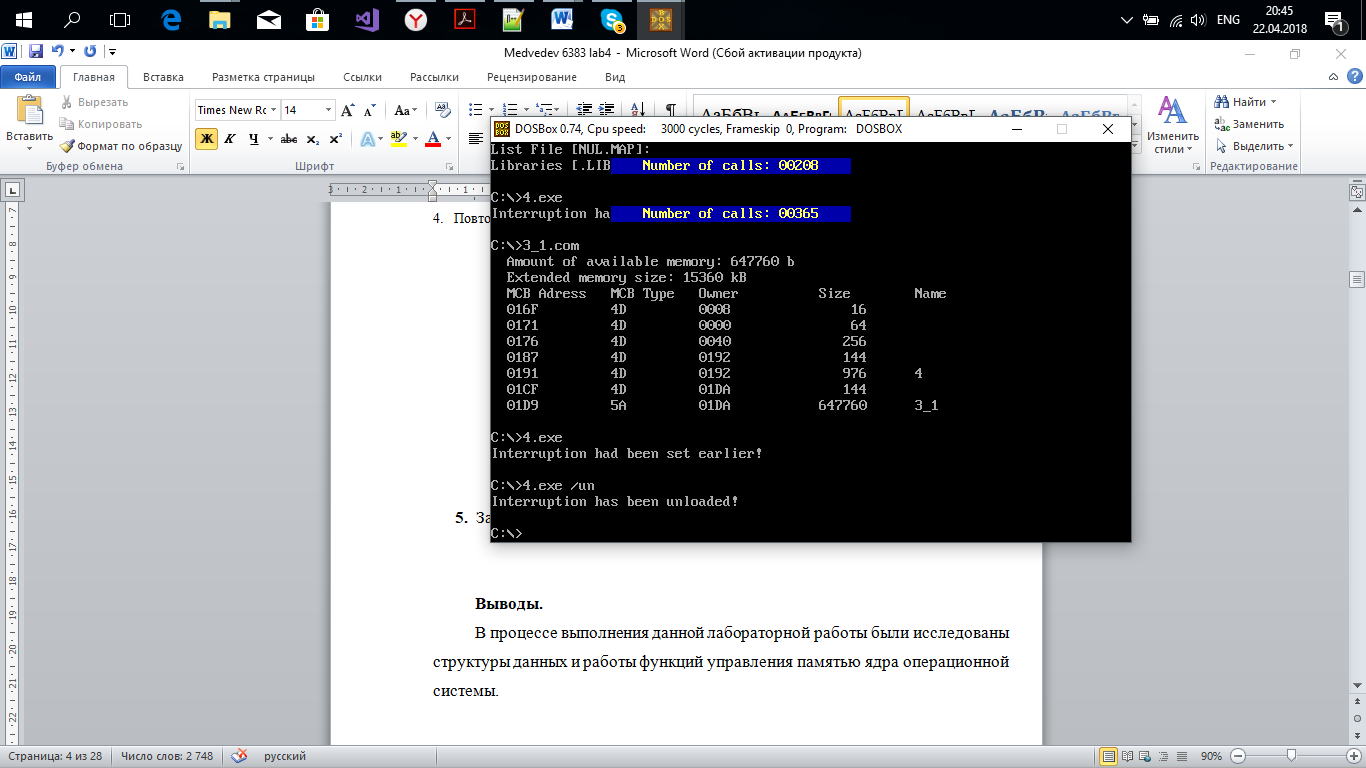


Рисунок 5 – Результат выполнения программы 4.exe с ключом /un

1. Запустим программу 3\_1.com, чтобы убедиться в том, что память освобождена.

