

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Автор |  | | |  | Ковалев Д. П. | | | |
|  | (подпись, дата) | | |  |  | | | |
| Обозначение | ИиВТ.10.05.01.13 | | Группа | | | | ВКБ32 |  |
| Направление подготовки | | 10.05.01 Компьютерная безопасность | | | | | |  |
| Профиль | Математические методы защиты информации | | | | | | |  |
| Проверил |  | | |  | |  | |  |
|  | (подпись, дата) | | |  | |  | |  |

Ростов-на-Дону

2024

Содержание

[Лабораторная работа №1 3](#_Toc184310959)

[Лабораторная работа №2 6](#_Toc184310960)

[Лабораторная работа №3 8](#_Toc184310961)

[Лабораторная работа №4 10](#_Toc184310962)

[Лабораторная работа №5 27](#_Toc184310963)

# **Лабораторная работа №1**

**Тема:** «Циклические и разветвляющиеся программы»

**Цель:** научиться работать с циклами и ветвлениями в asm.

**Задание 1**. Заменить все отрицательные числа нулями. Дан изначально массив байт.

Весь код лабораторной представлен ниже:

.model small

.stack 100h

.data

arr db 11010010b, 10010001b, 11111111b, 10101010b, 11010101b, 10000001b, 10111101b, 10001101b, 01010101b, 01001011b

ones\_count db 0

all\_bits\_count db 0

.code

mov ax, @data

mov ds, ax

mov si, 0 ; number of bytes

mov cx, 0 ; number of bits

mov bl, 3 ; multiple

xor ax, ax

byte\_loop:

mov dl, 10000000b

mov al, arr[si]

mov cx, 0

bit\_loop:

test al, dl

jnz check\_multiple

push ax

xor ax, ax

middle:

pop ax

inc all\_bits\_count

inc cx

cmp cx, 8

je next\_byte

shr dx, 1

jmp bit\_loop

next\_byte:

inc si

cmp si, 10

je end

jmp byte\_loop

check\_multiple:

push ax

xor ax, ax

mov al, all\_bits\_count

div bl ; AX / 3

cmp ah, 0

je plus\_one

jmp middle

plus\_one:

inc ones\_count

jmp middle

end:

mov al, ones\_count

mov ah, 4ch

int 21h

# **Лабораторная работа №2**

**Тема:** «Применение логических конструкций»

**Цель:** научиться работать с базовыми операторами логики в asm.

**Задание 1**. Дан массив из 10 байт. Посчитать количество единиц во всех разрядах, кратных трем: 3, 6, 9, …, 75, 78.

Весь код лабораторной представлен ниже:

segment DATA\_S

buffer db 11011101b, 11111000b

buffer\_s dw $-buffer

DATA\_S ends

segment CODE\_S

assume cs:CODE\_S, ds:DATA\_S

start:

mov ax, DATA\_S

mov ds, ax

mov cx, buffer\_s

mov si, offset buffer

xor dx, dx

enumerate\_buffer:

mov al, [si]

mov ah, 0

next\_bit:

inc ah

cmp ah, 9

jg continue

shr al, 1

jc try\_inc

skip\_increment\_counter:

jmp next\_bit

try\_inc:

cmp ah, 7

je increment\_counter

cmp ah, 4

je increment\_counter

jmp next\_bit

increment\_counter:

inc dx

jmp next\_bit

continue:

add si, 1

loop enumerate\_buffer

quit:

mov ax, 4c00h

int 21h

CODE\_S ends

end start

# **Лабораторная работа №3**

**Тема:** «Обработка символьной информации с помощью функций DOS»

**Цель:** научиться обрабатывать символьную информацию, используя asm.

**Задание 1.** Ввести с клавиатуры строку, состоящую из нескольких слов. Вывести каждое слово на экран в отдельной строке, т.е выдать слова в столбик.

