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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Операционные системы»

Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

Студент гр. 7381 _____ Трушников А.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерывания получает управление по прерыванию (int 09h) при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

Описание функций.

Название	Описание
Write_message	Вывод сообщения на экран
My_2F	Собственный обработчик прерывания для 2F. Проверяет, была ли программа установлена резидентной в памяти
My_09	Собственный обработчик прерывания для 09. Проверяет, не является ли введённый символ одним из следующих: «6», «7» или «8». Если это так – заменят его соответственно на «A», «B» или «C». Иначе передаёт управление стандартному обработчику прерываний.
Un_check	Проверяет, нет указал ли пользователь флаг «/un» при вызове программы
Keep_interr	Запоминает старые обработчики прерывания, используя функцию 35h прерывания int 21h
Load_interr	Устанавливает новые обработчики прерывания, используя функцию 25h прерывания int 21h
Unload_interr	Восстанавливает сохранённые заранее обработчики прерываний и выгружает резидентную программу
Make_resident	Оставляет программу резидентной в памяти
Main	Основная функция

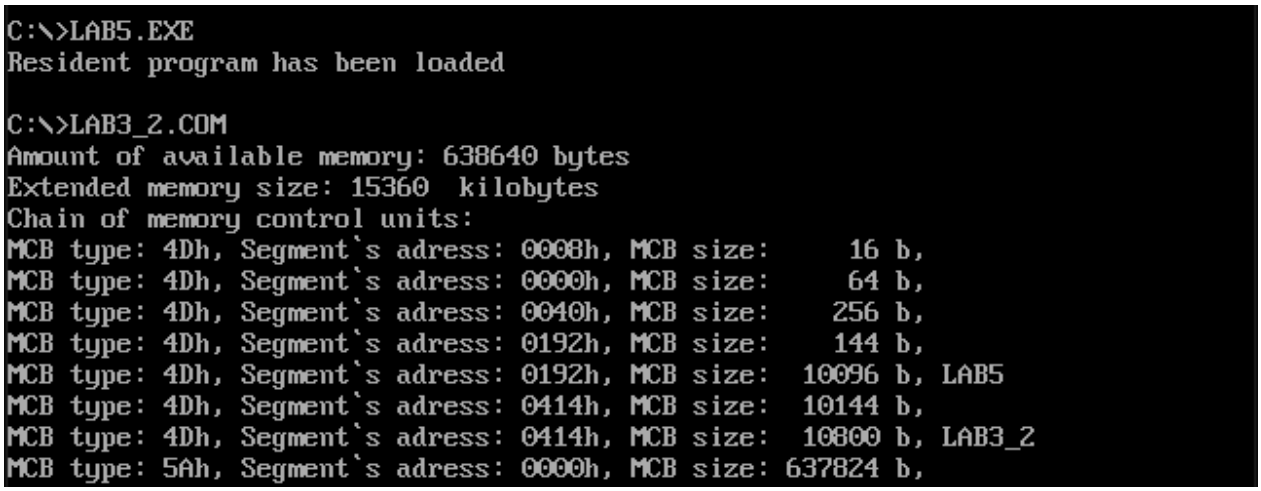
Описание структур данных.

Название	Описание
flag	флаг, равный 1, если программа не является резидентной, и 0, если наоборот
Message1	Сообщение о том, что программа только что была загружена в память

	резидентной
Message2	Сообщение о том, что резидентная программа была выгружена из памяти
Message3	Сообщение, выдающееся при попытке повторно оставить программу резидентной в памяти
keep_09	Переменная для хранения сегмента и смещения старого прерывания 09
keep_2f	Переменная для хранения сегмента и смещения старого прерывания 2f
keep_PSP	Переменная для хранения старого значения ES до того, как программа была оставлена резидентной в памяти
REQ_KEY_1	Скан-код клавиши «6»
REQ_KEY_2	Скан-код клавиши «7»
REQ_KEY_3	Скан-код клавиши «8»