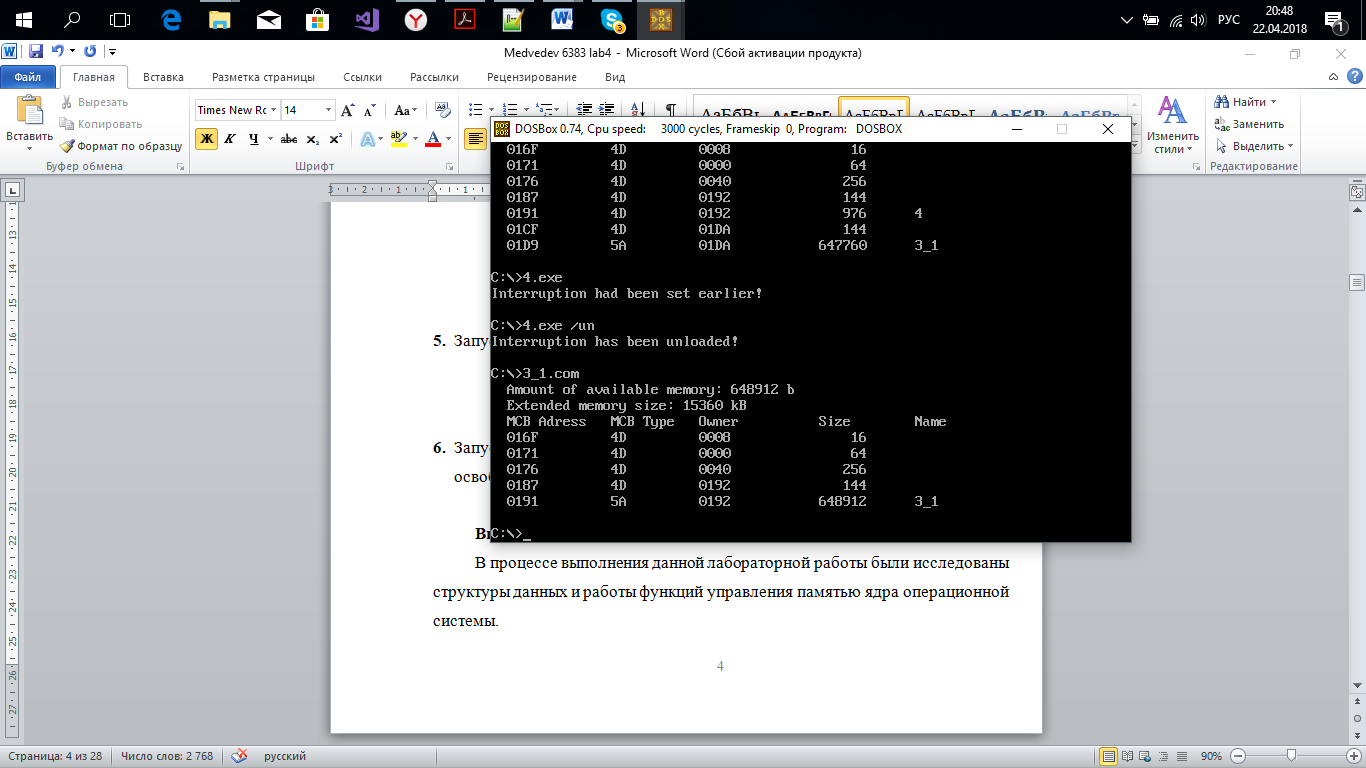


Рисунок 5 – Результат повторного выполнения программы 3\_1.com

**Выводы.**

В процессе выполнения данной лабораторной работы был построен обработчик прерываний сигналов таймера.

**Ответы на контрольные вопросы.**

1. Как реализован механизм прерывания от часов?

Данный механизм реализован следующим образом:

1. сохранение содержимого регистров;
2. определение источника прерывания, по номеру которого определяется смещение в таблице векторов прерывания;
3. сохранение в CS:IP;
4. передача управления по адресу CS:IP;
5. выполнение обработчика прерывания;
6. возврат управления прерванной программе.
7. Какого типа прерывания использовались в работе?

В данной лабораторной работе использовались аппаратные (int 1Сh) и программные (int 21h и int 10h) прерывания.

ПРИЛОЖЕНИЕ А

4.asm

STACK SEGMENT STACK

DW 100 DUP (?)