Весь код лабораторной представлен ниже:

DATA\_S segment

buffer db 255 dup('$')

new db 10, 13, '$'

DATA\_S ends

CODE\_S segment

assume cs:CODE\_S, ds:DATA\_S

start:

mov ax, DATA\_S

mov ds, ax

mov ah, 0ah

lea dx, buffer

int 21h

lea si, buffer + 2

print\_new\_line:

mov ah, 09h

lea dx, new

int 21h

iterate\_buffer:

mov al, [si]

cmp al, '$'

je quit

inc si

cmp al, ' '

je print\_new\_line

mov ah, 02h

mov dl, al

int 21h

jmp iterate\_buffer

quit:

mov ax, 4c00h

int 21h

CODE\_S ends

end start

# **Лабораторная работа №4**

**Тема:** «Подпрограммы»

**Цель:** научиться писать подпрограммы

**Задание 1**. Разработать подпрограмму, которая определяет, содержится ли одна заданная строка в другой заданной строке и, если да, то, начиная с какой позиции.

Код представлен ниже:

data segment

prompt1 db 'Enter first string: $'

prompt2 db 13,10,'Enter substring: $'

crlf db 13,10,'$'

txtno db 13,10,'Not '

txtyes db 'exists $'

buf1 db 255

len1 db ?

str1 db 255 dup (?)

buf2 db 255

len2 db ?

str2 db 255 dup (?)

counter db ?

data ends

code segment

assume cs:code,ds:data,ss:stack

start:

mov ax,data

mov ds,ax

mov es,ax

mov ah,9 ; вывод строки на экран

lea dx,prompt1 ; первый промт для вывода на экран

int 21h ; системное прерывание для 1 считывания с консоли

mov ah,10 ; считвание строки с консоли

lea dx,buf1 ; сохраняем данные с консоли

int 21h ; вызываем в консоль с последующим сохранением в dx

mov ah,9 ; вывод строки на экран

lea dx,prompt2 ; второй промпт для вывода на экран

int 21h ; вызов консоли

mov ah,10 ; cчитывание строки

lea dx,buf2 ; переносим в buf2

int 21h ; системное прерывание для считывания 2 строки с консоли

mov cl,len1

sub cl,len2

jb No

inc cl

xor ch,ch

mov counter,ch

lea di,str1

m1:

push cx ; сохраняем значение в стеке

push di ; сохраняем значение в стеке

lea si,str2 ; загружается адрес второй строки в регистр SI

mov cl,len2 ; в CL устанавливается длина второй строки.

xor ch,ch ; обнуляем

repe cmpsb ; сравниваются байты у SI и DI

jne m2 ; если не равны, то идет на m2

inc byte ptr [counter] ; если равны, то увеличиваем значение счетчика

m2:

pop di ; восстановление значений из стека

inc di ;

pop cx

loop m1

cmp byte ptr [counter],0

jne Yes

No:

lea dx,txtno ; отдаем адрес в регистр

mov ah,9 ; вывод строки на экран

int 21h ; вызов консоли

jmp quit ; покидаем программу

Yes:

lea dx,crlf

mov ah,9

int 21h

lea dx,txtyes

int 21h

mov al,counter

xor ah,ah

mov bx,300Ah

div bl

add bh,ah

xor ah,ah

div bl

mov dx,3030h

add dx,ax

mov ah,2

int 21h

mov dl,dh

int 21h

mov dl,bh

int 21h

quit:

mov ah,8

int 21h

mov ax,4C00h

int 21h

code ends

stack segment

dw 256 dup (?) ; double word (2 байта), в x86 архитектуре тут 2 байта

stack ends

end start

**Задание 2.** Разработать программу, которая вводит с клавиатуры две строки и сообщает содержится ли одна в другой и сколько раз.