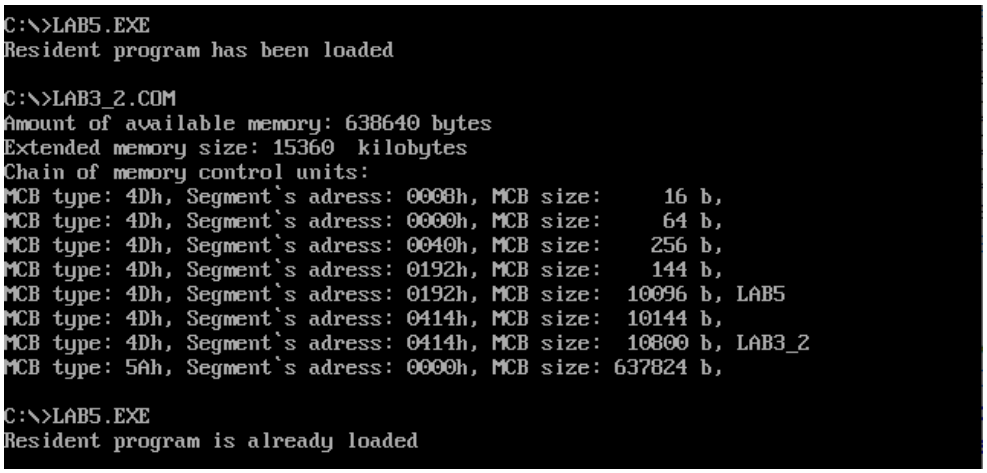
Выполнение работы.

Шаг 1.



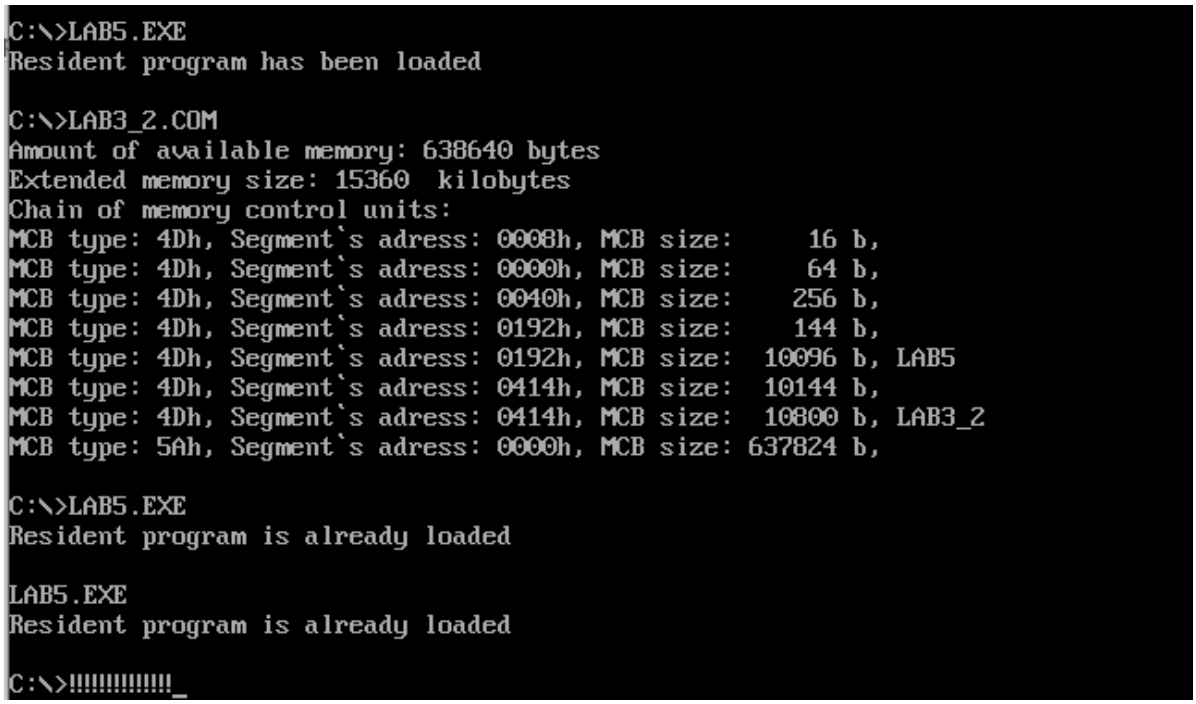
Убедились, что программа отображает результат работы прерывания и остаётся резидентной.

Шаг 2.



