

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
Задание	3
Структура программы	3
Сегменты программы.....	4
Сегмент данных	4
Сегмент стека	5
Сегмент кода	5
Макросы	6
P1- макрос вывода сообщений.....	6
P2- макрос ввода строки символов	6
Процедуры. Алгоритмы обработки введенных значений.....	7
DIAPAZON PROC.....	7
DOPUST PROC	7
AscToBin PROC	8
BinToAsc PROC	9
Алгоритмы арифметической обработки над двоичными числами	10
Нахождение минимального и максимального элемента.....	10
Нахождение отрицательных элементов и расчёт их суммы	10
Нахождение суммы положительных чисел	11
Нахождение среднего положительных чисел	12
Используемые функции прерываний.....	13
Результаты работы	14
Теоретическая часть. Вопрос 9. Функции прерывания 16h BIOS для ввода с клавиатуры.	15
Функция 00h: прочитать данные с клавиатуры	16
Функция 10h: прочитать данные с клавиатуры	18
Вывод.....	18

Цель работы

Приобретение навыков: разработки программы ввода-вывода десятичных чисел со знаком, логической и арифметической обработки введенных 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 вывода 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 обработке 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,col
        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 нахождения 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 расчет их суммы
    mov cx, siz          ; B (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 расчет их суммы
    mov cx, siz          ; B (cx) - размер массива
    mov si, offset number
    xor bx, bx
posFind:
    mov ax, [si]
    cmp ax, 0
    jnl 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 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

```
Input:5 numbers in [-29999,29999]
Press <Enter> after each number
Enter number:qwerty
Input error!
Enter number:-123
Enter number:-23453
Enter number:-9999
Enter number:1234
Enter number:4536

Overflow!
```

```
Input:5 numbers in [-29999,29999]
Press <Enter> after each number
Enter number:1234
Enter number:245
Enter number:2353
Enter number:-213
Enter number:-345

Max: 2353
Min: -345
Sum of negative: -558
Sum of positive: 3832
Positive Average: 1277
```

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 AX: младший байт - в регистре AL, старший - в AH. Если нажата ASCII-клавиша, в AL помещается ASCII-код символа, в AH - скэн-код. При нажатии специальных клавиш AL равен 0, а в AH 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 AX 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 нажатия 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 АН значение функции: 00h

После выполнения функции в регистр будет помещена следующая информация:

- АН – скан-код символа
- AL – ASCII-код символа

The screenshot shows the TASM/TD.EXE debugger window. The title bar indicates the file path is C:\tasm\TD.EXE. The menu bar includes File, Edit, View, Run, Breakpoints, Data, Options, Window, and Help. The status bar at the bottom shows function key shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, and F10-Menu.

The main window is divided into several panes. The top-left pane shows the assembly code being executed, with the instruction pointer (IP) at 0104. The top-right pane shows the current state of the registers: ax=2F76, bx=0000, cx=0000, dx=0000, si=0000, di=0000, bp=0000, sp=0080, ds=528C, es=528C, ss=528C, and ip=0104. The bottom-left pane shows the memory dump for the current instruction, displaying hexadecimal values and their ASCII equivalents. The bottom-right pane shows the stack pointer (ss) and the current stack address (0080), with the stack contents displayed in hexadecimal and ASCII.

```
C:\tasm\TD.EXE
E File Edit View Run Breakpoints Data Options Window Help
[ ] CPU 80486 1 [ ] READY
cs:0100 B400 mov ah,00 ax 2F76 c=0
cs:0102 CD16 int 16 bx 0000 z=0
cs:0104 0000 add [bx+si],al cx 0000 s=0
cs:0106 0000 add [bx+si],al dx 0000 o=0
cs:0108 0000 add [bx+si],al si 0000 p=0
cs:010A 0000 add [bx+si],al di 0000 a=0
cs:010C 0000 add [bx+si],al bp 0000 i=1
cs:010E 0000 add [bx+si],al sp 0080 d=0
cs:0110 0000 add [bx+si],al ds 528C
cs:0112 0000 add [bx+si],al es 528C
cs:0114 0000 add [bx+si],al ss 528C
cs:0116 0000 add [bx+si],al ip 0104
cs:0118 0000 add [bx+si],al
cs:011A 0000 add [bx+si],al
cs:011C 0000 add [bx+si],al

ds:0000 CD 20 FF 9F 00 9A F0 FE = Я ЬЕ
ds:0008 1D F0 DE 01 11 04 CC 0A +E ]<=>|<
ds:0010 9B 05 89 02 19 14 8B 05 W4M01M2
ds:0018 01 01 01 00 02 03 FF FF 000 0v
ds:0020 FF FF FF FF FF FF FF FF ss:0088 3D52
ss:0086 4554
ss:0084 5341
ss:0082 4C42
ss:0080 0D00

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

Нажатие клавиши v

АН: 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.

The screenshot shows the TASM debugger window titled "C:\tasm\TD.EXE". The menu bar includes File, Edit, View, Run, Breakpoints, Data, Options, Window, and Help. The status bar at the bottom lists function keys: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, and F10-Menu.

The main window is divided into several panes. The top-left pane shows assembly code with addresses from 0100 to 011C. The top-right pane shows register values (ax, bx, cx, dx, si, di, bp, sp, ds, es, ss, cs, ip). The bottom-left pane shows memory dump data. The bottom-right pane shows segment registers (ss) and their values.

Address	Instruction	Register Value
cs:0100	mov ah,00	ax 3D00
cs:0102	int 16	bx 0000
cs:0104	add [bx+sil],al	cx 0000
cs:0106	add [bx+sil],al	dx 0000
cs:0108	add [bx+sil],al	si 0000
cs:010A	add [bx+sil],al	di 0000
cs:010C	add [bx+sil],al	bp 0000
cs:010E	add [bx+sil],al	sp 0080
cs:0110	add [bx+sil],al	ds 528C
cs:0112	add [bx+sil],al	es 528C
cs:0114	add [bx+sil],al	ss 528C
cs:0116	add [bx+sil],al	cs 528C
cs:0118	add [bx+sil],al	ip 0104
cs:011A	add [bx+sil],al	
cs:011C	add [bx+sil],al	

Memory dump (ds:0000):

