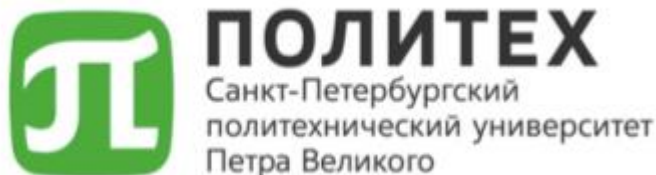


Федеральное государственное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и кибербезопасности  
Высшая школа программной инженерии



## КУРСОВАЯ РАБОТА

**Программирование на ассемблере**  
по дисциплине «Архитектура ЭВМ. Часть 1»

Выполнил  
студент гр. 5130904/30008

А. А. Золотухин

Руководитель  
доцент, к.т.н

А.В. Милицын

«26» декабря 2024 г.

Санкт-Петербург  
2024

## ОГЛАВЛЕНИЕ

Введение .....	3
Программа № 1 .....	4
Блок схема.....	4
Текст программы.....	5
Скриншоты .....	8
Программа № 2 .....	9
Блок схема.....	9
Список прерываний BIOS .....	10
Используемые устройства.....	10
Текст программы.....	11
Скриншоты .....	13

## ВВЕДЕНИЕ

Ассемблер – низкоуровневый машинно-ориентированный язык программирования. Реализация языка зависит от типа процессора и определяется архитектурой вычислительной системы. Ассемблер позволяет напрямую работать с аппаратурой компьютера. Программа на языке ассемблера включает в себя набор команд, которые после трансляции преобразуются в машинные команды.

Также программы, написанные на языке ассемблера, требуют значительно меньшего объема памяти и времени выполнения. Знание программистом языка ассемблера и машинного кода дает ему понимание архитектуры машины. Несмотря на то, что большинство специалистов в области программного обеспечения разрабатывают программы на языках высокого уровня, наиболее мощное и эффективное программное обеспечение полностью или частично написано на языке ассемблера.

Для разработки необходимых программ были использованы:

- Виртуальная машина Windows XP;
- ASMTTool для написания программ;
- Microsoft Visual Studio для написания программ;
- MASM в качестве компилятора.

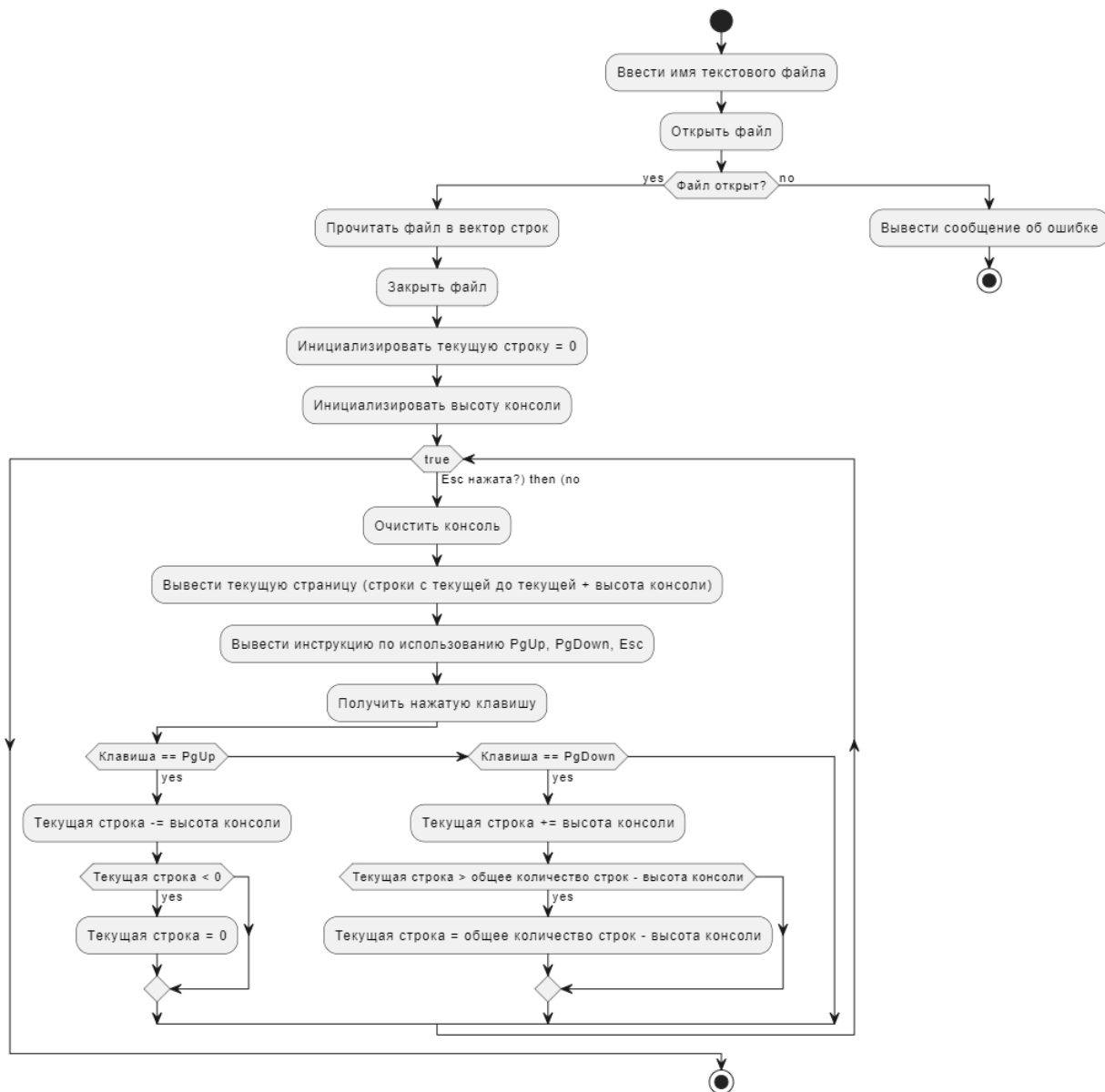
### Цели работы:

1. Организовать печатание произвольного текстового файла на экране. Вертикальный размер окна (количество строк) может меняться. Для печатания желательно использовать PgUP и PgDOWN.
2. Замените клавишу Del на CapsLock. Напишите программу, обработки прерывания клавиатуры, которая заменяет скан-коды клавиш. При каждом нажатии выдавайте звуковой сигнал с использованием порта 61h.

Программа № 1

# ПРОГРАММА № 1

## Блок схема



## Текст программы

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <conio.h>

#define PAGE_UP 73
#define PAGE_DOWN 81

using namespace std;

extern "C" void print_string(const char* str) {
    cout << str << endl;
}

int main() {
    setlocale(LC_ALL, "Russian");
    string filename;
    cout << "Введите имя текстового файла: ";
    cin >> filename;

    ifstream file(filename);
    if (!file.is_open()) {
        cerr << "Ошибка: не удалось открыть файл." << endl;
        return 1;
    }

    vector<string> lines;
    string line;
    while (getline(file, line)) {
        lines.push_back(line);
    }
```

## Программа № 1

```
file.close();

const int consoleHeight = 10;
int currentLine = 0;
size_t totalLines = lines.size();

while (true) {
    system("cls");

    for (int i = 0; i < consoleHeight && currentLine + i <
totalLines; ++i) {
        cout << lines[currentLine + i] << endl;
    }

    cout << "\nИспользуйте PgUp и PgDown для прокрутки, Esc
для выхода.";

    int key = _getch();
    if (key == 27)
        break;
    else if (key == 224) {
        key = _getch();
        __asm {
            mov eax, key
            cmp eax, PAGE_UP
            je page_up
            cmp eax, PAGE_DOWN
            je page_down
            jmp end_key_check

        page_up :
            mov eax, currentLine
            sub eax, consoleHeight
            cmp eax, 0
```

## Программа № 1

```
    jl reset_to_zero
    mov currentLine, eax
    jmp end_key_check

page_down :
    mov eax, currentLine
    add eax, consoleHeight
    mov ebx, totalLines
    sub ebx, consoleHeight
    cmp eax, ebx
    jg set_to_max
    mov currentLine, eax
    jmp end_key_check