STACK ENDS

;---------------------------------------------------------------

DATA SEGMENT

wasloaded DB 'Interruption had been set earlier!',0DH,0AH,'$'

unloaded DB 'Interruption has been unloaded!',0DH,0AH,'$'

loading DB 'Interruption has been set!',0DH,0AH,'$'

DATA ENDS

;---------------------------------------------------------------

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK

START: JMP BEGIN

;---------------------------------------------------------------

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

;---------------------------------------------------------------

setCurs PROC ; cursor position setting

push AX

push BX

push CX

mov AH,02h

mov BH,00h

int 10h

pop CX

pop BX

pop AX

ret

setCurs ENDP

;---------------------------------------------------------------

getCurs PROC ; function that determines the position and size of the cursor

push AX

push BX

push CX

mov AH,03h

mov BH,00h

int 10h

pop CX

pop BX

pop AX

ret

getCurs ENDP

;---------------------------------------------------------------

ROUT PROC FAR ; interrupt handler

jmp ROUT\_

\_DATA:

SIGN DB '0000'

KEEP\_CS DW 0

KEEP\_IP DW 0

KEEP\_PSP DW 0

VALUE DB 0

COUNTER DB ' Number of calls: 00000 $' ; counter

ROUT\_:

push AX

push DX

push DS

push ES

cmp VALUE, 1

je ROUT\_RES

call getCurs

push DX

mov DH,7

mov DL,15

call setCurs

ROUT\_SUM: ; counting the number of interrupts

push SI

push CX

push DS

push AX

mov AX,SEG COUNTER

mov DS,AX

mov bx, offset COUNTER

add bx, 22

mov si, 3

next\_:

mov ah, [bx+si]

inc ah

cmp ah, 58

jne ROUT\_NEXT

mov ah, 48

mov [bx+si], ah

dec si

cmp si, 0

jne next\_

ROUT\_NEXT:

mov [bx+si],ah

pop ds

pop si

pop bx

pop ax

push es

push bp

mov ax,SEG COUNTER

mov es,ax

mov ax,offset COUNTER

mov bp,ax

mov ah,13h

mov al,0

mov cx,30

mov bh,0

int 10h

pop bp

pop es

pop dx

call setCurs

jmp ROUT\_END

ROUT\_RES: ; restoring the interrupt vector

CLI

mov DX,KEEP\_IP

mov AX,KEEP\_CS

mov DS,AX

mov AH,25h

mov AL,1Ch

int 21h

mov ES, KEEP\_PSP

mov ES, ES:[2Ch]

mov AH, 49h

int 21h

mov ES, KEEP\_PSP

mov AH, 49h

int 21h

STI

ROUT\_END: ; restoring registers

pop ES

pop DS

pop DX

pop AX

iret

ROUT ENDP

;---------------------------------------------------------------

CHECKING PROC ; checking of interruption

mov AH,35h

mov AL,1Ch

int 21h

mov SI, offset SIGN

sub SI, offset ROUT

mov AX,'00'

cmp AX,ES:[BX+SI]

jne UNLOAD

cmp AX,ES:[BX+SI+2]

je LOAD

UNLOAD:

call SET\_INTERRUPT

mov DX,offset LAST\_BYTE

mov CL,4

shr DX,CL

inc DX

add DX,CODE

sub DX,KEEP\_PSP

xor AL,AL

mov AH,31h

int 21h

LOAD:

push ES

push AX

mov AX,KEEP\_PSP

mov ES,AX

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

jne BACK

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

jne BACK

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

je UNLOAD\_

BACK:

pop AX

pop ES

mov dx,offset wasloaded

call PRINT

ret

UNLOAD\_:

pop AX

pop ES

mov byte ptr ES:[BX+SI+10],1

mov dx,offset unloaded

call PRINT

ret

CHECKING ENDP

;---------------------------------------------------------------

SET\_INTERRUPT PROC ; adding a new interruption

push DX

push DS

mov AH,35h

mov AL,1Ch

int 21h

mov KEEP\_IP,BX

mov KEEP\_CS,ES

mov DX,offset ROUT

mov AX,seg ROUT

mov DS,AX

mov AH,25h

mov AL,1Ch

int 21h

pop DS

mov DX,offset loading

call PRINT

pop DX

ret

SET\_INTERRUPT ENDP

;---------------------------------------------------------------

BEGIN:

mov AX,DATA

mov DS,AX

mov KEEP\_PSP,ES

call CHECKING

xor AL,AL

mov AH,4Ch

int 21H

LAST\_BYTE:

CODE ENDS

END START