.186

assume cs:code, ds:data, ss:stk

; Сегмент команд программы

code segment para public use16 "code"

main proc

; Настроить сегментный регистр DS на сегмент данных

mov ax, data ; Установить регистр сегмента данных DS,

mov ds, ax ; указывающим на сегмент data

; \*\*\*\*\*\*\*\*\*\* Основная часть программы \*\*\*\*\*\*\*\*\*\*

; Ввести основную строку

mov si, offset Msg1

call PrnStr ; Распечатать приглашение ввода

mov si, offset MainSt

mov cx, StMaxLen

call InpStr ; Выполнить ввод

; Ввести подстроку

mov si, offset Msg2

call PrnStr ; Распечатать приглашение ввода

mov si, offset SubSt

mov cx, StMaxLen

call InpStr ; Выполнить ввод

; Выполнить поиск подстроки в строке

mov di, offset MainSt

mov si, offset SubSt

call StrPos

; Вывод результатов

cmp dl, 1

jne @21

; Подстрока найдена - в AX индекс вхождения

mov si, offset StPosStr

mov dh, 10 ; Вывести индекс AX в строку PosSt

call IntToStr ; в десятичной системе

mov si, offset Msg3

call PrnStr ; Распечатать результат - подстрока найдена

jmp @22

@21: ; Подстрока не найдена

mov si, offset Msg4

call PrnStr ; Распечатать результат - подстрока не найдена

@22: mov si, offset CRSt

call PrnStr ; Перевод строки

; \*\*\*\*\*\*\*\*\*\* Завершение работы программы \*\*\*\*\*\*\*\*\*\*

@31: mov ax, 4C00h ; Выход из

int 21h ; программы

main endp

; Процедура считывает строку с клавиатуры. После нажатия Enter-ввод

; прекращается. BackSpace - стирает символ слева от курсора.

;

; Вход: DS:SI - адрес начала памяти, куда нужно считывать строку

; CX - максимальная длина строки (без учета терминального нулевого символа)

; значение не может быть больше 254.

; Выход: Нуль-терминальная (оканчивающаяся нулем) строка по адресу DS:SI

;

InpStr proc

; Сохранить регистры

push ax

push cx

push dx

push bx

push si

push di

; Выполнить ввод строки во вспомогательный буфер

; с помощью функции DOS 0AH

push si

mov si, offset InpStrBuf

inc cl ; В функции 0AH максимальная длина указываается

; с учетом завершающего символа, так что

; добавить 1 к длине

mov byte ptr [si], cl ; В 0-м байте буфера нужно записать

; максимальную длину строки

mov byte ptr [si+1], 0

mov ah, 0Ah

mov dx, si

int 21h

; Преобразовать введенную строку в 0-терминальную

mov bx, 0

mov bl, [si+1] ; В 1-м байте буфера содержится фактическая

; длина строки

add si, 2 ; Фактические символы введенной строки

; начинаются со 2-го байта буфера

mov byte ptr bx[si], 0

; Скопировать данные из вспомогательного буфера в выходную строку

pop di

call StrCpy

; Выполнить перевод строки

mov si, offset CRSt

call PrnStr

; Восстановить регистры

pop di

pop si

pop bx

pop dx

pop cx

pop ax

ret

InpStr endp

; Выводит на экран в текущую позицию курсора указанную строку.

; Строка должна быть нультерминальной (признаком конца строки является 0)

;

; Вход: DS:SI - адрес начала нультерминальной строки

;

PrnStr proc

; Сохранить регистры

push ax

push dx

push bx

; Получить в AX длину строки

call StrLen

; Будем использовать для вывода функцию DOS 09H, поэтому

; 0-терминальную строку превращаем в '$'-терминальную

mov bx, ax

mov byte ptr bx[si], '$'

; Вывести строку

mov ah, 09h

mov dx, si

int 21h

; Снова сделать строку 0-терминальной

mov byte ptr bx[si], 0

; Восстановить регистры

pop bx

pop dx

pop ax

ret

PrnStr endp

; Возвращает длину нультерминальной строки (кол-во символов в строке)

;

; Вход: DS:SI - адрес начала нультерминальной строки, в которой выполняется поиск

; Выход: AX - длина строки

;

StrLen proc

; Сохранить регистры

push si

; Выполнить проход по строке пока не дойдем до конца (нулевой байт)

mov ax, 0

SL1: cmp byte ptr [si], 0

je SL2 ; Дошли до конца строки - выход

inc ax

inc si

jmp SL1

; Восстановить регистры

SL2: pop si

ret

StrLen endp

; Копирует нультерминальную строку в другую строку

;