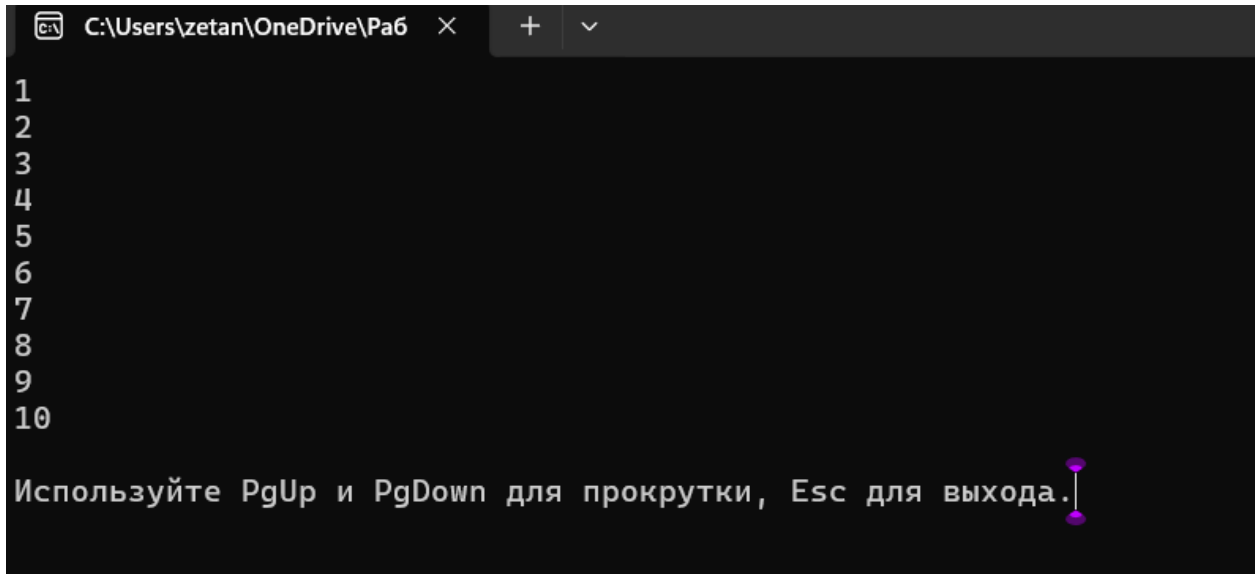
reset_to_zero :
    mov currentLine, 0
    jmp end_key_check