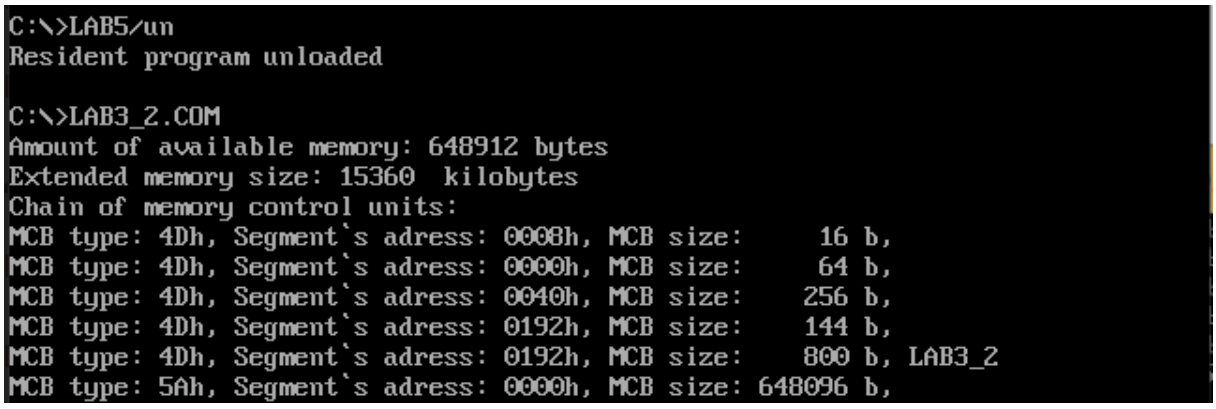
Убедились, что программа распознаёт то, что она уже была загружена резидентной в память.

Шаг 3.



Проверка работы прерывания при нажатии различных клавиш. При нажатии Ctrl+s выводиться знак «!»

Шаг 4.



Программа успешно выгружается по команде «/un».

Ответы на контрольные вопросы.

1. Какого типа прерывания использовались в работе?

В работе использовались аппаратное прерывание int 09h, прерывание BIOS int 16h, а также пользовательские прерывания int 2fh и int 21h.

2. Чем отличается скан код от кода ASCII?

Скан-код – код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознаёт, какая клавиша была нажата. ASCII-код – код, определяющий закреплённый за клавишей символ.

Выводы.

В ходе данной лабораторной работы я построил собственный обработчик прерывания для аппаратного прерывания 09h, возникающего при нажатии клавиши на клавиатуре, которое сравнивает полученный скан-код с имеющимися в программе и выводит на экран другой символ вместо того, который должен быть выведен стандартным обработчиком.

