**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Операционные системы»**

Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Медведев Г.О. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

**Описание функций и структур данных.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| PRINT | вызывает функцию печати строки |
| 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 |
| DELETE\_INTERRUPT | удаляет пользовательское прерывание, восстанавливает прерывание по умолчанию |

**Последовательность действий, выполняемых утилитой.**

1. Проверяет, установлено ли пользовательское прерывание с вектором 09h.
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, выход в DOS.

**Результаты выполнения программ.**

1. Состояние памяти до запуска программы 5.exe представлено на Рис.1, в качестве программы изображена работа файла 3\_1.com.

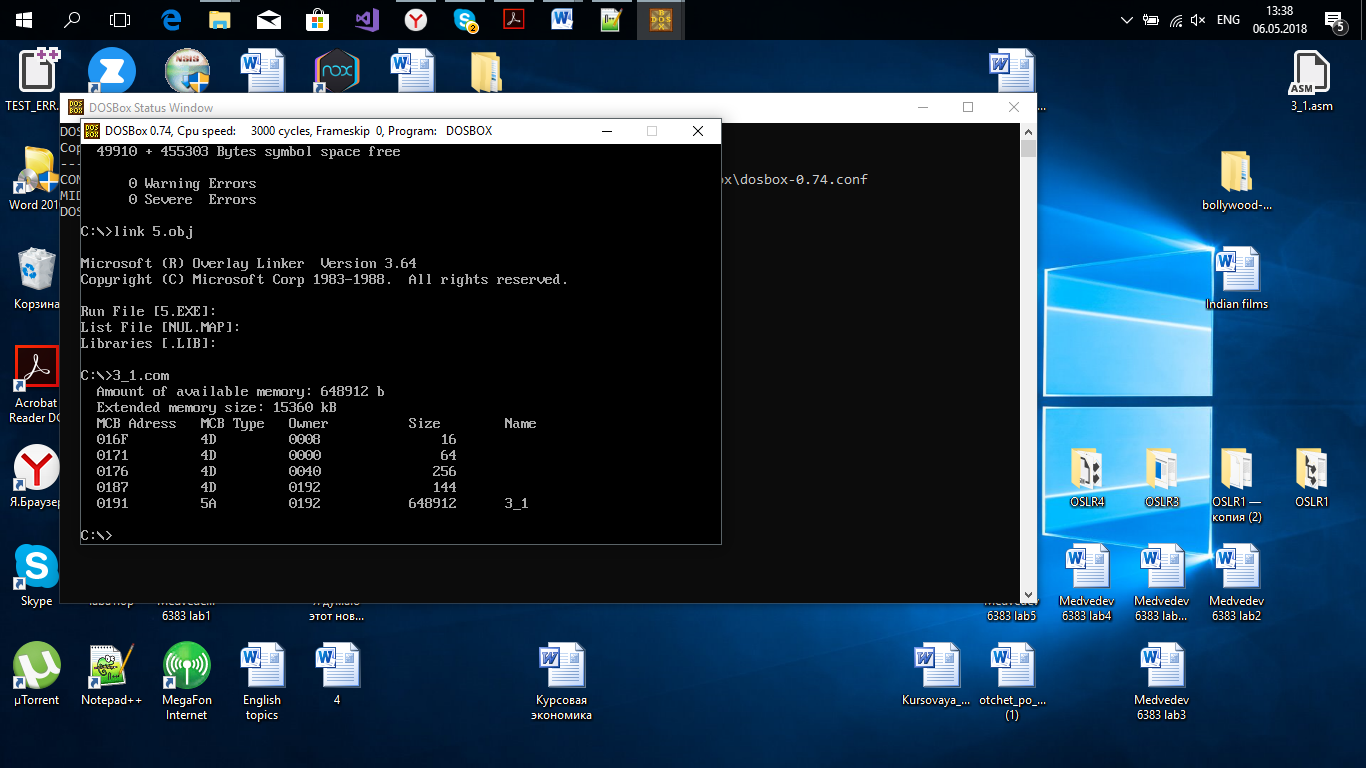


Рисунок 1 – Результат выполнения программы 3\_1.com

1. На Рис. 2 представлен запуск программы 5.exe

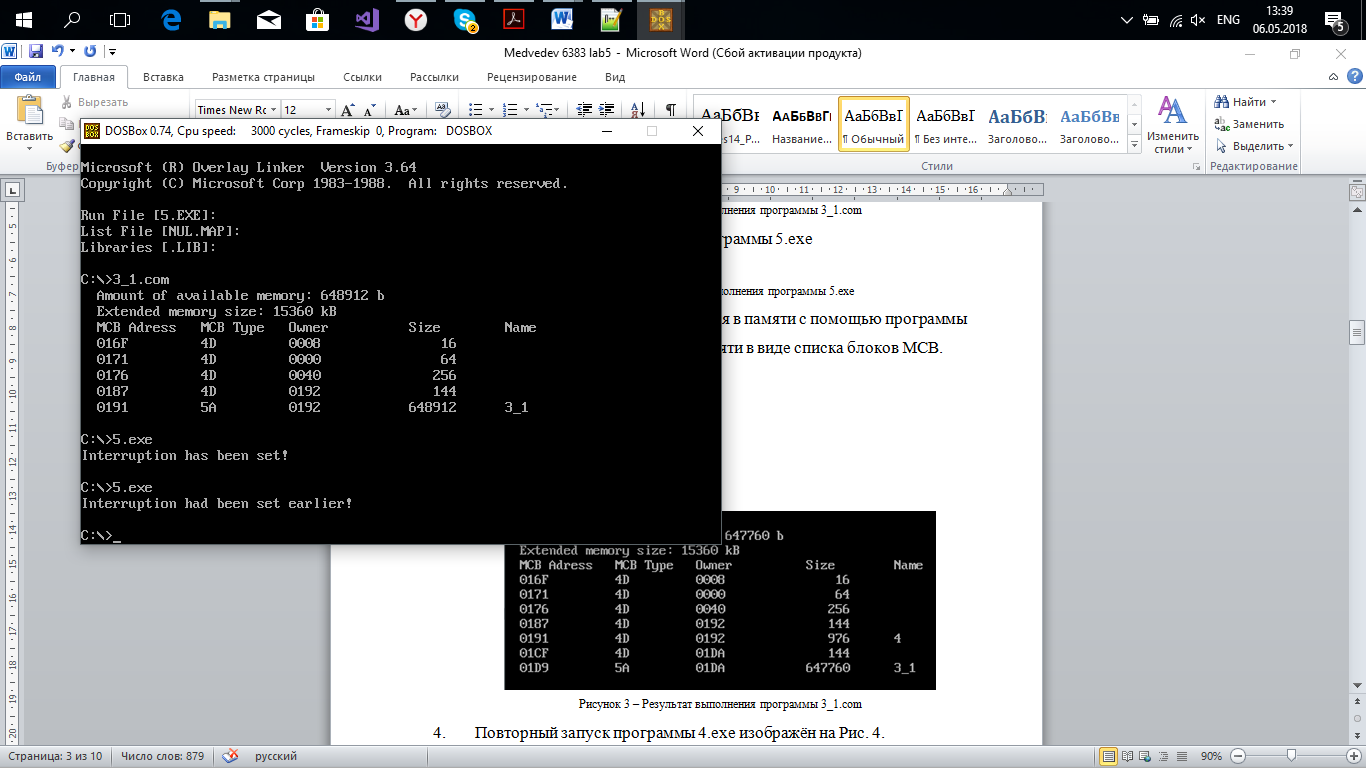


Рисунок 2 – Результат выполнения программы 5.exe

1. Проверим работу прерывания нажатием на различные клавиши. При нажатии на клавишу “~” печатается символ “\_”, при нажатии на остальные клавиши выполняется соответствующее действие. Действие программы показано на Рис. 3.