set_to_max :
    mov currentLine, ebx

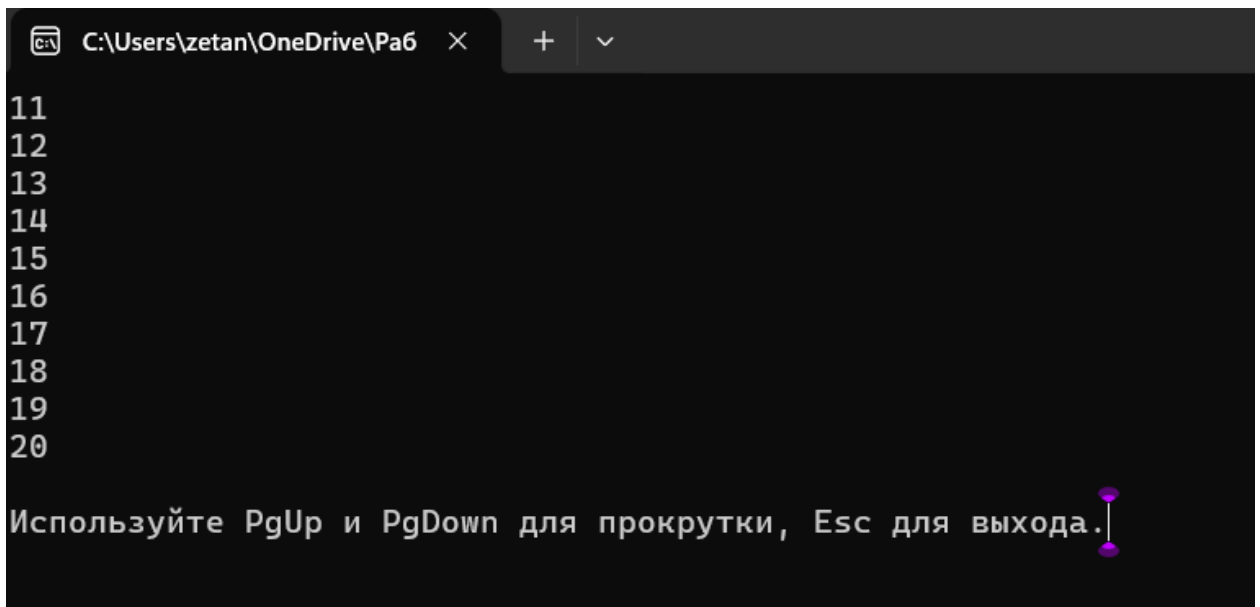
end_key_check :
    nop
}
}
}

return 0;
}
```

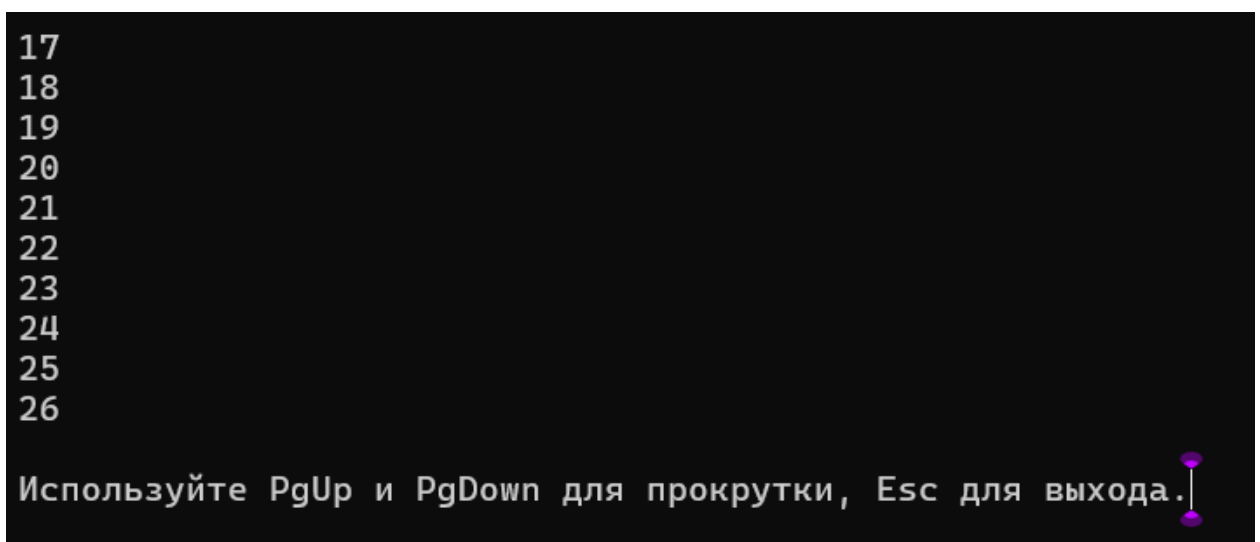
## Скриншоты



A screenshot of a terminal window with a dark background. The title bar shows the file path "C:\Users\zetan\OneDrive\Раб" and standard window controls. The terminal displays a list of numbers from 1 to 10 on the left side. At the bottom, there is a text prompt: "Используйте PgUp и PgDown для прокрутки, Esc для выхода." followed by a vertical scroll bar with purple handles.



A screenshot of a terminal window, similar to the first one. The terminal displays a list of numbers from 11 to 20 on the left side. At the bottom, there is a text prompt: "Используйте PgUp и PgDown для прокрутки, Esc для выхода." followed by a vertical scroll bar with purple handles.



A screenshot of a terminal window, similar to the previous ones. The terminal displays a list of numbers from 17 to 26 on the left side. At the bottom, there is a text prompt: "Используйте PgUp и PgDown для прокрутки, Esc для выхода." followed by a vertical scroll bar with purple handles.



Программа № 2

## ПРОГРАММА № 2

### Блок схема



## Список прерываний BIOS

№ Пр-ия	№ Ф-ии	Назначение	Параметры
21h	35h	Сохранение вектора прерывания	AL – номер прерывания.
	25h	Установка вектора прерывания	AL – номер прерывания, DS:DX – адрес обработчика прерываний.
	06h	Вывод на экран	DL – символ для вывода.
09h	-	Прерывание клавиатуры	Нет.
60h	-	Здесь будет храниться стандартное прерывание клавиатуры	Нет.

## Используемые устройства

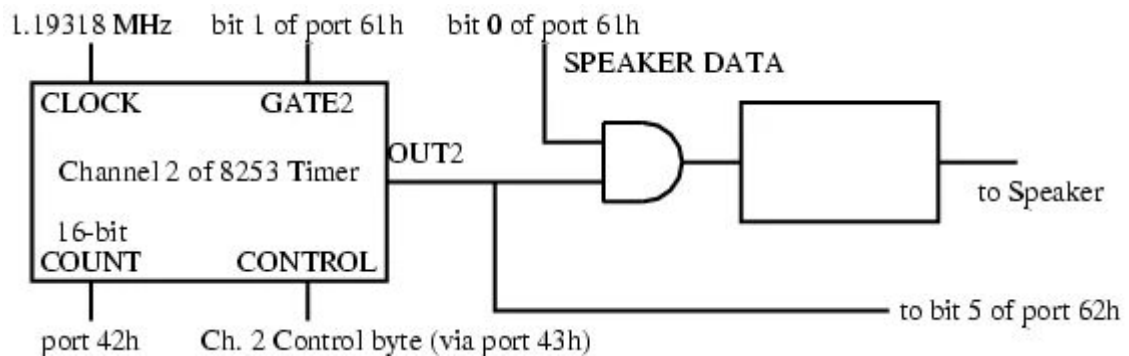


Рис. 1 Схема работы спикера

## Описание 43h порта (Programmable Interval Timer)

Биты	Использование
6 и 7	Выбор канала: <ul style="list-style-type: none"> <li>00 = Канал 0</li> <li>01 = Канал 1</li> <li>10 = Канал 2</li> <li>11 = Зарезервировано</li> </ul>
4 и 5	Уровень доступа: <ul style="list-style-type: none"> <li>00 = Латч счетчика</li> <li>01 = Запись только младшего байта</li> <li>10 = Запись только старшего байта</li> <li>11 = Запись сначала младшего, затем старшего байтов</li> </ul>
1 – 3	Режим: <ul style="list-style-type: none"> <li>000 = Однократный счетчик</li> <li>001 = Однократный счетчик с повторным запуском</li> <li>010 = Периодическая генерация импульсов</li> <li>011 = Генерация прямоугольных импульсов</li> <li>100 = Программно-триггерный режим</li> <li>110 и 111 = Зарезервировано</li> </ul>
0	Форма счетчика: <ul style="list-style-type: none"> <li>0 = Двоичный формат (16 бит)</li> </ul>

	• 1 = Десятичный формат BCD (4-разрядные цифры)
--	---

Исходя из схемы, для генерации звука нам надо подключить PIT к динамику, для этого нам надо установить биты 6 и 7 на 1 и 0 соответственно (канал 2). Это делается через байт, управления который отправим на порт 43h.

### Описание 61h порта (System Speaker)

Бит 0 порта 61h отвечает за выбор источника для динамика (0 = manual, 1 = PIT). Бит 1 порта 61h отвечает за включение/отключение динамика. Бит 7 – отвечает за блокировку и разблокировку клавиатуры.

В дальнейшем мы будем работать с этими битами для переключения соответствующих параметров.

### Текст программы

