

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

ОТЧЕТ по практической работе №4
по дисциплине «Операционные системы»
Тема: Обработка стандартных прерываний

Студент гр. 8382

Торосян Т.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт 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 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.

Необходимые сведения для составления программы.

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 начинает 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
MOV AL, 1CH           ; номер 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 программе.

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
```

Постановка задачи.

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

Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.

Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход по функции 4Ch прерывания int 21h.

Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

Выгрузка прерывания о соответствующему значению параметра в командной строке /up. Выгрузка прерывания состоит в восстановлении up. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Шаг 2. Далее необходимо запустить отлаженную программу и убедиться, что резидентный обработчик прерывания 1Ch установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого нужно запустить программу ЛР 3, которая отображает карту памяти в виде списка блоков МСВ.

Шаг 3. Затем необходимо запустить отлаженную программу еще раз и убедиться, что программа определяет установленный обработчик прерываний.

Шаг 4. Далее нужно запустить отлаженную программу с ключом выгрузки и убедиться, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3.

Процедуры используемые в программе.

Таблица 1 - Процедуры используемые в программе

Название процедуры	Назначение
ROUT	Функция обработчика прерывания
IS_INT_L	Проверка установки резидента
IS_FLAG_UN	Проверка команды '/un'
INT_LOAD	Загрузка резидента
INT_UNLOAD	Выгрузка резидента
PRINT	Вывод на экран

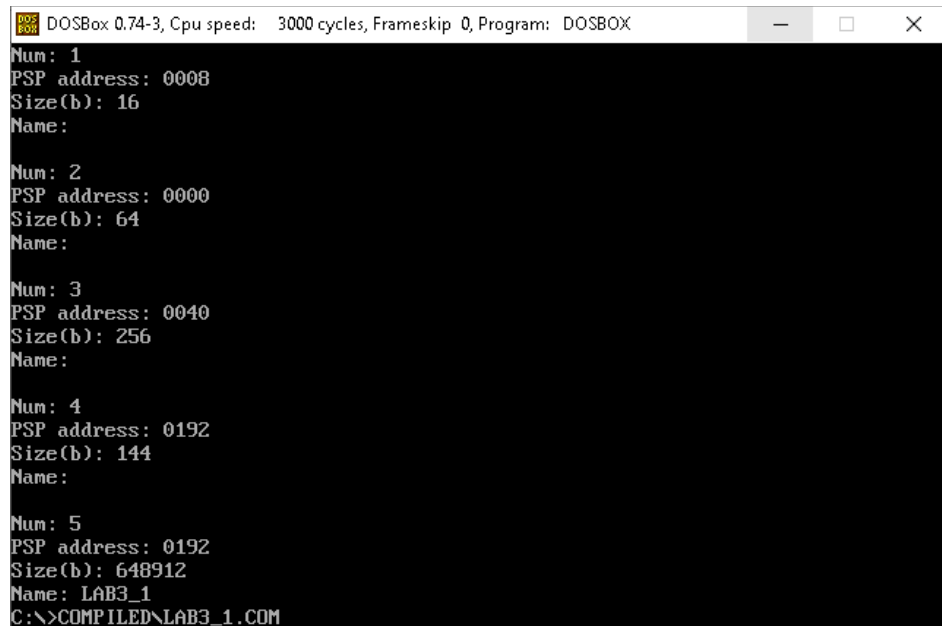
Структуры данных.

Таблица 2 - Структуры данных используемые в программе

Название поля данных	Тип	Назначение
NOT_LOADED	db	Резидент не загружен
LOADED	db	Резидент уже загружен
LOAD	db	Резидент загружен
UNLOAD	db	Резидент был выгружен

Результат работы.

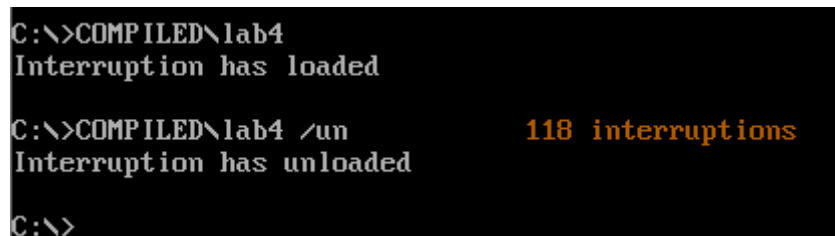
1) Состояние памяти до загрузки резидента (используем модуль, разработанный в третьей лабораторной работе). (Рис. 1)



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Num: 1
PSP address: 0008
Size(b): 16
Name:
Num: 2
PSP address: 0000
Size(b): 64
Name:
Num: 3
PSP address: 0040
Size(b): 256
Name:
Num: 4
PSP address: 0192
Size(b): 144
Name:
Num: 5
PSP address: 0192
Size(b): 648912
Name: LAB3_1
C:\>COMPILED\LAB3_1.COM
```