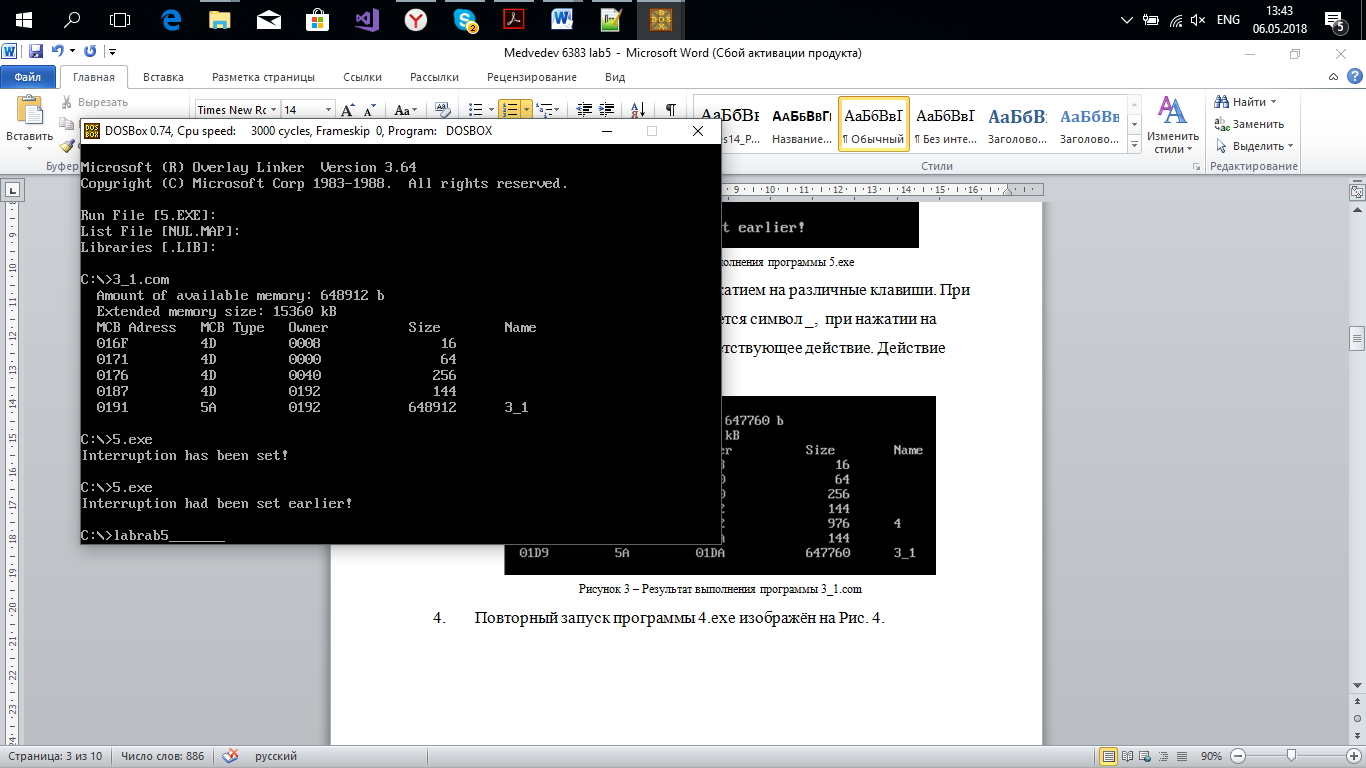


Рисунок 3 – Результат выполнения программы 5.exe

1. Проверим размещение прерывания в памяти в виде блоков МСВ. Выполнение программы 3\_1.com показано на Рис. 4.