```

MODEL TINY
STACK 256
DATA SEG
CODE SEG
start:
    mov ah, 35h
    mov al, 9
    int 21h
    cli
    mov ax, es
    mov ds, ax
    mov dx, bx
    mov ah, 25h
    mov al, 60h
    int 21h

    mov ax, seg cmp_cl
    mov ds, ax
    mov dx, offset cmp_cl
    mov ah, 25h
    mov al, 09h
    int 21h
    sti

infc:
    mov ah, 06h
    mov dl, 0FFh
    int 21h
    jz infc
    mov dl, al
    int 21h
    jmp infc

cmp_cl:
    in al, 60h
    cmp al, 83
    je DelPress
    cmp al, 211
    je DelUnpr
    int 60h
    jmp CLEnd

```

```

DelPress:
    in al,61h
    or al,10000011b
    out 61h,al
    and al,01111111b
    out 61h,al

    mov al,10100110b
    out 43h,al
    mov al,22
    out 42h,al
    push es
    mov ax,40h
    mov es,ax
    mov al,es:[17h]
    xor al,40h
    mov es:[17h],al
    mov al,es:[18h]
    or al,40h
    mov es:[18h],al
    pop es
    mov al,20h
    out 20h,al
    jmp CLEnd

DelUnpr:
    in al,61h
    and al,11111100b
    out 61h,al
    in al,61h
    or al,10000000b
    out 61h,al
    and al,01111111b
    out 61h,al
    push es
    mov ax,40h
    mov es,ax
    mov al,es:[18h]
    and al,191
    mov es:[18h],al
    pop es
    mov al,20h
    out 20h,al
CLEnd:
    iret
end start

```

## Скриншоты

