МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

ОТЧЕТ

по лабораторной работе № 3

«Разработка программы ввода-вывода и обработки   
числовой информации»

Вариант №19

по дисциплине

Принципы и методы  
организации системных программных средств

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Викулова Е. Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В. А.

\_\_\_19-В-2 .

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Оглавление

[Цель работы 3](#_Toc90336002)

[Задание 3](#_Toc90336003)

[Структура программы 3](#_Toc90336004)

[Сегменты программы 4](#_Toc90336005)

[Сегмент данных 4](#_Toc90336006)

[Сегмент стека 5](#_Toc90336007)

[Сегмент кода 5](#_Toc90336008)

[Макросы 6](#_Toc90336009)

[P1- макрос выводы сообщений 6](#_Toc90336010)

[P2- макрос ввода строки символов 6](#_Toc90336011)

[Процедуры. Алгоритмы обработки введенных значений 7](#_Toc90336012)

[DIAPAZON PROC 7](#_Toc90336013)

[DOPUST PROC 7](#_Toc90336014)

[AscToBin PROC 8](#_Toc90336015)

[BinToAsc PROC 9](#_Toc90336016)

[Алгоритмы арифметической обработки над двоичными числами 10](#_Toc90336017)

[Нахождение минимального и максимального элемента. 10](#_Toc90336018)

[Нахождение отрицательных элементов и расчёт их суммы 10](#_Toc90336019)

[Нахождение суммы положительных чисел 11](#_Toc90336020)

[Нахождение среднего положительных чисел 12](#_Toc90336021)

[Используемые функции прерываний 13](#_Toc90336022)

[Результаты работы 14](#_Toc90336023)

[Теоретическая часть. Вопрос 9. Функции прерывания 16h BIOS для ввода с клавиатуры. 15](#_Toc90336024)

[Функция 00h: прочитать данные с клавиатуры 16](#_Toc90336025)

[Функция 10h: прочитать данные с клавиатуры 18](#_Toc90336026)

[Вывод 18](#_Toc90336027)

# Цель работы

Приобретение навыков: разработки программы ввода-вывода десятичных чисел со знаком, логической и арифметической обработки введённых чисел, контроль ситуаций переполнения и деления на 0.

# Задание

Согласно варианту, написать программу, выполняющую:

* Ввод массива целых многозначных десятичных чисел со знаком;
* Арифметическую и логическую обработку массива чисел (по вариантам);
* Преобразование числового результата в ASCII-строку и отображение результата на экране.

В программе должен быть предусмотрен контроль ситуаций переполнения с выводом на экран соответствующего сообщения.

Вариант 19: Определить минимальный и максимальный элементы массива, среднее положительных чисел, сумму отрицательных.

# Структура программы

Разработана программа является много сегментной и имеет расширение exe. При создании программы была использована модель small. В программе имеет 3 сегмента:

* Сегмент данных .data, связанный с сегментным регистром ds
* Сегмент стека .stack, связанный с сегментным регистром ss.
* Сегмент кода .code, связанный с сегментным регистром cs;

В коде программы имеются следующие макросы:

* p1 - выводит указанную строку
* p2 - ввод с клавиатуры символов и сохранение в переданной строке

В коде программы имеются следующие процедуры:

* DIAPAZON – проверка диапазона вводимых чисел
* DOPUST – проверка допустимости вводимых символов
* AscToBin – перевод ASCII кодов чисел в двоичное число
* BinToAsc – обратная AscToBin

## Сегменты программы

### Сегмент данных

Сегмент данных используется для инициализации строк сообщений, массивов и переменных.

.data

mess0 db 'Input:5 numbers in [-29999,29999]',10,13,'$'

mess00 db 'Press <Enter> after each number',10,13,'$'

mess1 db 'Enter number:$'

in\_str label byte ;Строка символов (не более 6)

razmer db 7

kol db (?)

stroka db 7 dup (?) ;Знак числа (для отрицательных), 5 цифр, enter

number dw 5 dup (0) ;Массив чисел

siz dw ;Количество чисел

maxnum dw 0 ;Наибольшее число

minnum dw 0 ;Наименьшее число

negSum dw 0 ;Сумма отрицательных

posSum dw 0 ;Сумма положительных

posAvg dw 0 ;Среднее положительных чисел

perevod db 10,13,'$'

text\_err1 db 'Input error!',10,13,'$'

messovf db 13,10,7,'Overflow!','$'

mesposavg db 13,10,'Positive Average:','$'

messmax db 13,10,'Max:','$'

messmin db 13,10,'Min:','$'

mesposkol db 13,10,'Count of positive:','$'

mesnegSum db 13,10,'Sum of negative:','$'

mesposSum db 13,10,'Sum of positive:','$'

out\_str db 6 dup (' '),'$'

flag\_err equ 1

### Сегмент стека

Сегмент данных указывает адрес начала стека.