; Вход: DS:SI - адрес начала исходной нультерминальной строки

; DS:DI - адрес начала строки, в которую выполняется копирование

;

StrCpy proc

; Сохранить регистры

push ax

push si

push di

; Выполнять копир., пока не дойдем до конца исх. строки (нулевой байт)

SC1: mov al, [si]

mov [di], al

cmp al, 0

je SC2 ; Дошли до конца строки - выход

inc si

inc di

jmp SC1 ; Новая итерация

SC2: ; Восстановить регистры

pop di

pop si

pop ax

ret

StrCpy endp

; Ищет индекс 1-го вхождения подстроки в строку

;

; Вход: DS:DI - адрес начала нультерминальной основной строки, в которой

; выполняется поиск

; DS:SI - адрес начала нультерминальной подстроки, которую ищем

; Выход: DL - признак успешного завершения поиска (1-подстрока найдена;

; 0-подстрока не найдена). DL=0, если строка и/или подстрока - пустые.

; AX - индекс позиции (начиная с 0) в основной строке, с которой

; начинается 1-е вхождение подстроки. Имеет смысл только при DL=1

;

StrPos proc

; Сохранить регистры

push si

push di

push cx

push bx

push es

; Формируем кадр стека для хранения локальных переменных процедуры

push bp

mov bp, sp

push 0 ; Адрес начала основной строки (далее [bp - 2])

push 0 ; Длина основной строки (далее [bp - 4])

push 0 ; Адрес начала подстроки (далее [bp - 6])

push 0 ; Длина подстроки (далее [bp - 8])

; Начальная инициализация

mov dl, 0 ; Пока еще ничего не нашли

cld ; Сбросить флаг DF - проходы по строкам вперед

push ds ; Поместить в ES тот че адрес, что и в DS

pop es ; (для дальнейших строковых команд)

mov [bp - 2], di ; Сохранить адрес начала основной строки

mov [bp - 6], si ; Сохранить адрес начала подстроки

mov si, [bp - 2] ; Вычислить длину

call StrLen ; основной строки

cmp ax, 0 ; Проверить, возможно основная строка пустая

je @SP4 ; Если основная строка пустая - выход

mov [bp - 4], ax ; Сохранить длину основной строки

mov si, [bp - 6] ; Вычислить длину

call StrLen ; подстроки

cmp ax, 0 ; Проверить, возможно подстрока пустая

je @SP4 ; Если подстрока пустая - выход

mov [bp - 8], ax ; Сохранить длину подстроки

; Цикл поиска вхождения начального символа подстроки в основную строку.

; Поиск в основной строке начинаем с позиции, следующей за позицией,

; в которой нашли такое вхождение на предыдущей итерации цикла

; (на начальной итерации ищем от начального символа основной строки).

; Поиск ведем до 1-го совпадения или до конца основной строки.

mov di, [bp - 2] ; В DI - адрес начала основной строки

mov cx, [bp - 4] ; В CX - длина основной строки

@SP1: mov si, [bp - 6] ; В SI - адрес начала подстроки

mov al, [si] ; В AL - начальный символ подстроки

repne scasb ; Ищем позицию вхождения символа AL в осн. стр.

jne @SP4 ; Дошли до конца основной строки и не нашли

; символ - выход

; DI указывает на позицию символа следующего после найденного символа AL

; в основной строке

; Проверить, что начиная с DI в основной строке осталось символов не

; меньше, чем "длина\_подстроки-1"

mov bx, [bp - 8]

dec bx ; В BX - "длина\_подстроки-1"

cmp cx, bx ; В CX кол-во символов от DI до конца осн. стр.

