

**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.0382

Рубежова Н.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

### **Цель работы.**

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определенные вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передает управление по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

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.

### **Задание.**

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет следующие функции:

- 1) Проверяет, установлено ли пользовательское прерывание с вектором 1Ch;
- 2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход по функции 4Ch прерывания int 21h;
- 3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
- 4) Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Для того, чтобы проверить установку прерывания, можно поступить следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код, который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом, находящимся в резиденте, можно определить, установлен ли резидент. Если значения совпадают, то резидент установлен. Длина кода сигнатуры должна быть достаточной, чтобы сделать случайное совпадение маловероятным.

Программа должна содержать код устанавливаемого прерывания в виде удаленной процедуры. Этот код и будет работать после установки при возникновении прерывания. Он должен выполнять следующие функции:

- 1) Сохранить значения регистров SS и SP прерванной программы в рабочих переменных;
- 2) Организовать свой стек;
- 3) Сохранить значения регистров в стеке при входе и восстановить их при выходе;

4) При выполнении тела процедуры накапливать общее суммарное число прерываний и выводить на экран. Для вывода на экран следует использовать прерывание `int 10h`, которое позволяет непосредственно выводить информацию на экран.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания `1Ch` установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛРЗ, которая отображает карту памяти в виде списка блоков МСВ. Полученные результаты поместите в отчет.

Шаг 3. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

Шаг 4. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, т.е. сообщения на экран не выводятся, а память, занятая резидентом, освобождена. Для этого также следует запустить программу ЛР3. Полученные результаты поместите в отчет.

### **Выполнение работы.**

Шаг 1. Был написан .EXE модуль, в котором реализованы следующие процедуры:

USR\_INTERRUPT – пользовательский обработчик прерывания, который считает и выводит количество вызванных прерываний на экран с помощью вызова int 10h;

IS\_SET – процедура проверки, что пользовательский обработчик прерывания установлен. Проверяется сигнатура, установленная в обработчике прерывания. Помещает в переменную RESULT 1, если обработчик установлен; 0 – в случае, если не установлен.

CHECK\_COMMAND – проверка на наличие в командной строке параметра /un. Помещает в переменную RESULT 0, если параметр не установлен; 1 – если установлен.

USR\_INT\_SET – установка пользовательского обработчика прерывания. Сохраняет оригинальный вектор прерывания (получили через функцию 35h прерывания int 21h), устанавливает пользовательский обработчик прерывания (через функцию 25h прерывания int 21h) и для того, чтобы оставить процедуру прерывания резидентной в памяти использует функцию DOS 31h прерывания int 21h.

USR\_INT\_UNLOAD – процедура выгрузки обработчика прерывания.

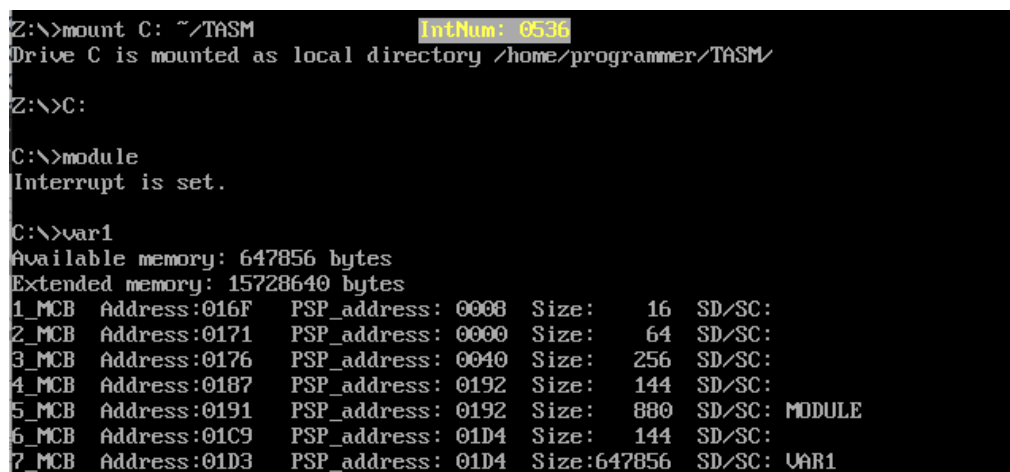
В процедуре MAIN вызывается процедура CHECK\_COMMAND для проверки наличия параметра /un в командной строке.

