Санкт-Петербургский государственный политехнический университет Институт компьютерных наук и технологий  
**Кафедра «Институт компьютерных наук и кибербезопасности»**

**КУРСОВАЯ РАБОТА**

Программирование на ассемблере

По дисциплине «Архитектура ЭВМ. Часть 1»

Выполнил студент гр. 5130904/30007 Голиков В.С

Руководитель

проф. д.т.н. С.А. Молодяков

Санкт-Петербург

2024

**Оглавление**

[Введение 3](#_Toc184990999)

[Программа 1 4](#_Toc184991000)

[Блок-схема 4](#_Toc184991001)

[Список использованных прерываний BIOS 5](#_Toc184991002)

[Текст программы 6](#_Toc184991003)

[Скриншоты 11](#_Toc184991004)

[Программа 2 12](#_Toc184991005)

[Блок-схема 12](#_Toc184991006)

[Список использованных прерываний BIOS 13](#_Toc184991007)

[Текст программы 14](#_Toc184991008)

[Скриншоты 18](#_Toc184991009)

# 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. Программа преобразуется в исполняемый машинный код с помощью ассемблера — программы-транслятора, которая и дала название этому языку. Команды ассемблера соответствуют командам процессора один к одному, представляя собой удобные для чтения символьные формы, известные как мнемокоды. При этом одной команде ассемблера может соответствовать несколько различных команд процессора.

**Программа 1**

В файле хранится каталог файлов в формате команды DIR. Разработать программу для вывода:

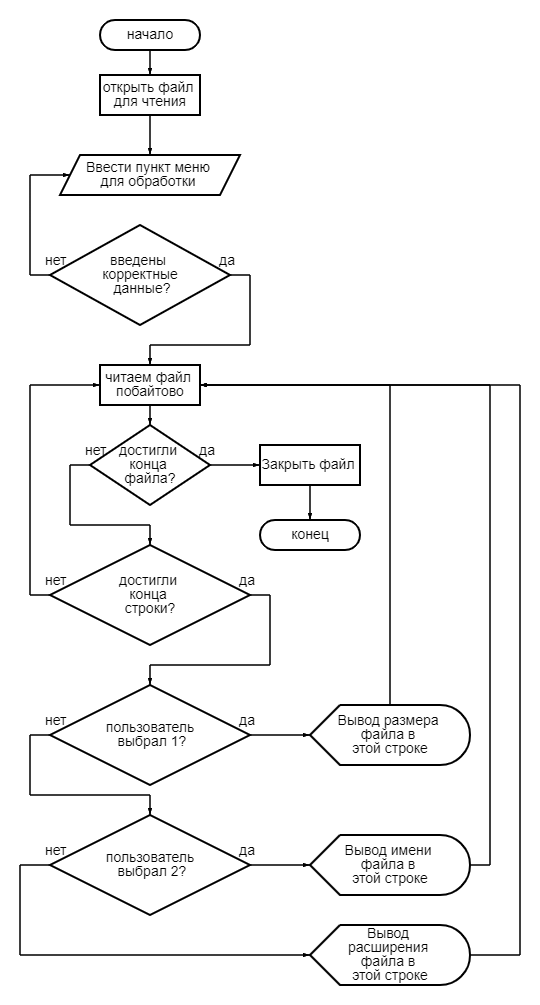
- имени всех файлов  
 - размера всех файлов  
 - расширений всех файлов

**Программа 2**

Программа "Будильник4". Задайте с клавиатуры время относительно текущего времени (например, 2 минуты). Перехватите прерывание клавиатуры (номер 9) и при повторном наборе (нажатии клавиш) ваша программа не реагирует на ввод (блокировка клавиатуры). При наступлении заданного времени выдайте звуковой сигнал

# Программа 1

## Блок-схема



## Список использованных прерываний BIOS

**INT 21h – Main DOS API**

|  |  |
| --- | --- |
| **AH** | **Description** |
| 01h | Read character |
| 02h | Character output |
| 09h | Display string |
| 3Dh | Open file |
| 3Eh | Close file |
| 3Fh | Read file or device |
| 42h | Move file pointer |
| 4Ch | Terminate with return code 0 |