ПРИЛОЖЕНИЕ А

LAB5.ASM

```
CODE   SEGMENT
        KEEP_09         DD 0 ;ПЕРЕМЕННАЯ ДЛЯ ХРАНЕНИЯ СЕГМЕНТА И СМЕЩЕНИЯ СТАРОГО ПРЕРЫВАНИЯ 09
        KEEP_2F         DD 0 ;ПЕРЕМЕННАЯ ДЛЯ ХРАНЕНИЯ СЕГМЕНТА И СМЕЩЕНИЯ СТАРОГО ПРЕРЫВАНИЯ 2F
        REQ_KEY_1   DB 7H ;ЦИФРА 6 БУДЕТ ЗАМЕНЯТЬ НА СИМВОЛ 'A'
        REQ_KEY_2   DB 8H ;ЦИФРА 7 БУДЕТ ЗАМЕНЯТЬ НА СИМВОЛ 'B'
        REQ_KEY_3   DB 9H ;ЦИФРА 8 БУДЕТ ЗАМЕНЯТЬ НА СИМВОЛ 'C'
        KEEP_PSP    DW ?
    ASSUME CS:CODE, DS:DATA, SS:ASTACK


;СОБСТВЕННЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ ДЛЯ 2F
MY_2F PROC
        CMP   AH, 080H ;СРАВНИВАЕМ ЗНАЧЕНИЕ В AH С УСТАНОВЛЕННЫМ РАНЕЕ ПЕРЕД ПРЕРЫВАНИЕМ
        JNE   NOT_LOADED ;ЕСЛИ ЗНАЧЕНИЯ НЕ РАВНЫ - ПРОГРАММА НЕ УСТАНОВЛЕНА РЕЗИДЕНТНОЙ В ПАМЯТИ
        MOV   AL, 0FFH ;УСТАНАВЛИВАЕМ В AL ЗНАЧЕНИЕ FF, ЧТО НА ВЫХОДЕ ИЗ ПРЕРЫВАНИЯ ОЗНАЧАЕТ, ЧТО ПРОГРАММА УСТАНОВЛЕНА
    NOT_LOADED:
        IRET ;ИНАЧЕ ПРОСТО ВОЗВРАЩАЕМСЯ В ПРОГРАММУ
MY_2F ENDP

;СОБСТВЕННЫЙ ОБРАБОТЧИК ПРЕРЫВАНИЯ ДЛЯ 1C
MY_09 PROC
        PUSH AX

        IN AL, 60H ;ЧИТАЕМ КЛЮЧ
        CMP AL, REQ_KEY_1 ;ПРОВЕРЯЕМ, НУЖНЫЙ ЛИ НАМ КЛЮЧ
        JE KEY1 ;ЕСЛИ ДА (ЦИФРА 6), ТО БУДЕМ ЗАМЕНЯТЬ ЕЁ НА 'A'

        CMP AL, REQ_KEY_2 ;ПРОВЕРЯЕМ, НУЖНЫЙ ЛИ НАМ КЛЮЧ
        JE    KEY2 ;ЕСЛИ ДА (ЦИФРА 7), ТО БУДЕМ ЗАМЕНЯТЬ ЕЁ НА 'B'

        CMP AL, REQ_KEY_3 ;ПРОВЕРЯЕМ, НУЖНЫЙ ЛИ НАМ КЛЮЧ
        JE    KEY3 ;ЕСЛИ ДА (ЦИФРА 8), ТО БУДЕМ ЗАМЕНЯТЬ ЕЁ НА 'C'

        POP AX
        JMP   DWORD PTR CS:[KEEP_09] ;ПЕРЕХОДИМ НА ПЕРВОНАЧАЛЬНЫЙ ОБРАБОТЧИК

    KEY1:
        MOV CL, 'A'
        JMP DO_REQ
```

```
KEY2:
        MOV CL, 'B'
        JMP DO_REQ
KEY3:
        MOV CL, 'C'
DO_REQ:
        IN      AL, 61H ;ВЗЯТЬ ЗНАЧЕНИЕ ПОРТА УПРАВЛЕНИЯ КЛАВАИАТУРОЙ
        MOV AH, AL  ;СОХРАНИТЬ ЕГО
        OR      AL, 80H ;УСТАНОВИТЬ БИТ РАЗРЕШЕНИЯ ДЛЯ КЛАВИАТУРЫ
        OUT 61H, AL ;И ВЫВЕСТИ ЕГО В УПРАВЛЯЮЩИЙ ПОРТ
        XCHG AH, AL ;ИЗВЛЕЧЬ ИСХОДНОЕ ЗНАЧЕНИЕ ПОРТА
        OUT 61H, AL ;И ЗАПИСАТЬ ЕГО ОБРАТНО
        MOV AL, 20H ;ПОСЛАТЬ СИГНАЛ "КОНЕЦ ПРЕРЫВАНИЯ"
        OUT 20H, AL ;КОНТРОЛЛЕРУ ПРЕРЫВАНИЙ 8259

        MOV AH, 05H ;ФУНКЦИЯ, ПОЗВОЛЯЮЩУАЯ ЗАПИСАТЬ СИМВОЛ В БУФЕР
КЛАВИАТУРЫ
        MOV CH, 00H ;СИМВОЛ В CL УЖЕ ЗАНЕСЁН РАНЕЕ, ОСТАЛОСЬ ОБНУЛИТЬ
CH
        INT 16H             ;ВЫПОЛНЯЕМ ФУНКЦИЮ
        OR      AL, AL ;ПРОВЕРКА ПЕРЕПОЛНЕНИЯ БУФЕРА
        JNZ SKIP        ;ЕСЛИ ПЕРЕПОЛНЕН - ИДЁМ В SKIP
        JMP RETURN  ;ИНАЧЕ  ПОДАЁМ  СИГНАЛ  "КОНЕЦ  ПРЕРЫВАНИЯ"  И
ВЫХОДИМ
SKIP:               ;ОЧИЩАЕМ БУФЕР
        PUSH ES
        PUSH SI
        MOV AX, 0040H
        MOV ES, AX
        MOV SI, 001AH
        MOV AX, ES:[SI]
        MOV SI, 001CH
        MOV ES:[SI], AX
        POP SI
        POP ES
RETURN:
        POP AX
        MOV AL, 20H
        OUT 20H, AL
        IRET
LAST_BYTE:
MY_09 ENDP

; ФУНКЦИЯ ПРОВЕРКИ НЕ ВВЁЛ ЛИ ПОЛЬЗОВАТЕЛЬ КОМАНДУ /UN
UN_CHECK  PROC      FAR
        PUSH AX