.stack 256

### Сегмент кода

Сегмент кода содержит код исполняемой программы и процедуры, вызываемые во время её исполнения.

## Макросы

### P1- макрос выводы сообщений

p1 macro f1;вывод сообщений на экран

push ax

push dx

mov dx,offset f1

mov ah,9

int 21h

pop dx

pop ax

endm

### P2- макрос ввода строки символов

p2 macro f2;ввод строки символов

push ax

push dx

mov dx,offset f2

mov ah,0ah

int 21h

pop dx

pop ax

endm

## Процедуры. Алгоритмы обработки введенных значений

### DIAPAZON PROC

Для обработки ситуации выхода за пределы допустимого диапазона было решено ограничить диапазон от -29999 до 29999. При такой обратке нам необходимо контролировать чтобы было введено 5 символом и первым числовым символом является цифра 2.

DIAPAZON PROC

;проверка диапазона вводимых чисел -29999,+29999

;буфер ввода - stroka

;через bh возвращается флаг ошибки ввода

xor bh,bh;

xor si,si ;номер символа в вводимом числе

;если ввели менее 5 символов проверим их допустимость

cmp kol,5

jb dop

;если ввели 5 или более символов проверим является ли первый минусом

cmp stroka,2dh

jne plus ;если 1 символ не минус, проверим число символов

;если первый - минус и символов меньше 6 проверим допустимость символов

cmp kol,6

jb dop

inc si ;иначе проверим первую цифру

jmp first

plus: cmp kol,6 ;введено 6 символов и первый - не минус

je error1 ; ошибка

first: cmp stroka[si],32h ;сравним первый символ с 2

jna dop ;если первый <=2 -проверим допустимость символов

error1: mov bh,flag\_err ;иначе bh=flag\_err

dop: ret

DIAPAZON ENDP

### DOPUST PROC

Проверки на допустимость недостаточно. Если строка имеет 5 символов и первый числовой символ это 2, то необходимо проверить все оставшиеся символы на вхождение в числовой диапазон. Для этого мы убеждаемся, что ASCII коды введенных символов принадлежат промежутку от 30h до 39h.

DOPUST PROC

;проверка допустимости вводимых символов

;буфер ввода - stroka

;si - номер символа в строке

;через bh возвращается флаг ошибки ввода

xor bh,bh

xor si,si

xor ah,ah

xor ch,ch

mov cl,kol ;в ch количество введенных символов

m11: mov al,[stroka+si] ; в al - первый символ

cmp al,2dh ;является ли символ минусом

jne testdop ;если не минус - проверка допустимости

cmp si,0 ;если минус - является ли он первым символом

jne error2 ;если минус не первый -ошибка

jmp m13

;является ли введенный символ цифрой

testdop:cmp al,30h

jb error2

cmp al,39h

ja error2

m13: inc si

loop m11

jmp m14

error2: mov bh, flag\_err ;при недопустимости символа bh=flag\_err

m14: ret

DOPUST ENDP

### AscToBin PROC

Для удобства работы с числами переводим их в двоичные. Для этого из кода каждой цифры вычитаем 30h, умножаем результат на вес разряда и складываем полученные произведения. Отрицательные числа представляем в дополнительном коде при помощи команды neg.

AscToBin PROC

;в cx количество введенных символов

;в bx - номер символа начиная с последнего

;буфер чисел - number, в di - номер числа в массиве

xor ch,ch

mov cl,kol

xor bh,bh

mov bl,cl

dec bl

mov si,1 ;в si вес разряда

n1: mov al,[stroka+bx]

xor ah,ah

cmp al,2dh ;проверим знак числа

je otr ;если число отрицательное

sub al,30h

mul si

add [number+di],ax

mov ax,si

mov si,10

mul si

mov si,ax

dec bx

loop n1

jmp n2

;представим отрицательное число в дополнительном коде

otr: neg [number+di]

n2: ret

AscToBin ENDP

### BinToAsc PROC

Для вывода полученных результатов переводим числа в их ASCII коды, т.е. выполняем обратную операцию.