## Текст программы

.model large

.386

.stack 100h

*;---------------BLOCK MACRO-----------------------*

writeChar MACRO chr *;wtites one symbol CHR on the screen*

mov dl, chr

mov ah, 02h

int 21h

ENDM

*;-------------------------------------------------------------------------------------*

newline MACRO *;entering new line*

writeChar 10 *;newline char*

writeChar 13 *;return char*

ENDM

*;-------------------------------------------------------------------------------------*

writeString MACRO string *;writes string on the screen*

mov dl, offset string

mov ah, 09h

int 21h

ENDM

*;-------------------------------------------------------------------------------------*

writeStringDispName MACRO string, disp *;writes string on the screen with displacement NAME*

local met1

mov di, disp

met1:

writeChar string[di]

inc di

cmp string[di], 46

jne met1

ENDM

*;-------------------------------------------------------------------------------------*

writeStringDispExt MACRO string, disp *;writes string on the screen with displacement EXTENSION*

local met1, met2

mov di, disp

met1:

cmp string[di], 46

je met2

inc di

jmp met1

met2:

writeChar string[di+1]

inc di

cmp string[di], 13

jne met2

ENDM

*;---------------------------------------------------------------------*

writeStringDisp MACRO string, disp *;writes string on the screen with displacement SIZE*

local met1, met2

mov di, disp

met1:

cmp string[di], 32

jne met2

inc di

jmp met1

met2:

writeChar string[di]

inc di

cmp string[di], 32

jne met2

ENDM

*;---------------------------------------------------------------------*

openFile MACRO data, filename

mov DS,data *;open file*

mov ah,3Dh *;function DOS 3Dh for opening file*

xor al,al *;mode - reading only*

mov dx,offset filename *;name of file*

xor cx,cx *;clear file. No atributes*

int 21h *;interrupt*

jc exit *;if flag C is up - error of opening*

mov bx,ax

xor cx,cx

xor dx,dx

mov ax,4200h

int 21h *;start reading from the first symbol*

ENDM

*;---------------------------------------------------------------------*

addEmptyLines MACRO *;writing 5 empty lines*

LOCAL L1

mov cx, 5

L1:

newline

loop L1

ENDM

*;---------------------------------------------------------------------*

writeMenu MACRO

writeString msgMenu1

writeString msgMenu2

writeString msgMenu3

writeString msgMenu4

writeString msgMenu5

ENDM

*;---------------------------------------------------------------------*

switchCase MACRO

switch: *; reading char from console*

mov ah, 01h *; DOS func for reading sybols*

int 21h

*; Check for right data (1, 2, 3)*

cmp al, '1'

je set\_value

cmp al, '2'

je set\_value

cmp al, '3'

je set\_value

cmp al, '0'

je close

newline

mov dx, offset msgIncorrectInput

mov ah, 09h

int 21h

jmp switch

set\_value:

push ax

newline

writeString msgResult

newline

ENDM

*;---------------END BLOCK MACROS-----------------*

*;---------------------------------------------------------------------*

*;---------------BLOCK OF DATA--------------------------*

.data

fString db 100 dup(?) *;string of data from file*

*;messages for user*

CrLf db 0Dh, 0Ah, '$'

msgSourceData db 'Original Data:', 0Dh, 0Ah, '$'

msgIncorrectInput db 'Incorrect input, try again', 0Dh, 0Ah, '$'

msgResult db 'RESULT: $'

msgMenu1 db 'Enter your choice:', 0Dh, 0Ah, '$'

msgMenu2 db ' 1. Get size', 0Dh, 0Ah, '$'

msgMenu3 db ' 2. Get name', 0Dh, 0Ah, '$'

msgMenu4 db ' 3. Get ext', 0Dh, 0Ah, '$'

msgMenu5 db ' 0. Exit', 0Dh, 0Ah, '$'

*; name of file we'll analise*

path db "file.txt", 0

buf db ?