```

ds:0000 CD 20 FF 9F 00 9A F0 FE = Я bE
ds:0008 1D F0 DE 01 11 04 CC 0A +E
ds:0010 9B 05 89 02 19 14 8B 05
ds:0018 01 01 01 00 02 03 FF FF
ds:0020 FF FF FF FF FF FF FF
  
```

Segment registers (ss):

```

ss:0088 3D52
ss:0086 4554
ss:0084 5341
ss:0082 4C42
ss:0080 0D00
  
```

Нажатие клавиши F3

AH: 3Dh – скан код символа F3

AL: 00 - ASCII код для функциональных клавиш

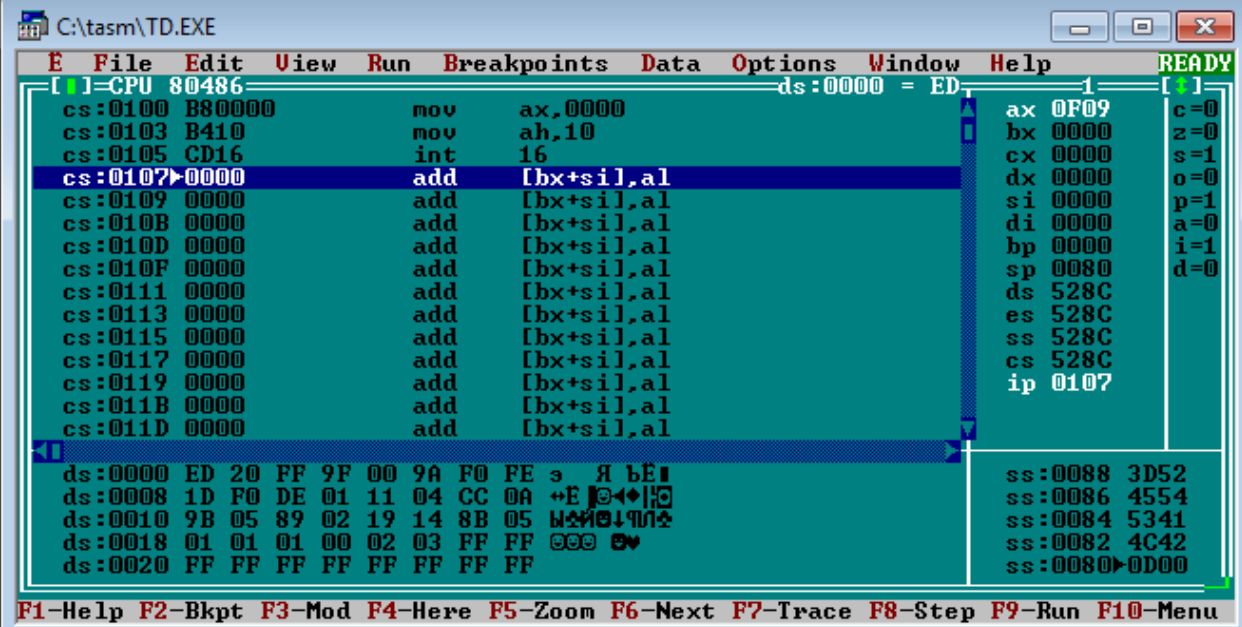
Функция 10h: прочитать данные с клавиатуры

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 АН значение функции: 10h

После выполнения функции в регистр будет помещена следующая информация:

- АН – расширенный скан-код символа
- АЛ – ASCII-код символа



The screenshot shows the TASM TD.EXE debugger window. The main window displays assembly code for a program. The code starts with `mov ax, 0000`, `mov ah, 10`, and `int 16`. The instruction `add [bx+si], al` is highlighted. The right pane shows the current state of registers: `ax 0F09`, `bx 0000`, `cx 0000`, `dx 0000`, `si 0000`, `di 0000`, `bp 0000`, `sp 0080`, `ds 528C`, `es 528C`, `ss 528C`, `cs 528C`, and `ip 0107`. The bottom status bar shows the current instruction pointer (IP) at 0107.

Нажатие клавиши tab

АН: 0Fh – скан код клавиши tab

АЛ: 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.