BinToAsc PROC

;преобразование числа в строку

;число передается через ax

xor si,si

add si,5

mov bx,10

push ax

cmp ax,0

jnl mm1

neg ax

mm1: cwd

idiv bx

add dl,30h

mov [out\_str+si],dl

dec si

cmp ax,0

jne mm1

pop ax

cmp ax,0

jge mm2

mov [out\_str+si],2dh

mm2: ret

BinToAsc ENDP

# Алгоритмы арифметической обработки над двоичными числами

## Нахождение минимального и максимального элемента.

Для нахождение максимального и минимального элементов формируем нулевую гипотезу: в max и min помещаем первое число.

Алгоритм

Цикл-Для каждого элемента массива:

Условие: элемент > max ?

Да: max = элемент

Нет: идем дальше

Условие: элемент < min ?

Да: min = элемент

Всё-цикл

;Поиск максимального и минимального элементов

mov ax,number

mov maxnum,ax

mov minnum,ax

max:

mov ax,number+si

cmp ax,maxnum

jle min

mov maxnum,ax

jmp min

min: cmp ax,minnum

jge next

mov minnum,ax

jmp next

next: inc si

inc si

loop max

## Нахождение отрицательных элементов и расчёт их суммы

Алгоритм

Цикл-Для каждого элемента массива:

Условие: элемент <0 ?

Да: Сумма\_отрицательных += элемент

Нет: переход к следующему элементу

Проверка переполнения

Всё-цикл

; Поиск отрицательных элементов и рассчет их суммы

mov cx, siz ; В (cx) - размер массива

mov si, offset number

negFind:

mov ax, [si]

cmp ax, 0

jge endNegFind

add negSum, ax ; Сумма отрицательных

jo OVR ; Если произошло переполнение

endNegFind:

inc si

inc si

loop negFind

Замечание: в данном случае переполнение – это выход из допустимого диапазона – затирается знаковый бит.

## Нахождение суммы положительных чисел

Алгоритм

Цикл-Для каждого элемента массива:

Условие: элемент > 0 ?

Да: Сумма\_положительных += элемент

Количество\_положительных+=1

Нет: переход к следующему элементу

Проверка переполнения

Всё-цикл

; Поиск положительных элементов и рассчет их суммы

mov cx, siz ; В (cx) - размер массива

mov si, offset number

xor bx, bx

posFind:

mov ax, [si]

cmp ax, 0

jl endPosFind

add posSum, ax ; Сумма положительных

jo OVR ; Если произошло переполнение

inc bx ; Считаем количество положительных элементов

endPosFind:

inc si

inc si

loop posFind

Замечание: в данном случае переполнение – это выход из допустимого диапазона – затирается знаковый бит.

## Нахождение среднего положительных чисел

Алгоритм

1. Заносим аккумулятор значение суммы положительных чисел.
2. Делим содержимое аккумулятора на количество положительных чисел, сохранённое в регистре bx.
3. Сохраняем полученное значение в переменной posAvg.

; Поиск среднего положительного

mov dx,0

mov ax,posSum

div bx

mov posAvg, ax

# Используемые функции прерываний

В программе используются следующие функции прерываний:

**int 10h** – функция стандартного видеосервиса ROM-BIOS, в программе она используется с ah=0(номер функции) – очистка экрана, установить поля BIOS, установить режим; al = 3 (номер режима).

mov ax, 03h ;Установка текстового видеорежима, очистка экрана

int 10h ;ah=0 (номер функции),al=3 (номер режима)

**int 21h –** вызов функций DOS.

09h – функция вывода строки

mov dx, offset srting

mov ah, 09h

int 21h

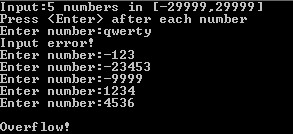
0Ah – извлечение буфера с пользовательским вводом

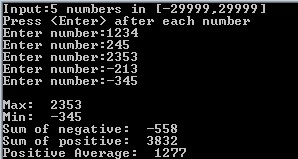
mov dx, offset srting

mov ah, 0Ah

int 21h

# Результаты работы





# Теоретическая часть. Вопрос 9. Функции прерывания 16h BIOS для ввода с клавиатуры.

Набор функций для работы с клавиатурой, предоставляемый в распоряжение программиста прерыванием BIOS INT 16h, включает в себя функции для выборки кода нажатого символа из буфера с ожиданием нажатия, функции для проверки содержимого буфера и для управления содержимым буфера.