Рисунок 1 – Память до загрузки резидента

2) Загрузка и выгрузка резидента в/из памяти. (Рис. 2)



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>COMPILED\lab4
Interruption has loaded
C:\>COMPILED\lab4 /un
Interruption has unloaded
118 interruptions
C:\>
```

Рисунок 2– Загрузки резидента

3) Проверки состояний резидента в памяти. (Рис. 3)

```

C:\>COMPILED\lab4
Interruption has loaded

C:\>COMPILED\lab4                                051 interruptions
Interruption loaded already

C:\>COMPILED\lab4 /un                              125 interruptions
Interruption has unloaded

C:\>COMPILED\lab4 /un
Interruption isn't loaded

C:\>

```

Рисунок 3 – Проверки состояний резидента

4) Состояние памяти при загрузке в неё резидента. (Рис. 4)

```

1
2 Available size: 643952
3 Extended size: 15360
4
5 Num: 1
6 PSP address: 0008
7 Size(b): 16
8 Name: NUL NUL NUL NUL NUL NUL NUL NUL
9
10 Num: 2
11 PSP address: 0000
12 Size(b): 64
13 Name: NUL NUL NUL NUL NUL NUL NUL NUL
14
15 Num: 3
16 PSP address: 0040
17 Size(b): 256
18 Name: NUL NUL NUL NUL NUL NUL NUL NUL
19
20 Num: 4
21 PSP address: 0192
22 Size(b): 144
23 Name: NUL NUL NUL NUL NUL NUL NUL NUL
24
25 Num: 5
26 PSP address: 0192
27 Size(b): 4784
28 Name: LAB4 NUL NUL NUL NUL
29
30 Num: 6
31 PSP address: 02C8
32 Size(b): 144
33 Name: NUL NUL NUL NUL NUL NUL NUL NUL
34
35 Num: 7
36 PSP address: 02C8
37 Size(b): 643952
38 Name: LAB3_1 NUL NUL

```

Рисунок 4 – Состояние памяти при загрузке резидента

5) Запускаем отложенную программу с ключом /un, тем самым

выгружаем резидент и смотрим состояние памяти после выгрузки резидента. (Рис. 5)

```
1
2 Available size: 648912
3 Extended size: 15360
4
5 Num: 1
6 PSP address: 0008
7 Size(b): 16
8 Name: NUL NUL NUL NUL NUL NUL NUL NUL
9
10 Num: 2
11 PSP address: 0000
12 Size(b): 64
13 Name: NUL NUL NUL NUL NUL NUL NUL NUL
14
15 Num: 3
16 PSP address: 0040
17 Size(b): 256
18 Name: NUL NUL NUL NUL NUL NUL NUL NUL
19
20 Num: 4
21 PSP address: 0192
22 Size(b): 144
23 Name: NUL NUL NUL NUL NUL NUL NUL NUL
24
25 Num: 5
26 PSP address: 0192
27 Size(b): 648912
28 Name: LAB3_1 NUL NUL
```

Рисунок 5 – Запуск готовой программы

Ответы на контрольные вопросы.

1) Как реализован механизм прерывания от часов?

Сначала сохраняется содержимое регистров, потом определяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерывания, сохраняется в CS:IP) запоминается в стеке вместе с регистром флагов. Затем в, передаётся управление по адресу CS:IP) запоминается в стеке вместе с регистром флагов. Затем в и происходит выполнение обработчика, и в конце происходит возврат управления прерванной программе. Аппаратное прерывание от таймера происходит каждые 55 мс.

2) Какого типа прерывания использовались в работе?

Аппаратные прерывания(08h), прерывания функций DOS(21h), прерывания функций BIOS(10h, 1Ch).

Вывод.

В ходе лабораторной работы был построен обработчик прерывания от сигналов таймера. Изучены дополнительные функции работы с памятью: установка программы-резидента и его выгрузка из памяти.