jl @SP4 ; В основной строке осталось символов меньше,

; чем в подстроке - выход

push cx ; Сохранить длину остатка основной строки

push di ; и позицию символа, следующего после

; найденного в основной строке (для последующих

; возможных поисков)

cmp bx, 0 ; Если "длина\_подстроки-1"=0, значит подстрока найдена

je @SP2

mov cx, bx ; В CX сравнений не больше, чем "длина\_подстроки-1"

inc si ; В подстроке перейти на символ следующий после

; начального

repe cmpsb ; Сравниваем остаток основной строки (начиная с

; с символа DI) с остатком подстроки (начиная с

; символа, следующего за начальным)

jne @SP3

@SP2: ; Подстрока успешно найдена в основной строке

mov ax, di ; DI указывает на позицию символа в осн. строке

; который следует за последним символом найденной

; подстроки

sub ax, [bp - 8] ; AX указывает на позицию в основной строке

; начального символа найденной подстроки

sub ax, [bp - 2] ; В AX индекс позиции (от 0) вхождения подстроки

; в основную строку

mov dl, 1 ; В DL - признак успешного нахождения подстроки

pop di ; Перед завершением извлечь ранее сохраненный DI

pop cx ; Перед завершением извлечь ранее сохраненный CX

jmp @SP4

@SP3: ; Подстрока не найдена

pop di ; В DI позиция в основной строке, с которой

; нужно начинать новый поиск

mov bx, di

sub bx, [bp - 2]

mov cx, [bp - 4] ; В CX длина остатка основной строки, начиная

sub cx, bx ; с позиции DI

jmp @SP1 ; Повторяем процесс поиска от новой позиции DI

@SP4: ; Восстановить регистры

mov sp, bp

pop bp

pop es

pop bx

pop cx

pop di

pop si

ret

StrPos endp

; Процедура выполняет преобразование двухбайтового целого числа без знака

; в символьное ASCII-представление в указанной системе счисления.

; Поддерживаются следующие системы счисления:

; - двочиная (цифры 0..1 и буква "B" в конце)

; - восьмеричная (цифры 0..7 и буква "O" в конце)

; - десятичная (цифры 0..9 без буквы в конце)

; - шестнадцатеричная (цифры 0..9,A..F и буква "H" в конце)

;

; Вход: AX - значение двухбайтового целого числа без знака

; DS:SI - адрес начала нультерминальной строки, в которую нужно записать результат

; DH - система счисления, в которую нужно преобразовать число

; 2 - двоичная;

; 8 - восьмеричная;

; 10 - десятичная;

; 16 - шестнадцатеричная;

;

IntToStr proc

; Сохранить регистры

push ax

push cx

push dx

push bx

; Получить в регистре BX длину символьного представления числа

push ax

push dx

mov bx, 0

mov cx, 0

mov cl, dh

mov dx, 0

ITS1: div cx

mov dx, 0

inc bx

cmp ax, 0

jne ITS1 ; Цикл до тех пор, пока не останется значащих разрядов в исходном числе

pop dx

pop ax

; Записать в конец строки символ обозначающий систему счисления и 0-завершающий символ

cmp dh, 2 ; Для двоичной системы счисления

jne ITS2

inc bx

mov byte ptr bx[si], 0

dec bx

mov byte ptr bx[si], 'B'

jmp ITS6

ITS2: cmp dh, 8 ; Для восьмеричной системы счисления

jne ITS3

inc bx

mov byte ptr bx[si], 0

dec bx

mov byte ptr bx[si], 'O'

jmp ITS6

ITS3: cmp dh, 10 ; Для десятичной системы счисления

jne ITS4

mov byte ptr bx[si], 0

jmp ITS6

ITS4: cmp dh, 16 ; Для шестнадцатеричной системы счисления

jne ITS5 ; Неверное значение входного параметра DH - выход

inc bx

mov byte ptr bx[si], 0

dec bx

mov byte ptr bx[si], 'H'

jmp ITS6

ITS5: mov bx, 0 ; Никакого преобразования не будет - пустая строка

mov byte ptr bx[si], 0

jmp ITS8

; Цикл преобразования числа в строковое представление

ITS6: mov cx, 0

mov cl, dh

ITS7: mov dx, 0

div cx

push si

mov si, dx

mov dl, DigSymbTab[si] ; Соотв. символ из строки DigSymbTab

pop si

dec bx

mov byte ptr bx[si], dl

cmp ax, 0

jne ITS7 ; Цикл до тех пор, пока не останется значащих разрядов в исходном числе

ITS8:

; Восстановить регистры

pop bx

pop dx

pop cx

pop ax

ret

IntToStr endp

code ends

; Сегмент данных программы

data segment para public use16 "data"

; Текстовые сообщения

Msg1 db 'STRING IS : ', 0

Msg2 db 'SUBSTRING IS : ', 0

Msg3 db 'SUBSTRING IS FOUND AT POSITION : '

StPosStr db 6 dup (0) ; Строковое значение найденной позиции

Msg4 db 'SUBSTRING IS NOT FOUND', 0

CRSt db 13,10, 0 ; Перевод строки

; Переменные программы

MainSt db 254+1 dup(0) ; Содержимое основной строки

SubSt db 254+1 dup(0) ; Содержимое подстроки

StMaxLen dw 254 ; Максимальное кол-во символов в строках

; Переменные для процедуры InpStr

InpStrBuf db 2+254+1 dup(0) ; Буфер ввода строки

; Таблица цифровых символов

DigSymbTab db '0123456789ABCDEF', 0

data ends

; Сегмент стека программы

stk segment stack use16

db 256 dup (0)

stk ends

end main ; Конец программы с точкой входа main

# **Лабораторная работа №5**

**Тема:** «Обработка прерываний»

**Цель:** научиться работать с прерываниями из-под asm для DOS

**Задание 1**. В программе имеется циклический счетчик, считающий от 1 до 6. При нажатии любой клавиши содержимое счетчика преобразуется в ASCII код и выводится в определенное место экрана, после чего счетчик продолжает считать. Для анализа нажатия использовать вектор 1Ch.

Код программы представлен ниже:

.186

assume cs:code, ds:data, ss:stk

; Сегмент команд программы

code segment para public use16 "code"