        MOV    AX, KEEP_PSP
```

```
                MOV    ES, AX
                SUB    AX, AX
                CMP    BYTE PTR ES:[82H],'/'
                JNE    NOT_UN
                CMP    BYTE PTR ES:[83H],'U'
                JNE    NOT_UN
                CMP    BYTE PTR ES:[84H],'N'
                JNE    NOT_UN
                MOV    FLAG,0

        NOT_UN:
                POP    AX
                RET
UN_CHECK  ENDP

;ФУНКЦИЯ, СОХРАНЯЮЩАЯ СТАНДАРТНЫЕ ОБРАБОТЧИКИ ПРЕРЫВАНИЙ
KEEP_INTERR  PROC
                PUSH AX
                PUSH BX
                PUSH ES

                MOV AH, 35H ;ФУНКЦИЯ, ВЫДАЮЩАЯ ЗНАЧЕНИЕ СЕГМЕНТА В ES,
СМЕЩЕНИЕ В BX
                MOV AL, 09H ;ДЛЯ ПРЕРЫВАНИЯ 09
                INT 21H
                MOV WORD PTR KEEP_09, BX
                MOV WORD PTR KEEP_09+2, ES

                MOV AH, 35H ;ФУНКЦИЯ, ВЫДАЮЩАЯ ЗНАЧЕНИЕ СЕГМЕНТА В ES,
СМЕЩЕНИЕ В BX
                MOV AL, 2FH ;ДЛЯ ПРЕРЫВАНИЯ 2F
                INT 21H
                MOV WORD PTR KEEP_2F, BX
                MOV WORD PTR KEEP_2F+2, ES

                POP ES
                POP BX
                POP AX
                RET
KEEP_INTERR  ENDP

; ФУНКЦИЯ, ЗАГРУЖАЮЩАЯ СОБСТВЕННЫЕ ОБРАБОТЧИКИ ПРЕРЫВАНИЯ
LOAD_INTERR          PROC
                PUSH DS
                PUSH DX
                PUSH AX

                CALL KEEP_INTERR ;СОХРАНЯЕМ СТАРЫЕ ОБРАБОТЧИКИ ПРЕРЫВАНИЙ
```

```
                PUSH DS
                MOV DX, OFFSET MY_09
                MOV AX, SEG MY_09
                MOV DS, AX
                MOV AH, 25H          ;ФУНКЦИЯ, МЕНЯЮЩАЯ ОБРАБОТЧИК
ПРЕРЫВАНИЙ НА УКАЗАННЫЙ В DX И AX
                MOV AL, 09H     ;ДЛЯ ПРЕРЫВАНИЯ 1C
        INT 21H

        MOV DX, OFFSET MY_2F
                MOV AX, SEG MY_2F
                MOV DS, AX
                MOV AH, 25H          ;ФУНКЦИЯ, МЕНЯЮЩАЯ ОБРАБОТЧИК
ПРЕРЫВАНИЙ НА УКАЗАННЫЙ В DX И AX
                MOV AL, 2FH     ;ДЛЯ ПРЕРЫВАНИЯ 2F
        INT 21H
                POP DS

                POP AX
                POP DX
                POP DS
                RET
LOAD_INTERR  ENDP

; ВЫГРУЖАЕМ ОБРАБОТЧИКИ ПРЕРЫВАНИЙ
UNLOAD_INTERR  PROC
                PUSH DS

                MOV AH, 35H
                MOV AL, 09H
                INT 21H
                MOV DX, WORD PTR ES:KEEP_09
                MOV AX, WORD PTR ES:KEEP_09+2
                MOV WORD PTR KEEP_09, DX
                MOV WORD PTR KEEP_09+2, AX

                MOV AH, 35H
                MOV AL, 2FH
                INT 21H
                MOV DX, WORD PTR ES:KEEP_2F
                MOV AX, WORD PTR ES:KEEP_2F+2
                MOV WORD PTR KEEP_2F, DX
                MOV WORD PTR KEEP_2F+2, AX

                CLI
                MOV DX, WORD PTR KEEP_09
                MOV AX,         WORD PTR KEEP_09+2
```

```
                MOV DS, AX
                MOV AH, 25H  ;ВЫГРУЖАЕМ ОБРАБОТЧИК ДЛЯ 09
                MOV AL, 09H