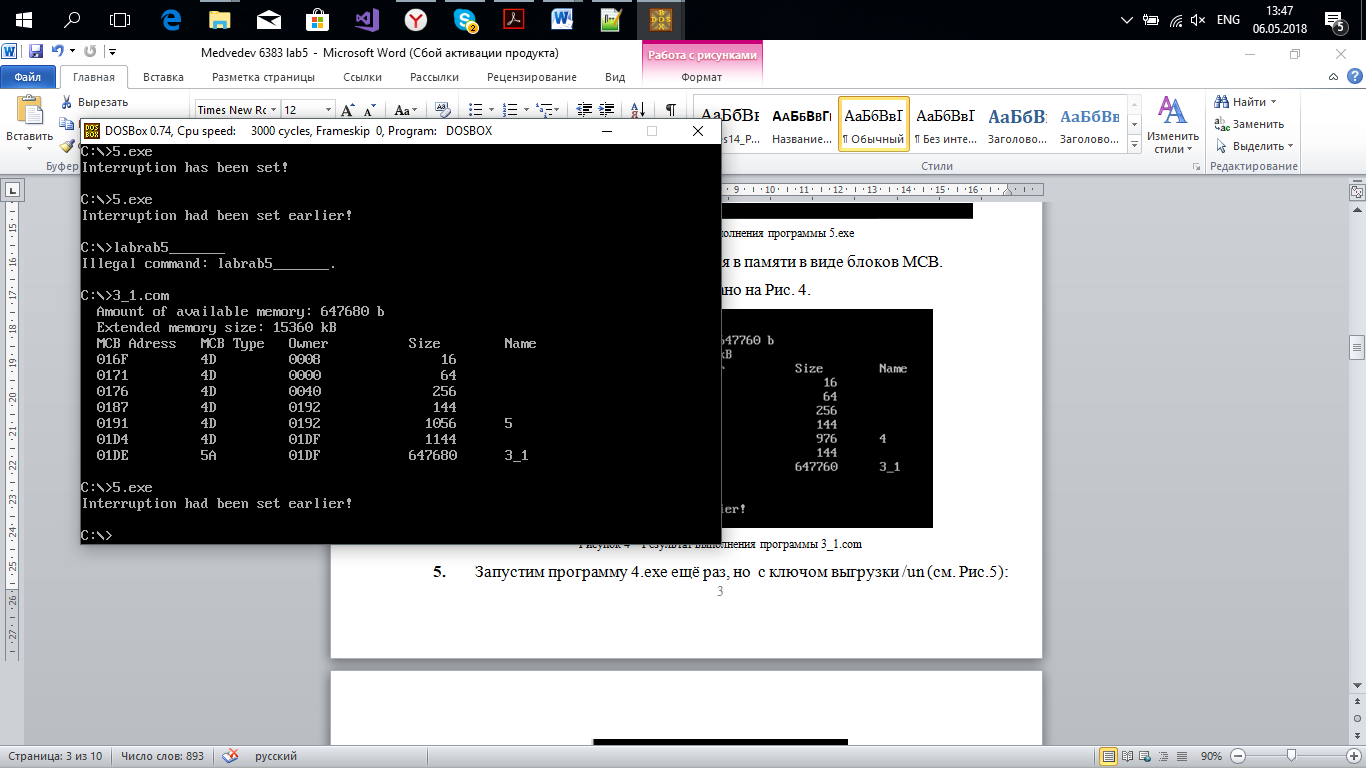


Рисунок 4 – Результат выполнения программы 3\_1.com

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

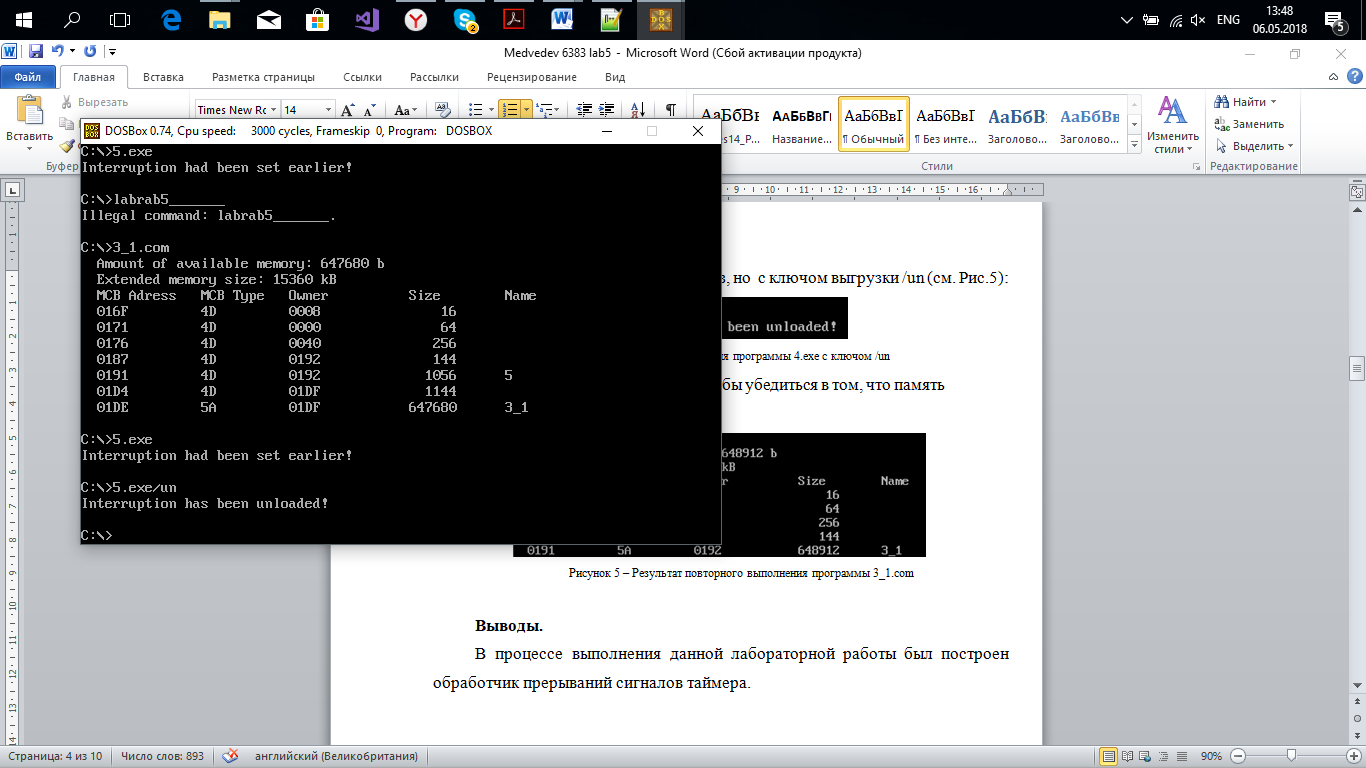


Рисунок 5 – Результат выполнения программы 5.exe с ключом /un

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

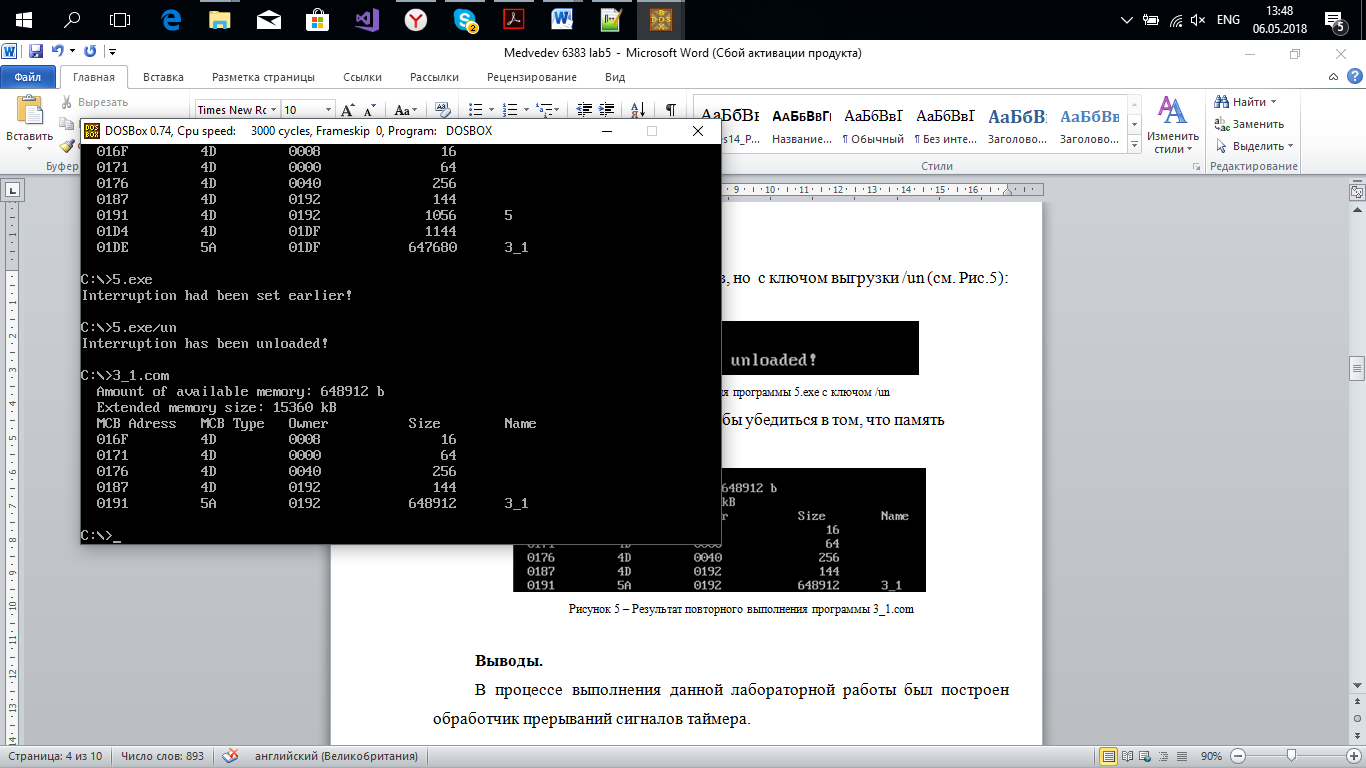


Рисунок 6 – Результат повторного выполнения программы 3\_1.com

**Выводы.**

В процессе выполнения данной лабораторной работы была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

**Ответы на контрольные вопросы.**

1. Какого типа прерывания использовались в работе?

В данной лабораторной работе использовались аппаратные прерывание (1Ch) и программные прерывания (21h).

1. Чем отличается скан-код от кода ASCII?

Код ASCII является числовым представлением какого-либо символа. Скан-код присваивается каждой клавише, с помощью него драйвер клавиатуры распознает, какая клавиша была нажата.

