

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

**ОТЧЕТ**  
**по лабораторной работе №5**  
**по дисциплине «Организация ЭВМ и систем»**  
**ТЕМА: РАЗРАБОТКА СОБСТВЕННОГО ПРЕРЫВАНИЯ.**

Студент гр. 9383

\_\_\_\_\_

Орлов Д.С.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Изучить принципы работы с прерываниями. Применить на практике полученные знания о прерываниях.

### **Краткие сведения.**

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т.д.). Когда возникает прерывание, процессор прекращает выполнение текущей программы (если ее приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата(CS:IP) - места, с которого будет продолжена прерванная программа. Затем в CS:IP загружается адрес программы обработки прерывания и ей передается управление. Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP, во вторых - CS) и хранятся в младших 1024 байтах памяти. Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

Программа обработки прерывания - это отдельная процедура, имеющая структуру:

```
SUBR_INT PROC FAR
PUSH AX ; сохранение изменяемых регистров
...
<действия по обработке прерывания>
POP AX ; восстановление регистров
...
MOV AL, 20H
OUT 20H,AL
IRET
SUBR_INT ENDP
```

Две последние строки обработчика прерывания, указанные перед командой IRET выхода из прерывания, необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

Замечание: в лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка цикла задержки в вывод сообщения или включение звукового сигнала.

Программа, использующая новые программы обработки прерываний при своем завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В соответствии с этим, программа должна содержать следующие инструкции:

; -- в сегменте данных

KEEP\_CS DW 0 ; для хранения сегмента

KEEP\_IP DW 0 ; и смещения вектора прерывания

; -- в начале программы

MOV AH, 35H ; функция получения вектора

MOV AL, 1CH ; номер вектора

INT 21H

MOV KEEP\_IP, BX ; запоминание смещения

MOV KEEP\_CS, ES ; и сегмента вектора прерывания

Для установки адреса нового обработчика прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая помещает заданные адреса сегмента и смещения обработчика в вектор прерывания с заданным номером.

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, 60H ; номер вектора

INT 21H ; меняем прерывание

POP DS

Далее может выполняться вызов нового обработчика прерывания.

В конце программы восстанавливается старый вектор прерывания

CLI

PUSH DS

MOV DX, KEEP\_IP

MOV AX, KEEP\_CS

MOV DS, AX

MOV AH, 25H

MOV AL, 1CH

INT 21H ; восстанавливаем старый вектор прерывания

POP DS

STI

Для работы с портами существуют специальные команды IN и OUT. IN вводит значение из порта ввода-вывода, OUT выводит.

43h - запись управляющего слова в регистр режима канала

42h - загрузка счетчика канала 2, чтение счетчика канала 2

Канал 2 - генератор звука

Инструкция loop уменьшает значение в регистре CX в реальном режиме или ECX в защищённом. Если после этого значение CX не ноль, то команда loop прыгает на метку.

Команды CLI и STI служат для установки или сброса флага прерываний, что позволяет включать или отключать реакцию на внешние прерывания. Команда CLI(Clear Interrupt flag) сбрасывает флаг IF в значение 0, что запрещает прерывания. Команда STI (Set -//-) устанавливает флаг IF в значение 1, что разрешает прерывания.

### **Текст задания.**

Вариант 1В.

Разработать собственное прерывание.

1 - 1Ch - прерывание от часов - генерируется автоматически операционной системой 18 раз в сек;

В - Выдача звукового сигнала;

### **Ход работы.**

По заданию нужно реализовать своё прерывание, генерирующее звуковой сигнал 18 раз в секунду. Прерывание основано на прерывании от часов 1Ch.

Для реализации данного задания в сегменте данных создаются переменные типа DW для хранения сегмента и смещения прерывания (KEEP\_CS, KEEP\_IP).

В сегменте кода реализована процедура обработки прерывания SUBR\_INT. Данная процедура выдает звуковой сигнал.

Для реализации смены прерывания, была использована функция 35h прерывания 21h. Был получен нужный вектор прерывания 1Ch. После запоминаем смещение и смещение сегмента вектора прерывания в переменные KEEP\_CS и KEEP\_IP. Далее добавляем с помощью функции 25h прерывания 21h нужное нам прерывание 1Ch. Восстанавливаем вектор прерывания.