*;---------------END BLOCK OF DATA-----------------*

*;-------------------------------------------------------------------------------------*

*;---------------BLOCK OF CODE--------------------------*

.code

start:

mov ax,@data

openFile ax, path

writeMenu

switchCase

analise\_file:

lea di,fString

read\_byte:

mov ah,3fh *; start reading file*

mov cx,1 *; reading by 1 byte*

lea dx,buf

int 21h

inc dx

cmp ax,cx *; waiting for EoF or error*

jnz close *; close file*

mov dl,buf[0]

cmp dl, 10

je get\_data

mov [di],dl

inc si

inc di

jmp read\_byte

get\_data:

pop ax

cmp al, '1'

je get\_size

cmp al, '2'

je get\_name

cmp al, '3'

je get\_ext

get\_size: *;get size of file*

push ax

cmp fString[0], '0'

jb analise\_file

cmp fString[0], '9'

ja analise\_file

writeStringDisp fString, 30

newline

jmp analise\_file

get\_name: *;get name of file*

push ax

cmp fString[0], '0'

jb analise\_file

cmp fString[0], '9'

ja analise\_file

writeStringDispName fString, 38

newline

jmp analise\_file

get\_ext: *;get extension of file*

push ax

cmp fString[0], '0'

jb analise\_file

cmp fString[0], '9'