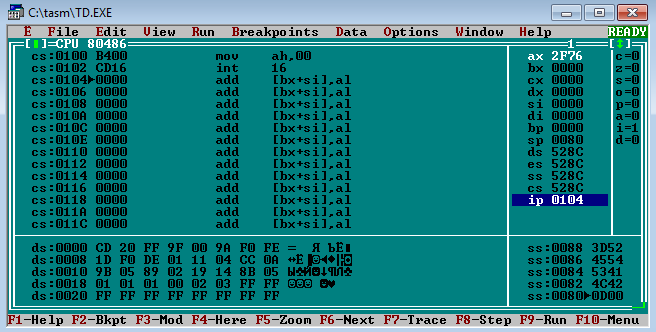
* АН = 00h - чтение с ожиданием двухбайтового кода из буфера клавиатуры. Прочитанный код возвращается в регистре АХ: младший байт - в регистре AL, старший - в АН. Если нажата ASCII-клавиша, в AL помещается ASCII-код символа, в АН - скэн-код. При нажатии специальных клавиш AL равен 0, а в АН возвращается расширенный скэн-код.
* АН = 01h - чтение без ожидания двухбайтового кода из буфера клавиатуры. Если буфер пуст, в 1 выставляется флаг нуля ZF. В противном случае в АХ возвращается двухбайтовый код из буфера клавиатуры, но продвижение указателя "головы" буфера не производится, т.е. код "остается" в буфере.
* АН = 02h - определение состояния шифт- и триггерных клавиш.
* АН = 03h - управление режимом автоповтора.
* АН = 04h - вкл/выкл звуковой сигнал клавиш
* Функция АН = 05h может использоваться для имитации нажатии клавиш в демонстрационных программах, программах переноса текста и т.д.
* Функции АН = 10 - 12h являются аналогами функций 00 - 02h, но предназначены для использования в компьютерах с клавиатурой 101 /102 клавиши.

## Функция 00h: прочитать данные с клавиатуры

Перед вызовом прерывания требуется записать в регистр AH значение функции: 00h

После выполнения функции в регистр будет помещена следующая информация:

* AH – скан-код символа
* AL – ASCII-код символа

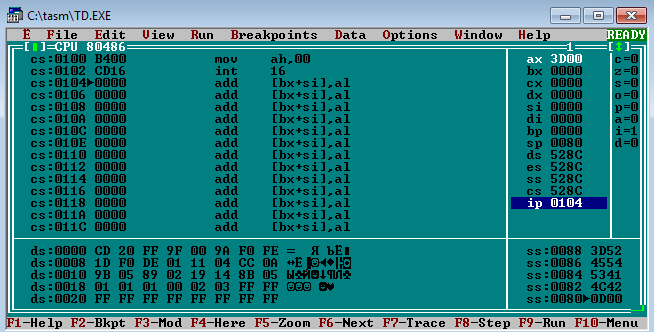


Нажатие клавиши v

AH: 2Fh – скан код символа v

AL: 76h - ASCII код символа v

При нажатии управляющих и функциональных клавиш функция выдает ASCII-код со значением 0, благодаря чему их можно отличить от алфавитно-цифровых. Однако таким способом функциональные клавиши можно отличать лишь по скан-кодам.



Нажатие клавиши F3

AH: 3Dh – скан код символа F3

AL: 00 - ASCII код для функциональных клавиш

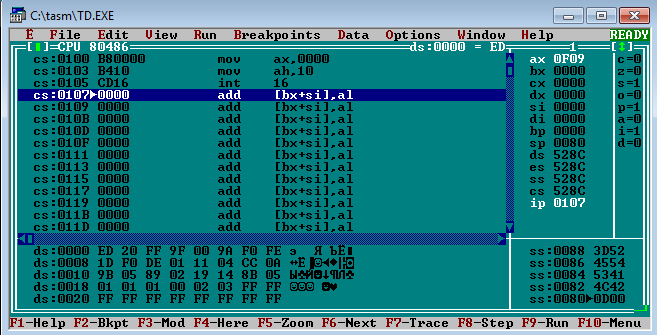
## Функция 10h: прочитать данные с клавиатуры

Функция 10h является усовершенствованным вариантом функции 00h. Она позволяет получить скан-коды функциональных клавиш F1-F12, а также клавиш управления курсором.

Перед вызовом прерывания требуется записать в регистр AH значение функции: 10h

После выполнения функции в регистр будет помещена следующая информация:

* AH – расширенный скан-код символа
* AL – ASCII-код символа



Нажатие клавиши tab

AH: 0Fh – скан код клавиши tab

AL: 09h – ASCII код клавиши tab

# Вывод

В ходе выполнения данной работы были получены навыки разработок программ ввода-вывода десятичных чисел со знаком, логической и арифметической обработки введенных чисел со знаком, контроль ситуации переполнения и деления на 0.