Исходный код программы представлен в приложении А.

### **Выводы.**

Были изучены основные принципы работы с прерываниями. было разработано собственное прерывание.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

#### **main.asm**

AStack SEGMENT STACK

DB 1024 DUP(?)

AStack ENDS

DATA SEGMENT

KEEP\_CS DW 0 ;хранение  
сегмента

KEEP\_IP DW 0 ;хранение  
смещения прерывания

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,  
DS:DATA, SS:AStack

SUBR\_INT PROC FAR

jmp procedure

KEEP\_SS DW 0

KEEP\_SP DW 0

KEEP\_AX DW 0

MY\_STACK DW 1024

DUP(?)

procedure:

MOV KEEP\_SP, SP

MOV KEEP\_AX, AX

MOV AX, SS

MOV KEEP\_SS, AX

MOV AX, KEEP\_AX

MOV SP, OFFSET

procedure

MOV AX, seg

MY\_STACK

MOV SS, AX

push ax ;сохраняем все  
изменяемые регистры

push dx

; подача звукового сигнала

Sound:

; Управление высотой  
звука динамика

MOV AL, 10110110b ;  
Канал 2, режим 2, оба  
байта

OUT 43h, AL ;  
Установка режима для 2-  
го канала

MOV AX, 300 ;  
Выбор высоты звука

OUT 42h, AL ;  
Включить таймер,  
который

; будет

выдавать импульсы на  
; динамик  
с заданной частотой

MOV AL, AH

OUT 42h, AL ;  
Занесение высоты звука в  
порт динамика

IN AL, 61h ;  
Получить состояние  
динамика

MOV AH, AL ; И  
сохранить его в AH

OR AL, 00000011b ;  
Установить два младших  
бита

OUT 61h, AL ;  
Включить динамик

mov cx, 0Fh ;задержка

mov dx, 4240h ;для  
корректного

mov ah, 86h ;вывода  
звукового

int 15h ;сигнала

; выключаем звук

Sound\_OFF:

loop Sound\_OFF

mov al, 61h

out 61h, al

pop dx ;  
восстанавливаем регистры



pop ax

MOV KEEP\_AX, AX

MOV SP, KEEP\_SP

MOV AX, KEEP\_SS

MOV SS, AX

MOV AX, KEEP\_AX

mov al, 20h ; разрешение  
обработки прерываний

out 20h, al ; более низкого  
уровня

iret ; конец прерывания

SUBR\_INT ENDP

MAIN PROC FAR

mov ax, DATA

mov ds, ax

mov ah, 35h ; ф-ия  
получения вектора

mov al, 1Ch ; номер  
прерывания

int 21h

mov KEEP\_IP, bx ;  
запомнили смещение

mov KEEP\_CS, es ;  
запомнили сегмент  
вектора прерывания

```
push ds          ; сохранили  
ds
```

```
mov ax, seg SUBR_INT ;  
    сегмент процедуры в ax
```

```
mov dx, offset SUBR_INT ;  
    смещение процедуры
```

```
mov ds, ax
```

```
mov ah, 25h      ;  
    функция установки  
    вектора
```

```
mov al, 1Ch      ; номер  
    вектора
```

```
int 21h          ; изменение  
    прерывания
```

```
pop ds           ;  
    восстанавливаем ds
```

```
BUTTON_LOOP:
```

```
    mov ah, 00h
```

```
    int 16h
```

```
    cmp al, 27
```

```
    je exit
```

```
    loop BUTTON_LOOP
```

```
exit:
```

```
CLI
```

```
push ds
```

```
mov dx, KEEP_IP ;  
    восстановили смещение  
    для прерывания
```

```
mov ax, KEEP_CS    ;  
    восстановили сегмент  
    прерывания
```

```
mov ds, ax
```

```
mov ah, 25h        ; функция  
    установки вектора
```

```
mov al, 1Ch        ; номер  
    нашего прерывания
```

```
int 21h            ; изменили  
    прерывания
```

```
pop ds
```

```
STI
```

```
mov ah, 4ch
```

```
int 21h
```

```
MAIN ENDP
```

```
CODE ENDS
```

```
END MAIN
```