В случае, если нашли параметр /un в командной строке, переменная RESULT равна 1, то нам необходимо выгрузить обработчик прерываний(восстановить стандартный вектор прерывания), значит переходим

на метку `int_not_set`. Если на данный момент обработчик не установлен, т.е. переменная `RESULT` после вызова `IS_SET` равна 0, то выгружать нам нечего, поэтому просто выводим соответствующее сообщение «User interrupt was unloaded». Если обработчик уже был установлен, то сначала нам нужно его выгрузить, поэтому вызываем `USR_INT_UNLOAD` и после сообщаем, что обработчик теперь не установлен «User interrupt was unloaded».

В случае, если параметр `/un` в командной строке отсутствует, т.е. переменная `RESULT` равна 0, то нам необходимо установить пользовательский обработчик прерываний. Для этого сначала проверим, не установлен ли он уже вызовом `IS_SET`. Если значение переменной `RESULT` после вызова процедуры равно 1, т.е. пользовательский обработчик прерываний уже установлен, то выведем соответствующее сообщение на экран «User interrupt has already set». В противном случае нам нужно его установить: переходим к метке `int_set`, вызываем процедуру установки обработчика прерываний `USR_INT_SET` и выводим на экран «User interrupt was set».

Шаг 2. Запустим программу `module.asm`, обработчик прерывания установлен, количество вызовов прерывания отображается на экране. Далее запустим `var1.asm` из предыдущей лабораторной работы, чтобы убедиться, что обработчик прерывания загружен в память. Результат см. на рис. 1. Видим, что есть блок памяти, используемый обработчиком прерывания.



```
Z:\>mount C: ~/TASM
Drive C is mounted as local directory /home/programmer/TASM/

Z:\>C:

C:\>module
Interrupt is set.

C:\>var1
Available memory: 647856 bytes
Extended memory: 15728640 bytes
1_MCB Address:016F PSP_address: 0008 Size: 16 SD/SC:
2_MCB Address:0171 PSP_address: 0000 Size: 64 SD/SC:
3_MCB Address:0176 PSP_address: 0040 Size: 256 SD/SC:
4_MCB Address:0187 PSP_address: 0192 Size: 144 SD/SC:
5_MCB Address:0191 PSP_address: 0192 Size: 880 SD/SC: MODULE
6_MCB Address:01C9 PSP_address: 01D4 Size: 144 SD/SC:
7_MCB Address:01D3 PSP_address: 01D4 Size:647856 SD/SC: VAR1
```

Рис. 1 – Установка обработчика прерывания и проверка размещения в памяти

Шаг 3. При повторном запуске программы обработчик прерывания уже установлен, поэтому выводится соответствующее сообщение. Поведение программы в этом случае верное. Результат запуска см. на рис. 2.

```
C:\>module
Interrupt is set.

C:\>module
Interrupt has already set.
```

Рис. 2 – Повторный запуск module.exe

Шаг 4. Запустим module.exe с параметром /un, чтобы выгрузить обработчик прерывания и проверим его размещение в памяти. Для этого запустим модуль прошлой лабораторной работы var1.com. Результат представлен на рисунке 3. Видим, что обработчик прерывания был выгружен и удален из памяти (не осталось блоков памяти, используемых пользовательским обработчиком прерывания).

```
C:\>var1
Available memory: 648912 bytes
Extended memory: 15728640 bytes
1_MCB Address:016F PSP_address: 0008 Size: 16 SD/SC:
2_MCB Address:0171 PSP_address: 0000 Size: 64 SD/SC:
3_MCB Address:0176 PSP_address: 0040 Size: 256 SD/SC:
4_MCB Address:0187 PSP_address: 0192 Size: 144 SD/SC:
5_MCB Address:0191 PSP_address: 0192 Size:648912 SD/SC: VAR1
```

Рис. 3 – Выгрузка обработчика прерывания и освобождение памяти из-под него

По результатам запусков программа работает корректно во всех случаях.

### Ответы на вопросы.

1. Как реализован механизм прерывания от часов?

Механизм прерывания от часов реализован так: примерно 18 раз в секунду по каждому тикку аппаратных часов вызывается прерывание 1Ch, которое первоначально указывает на единственную команду IRET. Но можно заменить обработчик данного прерывания на пользовательский. При вызове прерывания текущее CS:IP сохранится в стеке вместе с регистром флагов, в CS:IP загрузится адрес точки входа программы обработки прерывания,

процессор переключится на выполнение кода обработчика, а затем вернется на выполнение прерванной программы, используя CS:IP из стека.

## 2. Какого типа прерывания использовались в работе?

В данной работе использовались аппаратное прерывание - 1Ch и программные прерывания – int 10h (для вывода информации на экран) и int 21h (для работы с DOS). Обработчик прерываний устанавливался пользовательский.

## **Выводы.**

В результате работы был реализован пользовательский обработчик прерываний сигналов таймера, который выводит информацию о количестве вызовов на экран, а также реализована установка и выгрузка этого обработчика.