ja analise\_file

writeStringDispExt fString, 40

newline

jmp analise\_file

*;------ENDING THE PROGRAM--------------------------------*

close: *; closing file after work*

mov ah,3eh

int 21h

exit: *; ending*

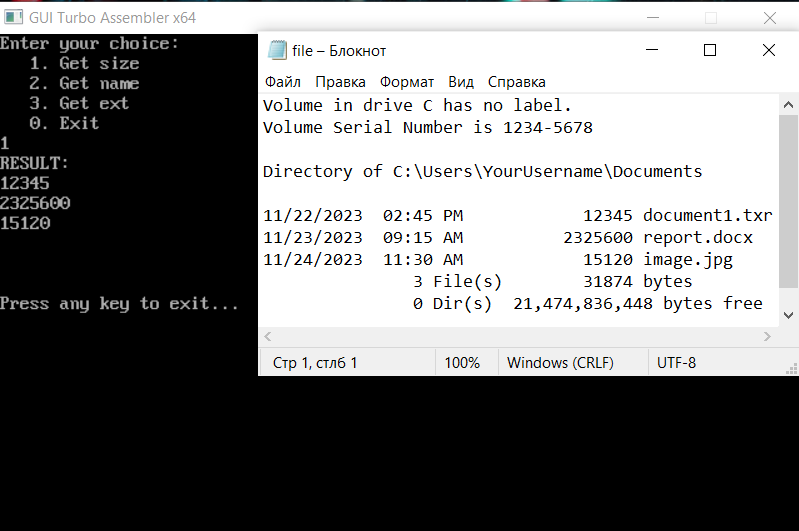
mov ah,4ch

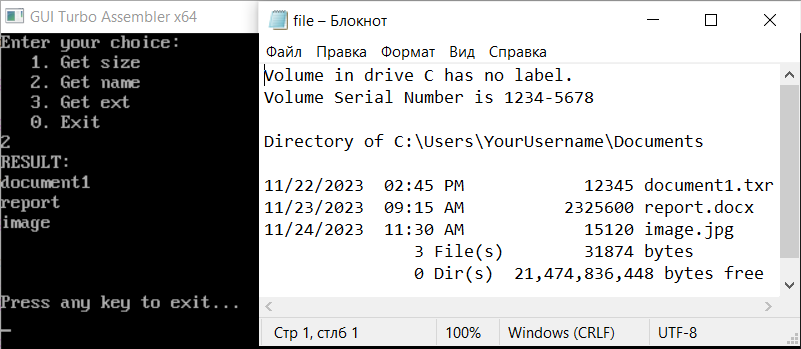
int 21h

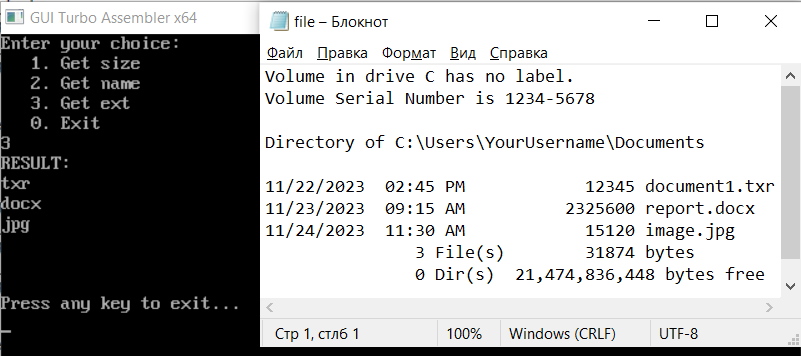
end start

*;-------------END CODE------------------------------*

## Скриншоты

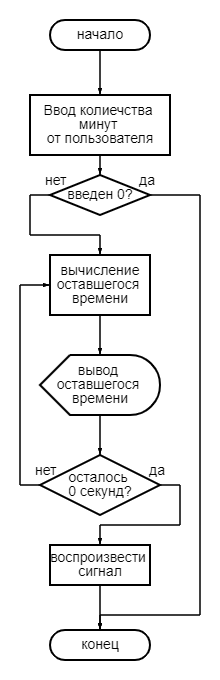






# Программа 2

## Блок-схема



## Список использованных прерываний BIOS

**INT 21h – Main DOS API**

|  |  |
| --- | --- |
| **AH** | **Description** |
| 01h | Read character |
| 02h | Character output |
| 09h | Display string |
| 0Ah | Buffered keyboard input |

**INT 1Ah – BIOS keyboard services**

|  |  |
| --- | --- |
| **AH** | **Description** |
| 00h | Get real time (system) |

**INT 10h – BIOS keyboard services**

|  |  |
| --- | --- |
| **AH** | **Description** |
| 0Eh | Sound signal |

## Текст программы

.386

.model small

.stack 100h

*;--------DATA BLOCK--------------------*

.data

message db 'Enter delay: $'

str db 5,7 dup (0)

max dw 0

min dw 0

num\_18 dw 18

num\_60 dw 60

*;---------END OF DATA------------------*

*;---------CODE BLOCK-------------------*

.code

start:

mov ax,@data

mov ds,ax

mov es,ax

mov ah,09h

lea dx,message

int 21h

mov ah,0ah

lea dx,str

int 21h

lea si,str[2]

call at\_to\_int

test ax,ax

jz exit

mul num\_18

mul num\_60

mov si,ax

mov di,dx

mov ah,0

int 1ah

add dx,si

adc cx,di

mov min,dx

mov max,cx

kol:

mov ah,0

int 1ah

mov si,min

mov di,max

sub si,dx

sbb di,cx

mov ax,si

mov dx,di

test ax,ax

jnz del

test dx,dx

jnz del

jmp snd

del:

div num\_18

cwd

div num\_60

push ax dx

mov ah,2

mov bh,0

mov bh,0

mov dh,0

mov dl,25

int 10h

pop dx ax

push ax dx

mov dl,'0'

mov ah,2

int 21h

pop dx ax

min1:

call ost\_min

push dx

mov ah,2

mov dl,'m'

int 21h

pop ax

cmp ax,10

jnc sec1

push ax

mov dl,'0'

mov ah,2

int 21h

pop ax

sec1:

call ost\_min

mov ah,2

mov dl,'s'

int 21h

jmp kol

snd:

call Sound1

exit:

mov ah,1

int 21h

mov ah,4ch

int 21h

*;---------PROCESSES------------------*

Sound1 proc near *;sound*

mov ah,0Eh

mov al,'E'

int 10h

call delay

Sound1 endp

delay proc *;delay for sound*

mov cx, 0FFFFh

delay\_loop:

loop delay\_loop

ret

delay endp

at\_to\_int proc

push dx

push si

xor dx,dx

search:

xor ax,ax

lodsb

cmp al,13

jz return

cmp al,'9'

jnbe search

cmp al,'0'

jb search

sub ax,'0'

shl dx,1

add ax, dx

shl dx, 2

add dx, ax

jmp search

return:

mov ax,dx

pop si

pop dx

ret

at\_to\_int endp

ost\_min proc *;minutes left*

push cx

push dx

push bx

mov bx,10

xor cx, cx

met1:

xor dx,dx

div bx

push dx

inc cx

test ax,ax

jnz met1

mov ah,2

met2:

pop dx

add dl,'0'

int 21h

loop met2

pop bx

pop dx

pop cx

ret

ost\_min endp

*;-----------END PROCESSES---------------------*

end start

*;-----------END CODE--------------------------*

End

## Скриншоты