main proc

; Настроить сегментный регистр DS на сегмент данных

mov ax, data ; Установить регистр сегмента данных DS,

mov ds, ax ; указывающим на сегмент data

; \*\*\*\*\*\*\*\*\*\* Основная часть программы \*\*\*\*\*\*\*\*\*\*

; Вывести начальное сообщение

mov ah, 09h

mov dx, offset StartMsg

int 21h

; Считать текущие параметры экрана

call GetTextScreenPrm

; Сохранить адрес предыдущего обработчика прерывания 1Ch

mov ah, 35h ; Функция DOS: получить адрес обработчика

mov al, 1Ch ; Вектор преывания, адрес обработчика которого получаем

int 21h

mov word ptr cs:Old\_Int1Ch, bx ; Сохранить смещение

mov word ptr cs:Old\_Int1Ch+2, es ; Сохранить сегмент

; Установить собственный обработчик прерывания 1Ch

mov ah, 25h ; Функция DOS: установить адрес обработчика

mov al, 1Ch ; Вектор преывания, адрес обработчика которого устанавливаем

mov dx, offset Int1Ch\_Handler ; Смещение обработчика

push ds

push cs

pop ds ; Сегмент обработчика

int 21h

pop ds

; Цикл изменения значений циклического счетчика

@11: inc CounterVal

cmp CounterVal, 6

jbe @12

mov CounterVal, 1 ; Зацикливание счетчика: 6+1 = 1

@12: cmp KeypressFlag, 1 ; Проверка флага нажатия на клавишу

jne @13

mov al, CounterVal ; На клавишу нажали - вывод символа на экран

call OutSymb

mov KeypressFlag, 0 ; Сбросить счетчик нажатия на клавишу

@13: cmp ExitFlag, 1 ; Проверка флага завершения работы

jne @11

; Восстановить предыдущий обработчик прерывания 1Ch

mov ah, 25h ; Функция DOS: установить адрес обработчика

mov al, 1Ch ; Вектор преывания, адрес обработчика которого устанавливаем

push ds

mov dx, word ptr cs:Old\_Int1Ch ; Восстановить смещение

push word ptr cs:Old\_Int1Ch+2 ; Восстановить сегмент

pop ds

int 21h

pop ds

; \*\*\*\*\*\*\*\*\*\* Завершение работы программы \*\*\*\*\*\*\*\*\*\*

@31: mov ax, 4C00h ; Выход из

int 21h ; программы

main endp

; Получает параметры текущего текстового режима экрана

GetTextScreenPrm proc

; Сохранить регистры

push ax

push bx

; Выполнить считывание информации

mov ah, 0Fh ; Получить параметры видеорежима

int 10h ; на выходе в BH - номер активной видеостраницы

mov ScrActPgNum, bh

; Восстановить регистры

pop bx

pop ax

ret

GetTextScreenPrm endp

; Выводит указанный символ на текущую страницу экрана в текущую позицию курсора

; Вход: AL - ASCII-код символа

OutSymb proc

; Сохранить регистры

push ax

push bx

push cx

; Выполнить вывод символа в текущую позицию курсора

mov ah, 0Ah

mov bh, ScrActPgNum ; на видеостраницу BH

;mov al, al ; в AL указать код символа

mov cx, 1 ; количество повторо символа

int 10h

; Восстановить регистры

pop cx

pop bx

pop ax

ret

OutSymb endp

; Обработчик прерывания 1Ch

Int1Ch\_Handler proc

; Вызвать предыдущий обработчик прерываний - цепочку обработчиков

pushf

call dword ptr cs:[Old\_Int1Ch]

; Сохранить регистры

pusha

push ds

; Отключить прерывания от таймера (чтобы не было повторной входимости)

cli ; Запретить все прерывания

in al, 21h ; Прочитать текущую маску контроллера прерываний Master

or al, 00000001b ; Устанавливаем 0-й бит, отвечающий за IRQ0-таймер

; - тем самым запрещаем прерывания от него

out 21h, al ; Записываем маску в контроллер - таймер запрещен

sti ; Разрешить все прерывания

; Выполнить проверку нажатия клавиши

mov ax, data ; В DS записать адрес сегмента данных для

mov ds, ax ; доступа к переменным программы

mov ah, 01h ; Проверить, нажата ли клавиша

int 16h

jz @IH1 ; Если ZF-0, то клавиша нажата

mov ah, 00h ; Если клавиша нажата, то считать код клавиши

int 16h

mov KeypressFlag, 1 ; Установить признак нажатия на любую клавишу

cmp al, '0' ; Если была нажата клавиша "0",

jne @IH1

mov ExitFlag, 1 ; то установить флаг завершения работы

@IH1: ; Включить прерывания от таймера (чтобы таймер снова работал)

cli ; Запретить все прерывания

in al, 21h ; Прочитать текущую маску контроллера прерываний Master

and al, 11111110b ; Сбрасываем 0-й бит, отвечающий за IRQ0-таймер

; - тем самым разрешаем прерывания от него

out 21h, al ; Записываем маску в контроллер - таймер разрешен

sti ; Разрешить все прерывания

; Восстановить регистры

pop ds

popa

iret

; Поля данных обработчика в сегменте команд

Old\_Int1Ch dd ? ; Адрес предыдущего обработчика прерывания 1Ch

Int1Ch\_Handler endp

code ends

; Сегмент данных программы

data segment para public use16 "data"

; Стартовое текстовое сообщение

StartMsg db 'Текущее значение счетчика: ', '$'

; Текущие параметры экрана

ScrActPgNum db 0 ; Номер активной видеостраницы

; Текущее значение счетчика (1..6)

CounterVal db 0

; Признак нажатия клавиши (необходимости вывести текущее значение счетчика)

; 0-нет; 1-да

KeypressFlag db 0

; Признак необходимости завершить программу (нажали клавишу "0")

; 0-нет; 1-да

ExitFlag db 0

data ends

; Сегмент стека программы

stk segment stack use16

db 256 dup (0)

stk ends

end main ; Конец программы с точкой входа main

# Проект

Тема: “Создание схемы на основе Arduino UNO ”