ПРИЛОЖЕНИЕ А

5.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 ; interrupt handler

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

ROUT PROC FAR

jmp ROUT\_

\_DATA:

STACK\_ DW 64 DUP (?)

SIGN DB '0000'

KEEP\_IP DW 0

KEEP\_CS DW 0

KEEP\_PSP DW 0

KEEP\_SS DW 0

KEEP\_AX DW 0

KEEP\_SP DW 0

ROUT\_:

mov KEEP\_SS, ss

mov KEEP\_AX, ax

mov KEEP\_SP, sp

mov ax, seg STACK\_

mov ss, ax

mov sp, 0

mov ax, KEEP\_AX

push ax

push dx

push ds

push es

in al, 60H

cmp al, 29h

je DO\_REQ

pushf

call dword ptr CS:KEEP\_IP

jmp ROUT\_END

DO\_REQ:

push ax

in al, 61h

mov ah, al

or al, 80h

out 61h, al

xchg ah, al

out 61h, al

mov al, 20h

out 20h, al

pop ax

ADDSYMB:

mov ah, 05h

mov cl, '\_'

mov ch, 00h

int 16h

or al, al

jz ROUT\_END

CLI

mov ax,es:[1Ah]

mov es:[1Ch],ax

STI

jmp ADDSYMB

ROUT\_END:

pop es

pop ds

pop dx

pop ax

mov AX, KEEP\_SS

mov SS, AX

mov SP,KEEP\_SP

mov AX,KEEP\_AX

iret

ROUT ENDP

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

CHECKING PROC ; checking of interruption

mov ah,35h

mov al,09h

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,CS: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

call DELETE\_INTERRUPT

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,09h

int 21h

mov CS:KEEP\_IP,bx

mov CS:KEEP\_CS,es

mov dx,offset ROUT

mov ax,seg ROUT

mov ds,ax

mov ah,25h

mov al,09h

int 21h

pop ds

mov dx,offset loading

call PRINT

pop dx

ret

SET\_INTERRUPT ENDP

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

DELETE\_INTERRUPT PROC ;delete an interrupt

push ds

CLI

mov dx,ES:[BX+SI+4]

mov ax,ES:[BX+SI+6]

mov ds,ax

mov ax,2509h

int 21h

push es

mov ax,ES:[BX+SI+8]

mov es,ax

mov es,es:[2Ch]

mov ah,49h

int 21h

pop es

mov es,ES:[BX+SI+8]

mov ah, 49h

int 21h

STI

pop ds

ret

DELETE\_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