                INT 21H

                MOV DX, WORD PTR KEEP_2F
                MOV AX,      WORD PTR KEEP_2F+2
                MOV DS, AX
                MOV AH, 25H  ;ВЫГРУЖАЕМ ОБРАБОТЧИК ДЛЯ 2F
                MOV AL, 2FH
                INT 21H
                STI

                POP DS

                MOV ES, ES:KEEP_PSP
                MOV AX, 4900H           ;ОСВОБОЖДАЕМ ПАМЯТЬ ПО АДРЕСУ
ES:KEEP_PSP
                INT 21H

                MOV FLAG, 1             ;ЗАПОМИНАЕМ, ЧТО ПАМЯТЬ БЫЛА
ОСВОБОЖДЕНА
                MOV DX, OFFSET MESSAGE2
                CALL WRITE_MESSAGE  ;ВЫВОДИМ СООТВЕТСТВУЮЩЕЕ СООБЩЕНИЕ

                MOV ES, ES:[2CH]
                MOV AX, 4900H           ;ОСВОБОЖДАЕМ ПАМЯТЬ ПО АДРЕСУ
ES:[2CH]
                INT 21H

                MOV AX, 4C00H           ;ВЫХОД ИЗ ПРОГРАММЫ ЧЕРЕЗ ФУНКЦИЮ 4C
                INT 21H
        UNLOAD_INTERR  ENDP

        MAKE_RESIDENT  PROC
                MOV AX, ES
                MOV KEEP_PSP, AX
                MOV DX, OFFSET LAST_BYTE
                ADD DX, 200H

                MOV AH, 31H ;31H ЗАВЕРШАЕТ ПРОГРАММУ, ОСТАВЛЯЯ ЕЁ
РЕЗИДЕНТНОЙ В ПАМЯТИ
                MOV AL, 0
                INT 21H
        MAKE_RESIDENT  ENDP

        ; ФУНКЦИЯ ВЫВОДА СООБЩЕНИЯ НА ЭКРАН
        WRITE_MESSAGE      PROC
```

```
                PUSH AX
                MOV AH, 09H
                INT 21H
                POP AX
                RET
WRITE_MESSAGE           ENDP

; ГЛАВНАЯ ФУНКЦИЯ
MAIN   PROC
                PUSH DS
                XOR AX, AX
                PUSH AX
                MOV AX, DATA
                MOV DS, AX
                MOV KEEP_PSP, ES

                MOV    AX, 8000H ;НАМ НУЖНЫ НОМЕРА В АН ОТ 80H ДО 0FFH
                INT 2FH
                CMP    AL,0FFH        ; 2FH ВОЗВРАЩАЕТ 0FFH, ЕСЛИ ПРОГРАММА
УСТАНОВЛЕНА РЕЗИДЕНТНОЙ В ПАМЯТИ
                JNE LOADING

                CALL UN_CHECK
                CMP FLAG, 0
                JNE ALR_LOADED

                CALL UNLOAD_INTERR      ;ПОЛЬЗОВАТЕЛЬ ВВЁЛ /UN И ПРОГРАММА
ЕЩЁ НЕ БЫЛА ВЫГРУЖЕНА
        LOADING:                        ;ПРОГРАММА НЕ ЯВЛЯЕТСЯ РЕЗИДЕНТНОЙ В
ПАМЯТИ
                CALL LOAD_INTERR

                LEA DX, MESSAGE1
                CALL WRITE_MESSAGE

                CALL MAKE_RESIDENT
        ALR_LOADED:                     ;ПРОГРАММА УЖЕ БЫЛА
РЕЗИДЕНТНОЙ
                LEA DX, MESSAGE3
                CALL WRITE_MESSAGE
                MOV AX, 4C00H
                INT 21H
MAIN   ENDP
CODE            ENDS

ASTACK          SEGMENT  STACK
    DW 256 DUP(?)
ASTACK  ENDS
```

```
DATA            SEGMENT
   FLAG                    DW 1
   MESSAGE1      DB 'RESIDENT PROGRAM HAS BEEN LOADED', 0DH, 0AH, '$'
   MESSAGE2     DB 'RESIDENT PROGRAM UNLOADED', 0DH, 0AH, '$'
   MESSAGE3          DB 'RESIDENT PROGRAM IS ALREADY LOADED', 0DH, 0AH, '$'
DATA            ENDS
        END MAIN
```