Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчет

Лабораторная работа №3

Выполнил студент гр. 150505: Лисичкин Д.А.

Проверил: Туровец Н.О.

Минск 2022

**Тема работы:** Целочисленные арифметические операции. Обработка массивов числовых данных.

**Цель работы:** Ознакомиться с арифметическими операциями над целочисленными данными, обработкой массивов чисел, ознакомиться с правилами оформления ассемблерных процедур.

**Вариант лабораторной работы:** 11.В выходной файл поместить только те строки входного файла, которые содержат все указанные символы.

**Код программы**

data segment

otst db ' ',0Dh,0Ah,'$'

start\_mes db " Fill the array!$"

EX\_INCORRECT\_INPUT db " Invalid input!$"

EX\_BORDER db " Not valid borders!$"

press\_key db " Press any key!$"

write\_left db " Write left border: $"

write\_right db " Write right border: $"

your\_massive db " Your massive:$"

your\_elements db " Searched elements:$"

mas dw 11,11 dup (0)

el dw 1,1 dup (0)

left dw 1,1 dup(0)

right dw 1,1 dup(0)

tab db " $"

size equ 10

ends

stack segment

db 100

ends

code segment

proc to\_string

push ax

push bx

push cx

mov bx, 10

mov ax, [di]

xor cx, cx

xor dx, dx

test ax, ax

jne choose\_sign

mov [el], 1

mov [el+1], 1

mov [el+2], '0'

mov [el+3], '$'

jmp exit

choose\_sign:

cmp ax, 0

js negative

jmp count

negative:

mov [el+2], '-'

neg ax

mov dx, 1

count:

push ax

push dx

c5:

xor dx, dx

test ax, ax

je make\_string

inc cx

div bx

add ax, 0

jmp c5

make\_string:

pop dx

add dx, cx

lea si, el

mov [si], dl

mov [si+1], dl

add si, 1

add si, dx

mov [si+1], '$'

pop ax

c6:

xor dx, dx

div bx

xchg ax, dx

add al, '0'

mov [si], al

dec si

xchg ax, dx

cmp ax, 0

je exit

loop c6

exit:

pop cx

pop bx

pop ax

ret

endp

proc output\_mas\_el

push ax

push bx

push cx

call to\_string

output el

output tab

pop cx

pop bx

pop ax

ret

endp

proc outputmas

lea di, mas

mov cl, [di+2]

add di, 4

c1:

call output\_mas\_el

add di, 2

loop c1

ret

endp

proc inputmas

mov [mas], size

mov [mas+2], size

lea di, mas

add di, 4

mov cx, [mas + 2]

lea si, el

c2:

call get\_number

cmp [si+2], '-'

jne continue

neg bx

continue:

mov [di], bx

add di, 2

endl

loop c2

ret

endp

proc get\_number

push di

push si

push cx

begin:

input el

lea di, el

test di, di

jz error

add di, 2

mov si, 10

xor ax, ax

xor bx, bx

mov cx, [el+1]

xor ch, ch

cmp [di], '+'

jne skip

skip\_letter:

inc di

dec cx

jmp cycle1

skip:

cmp [di], '-'

je skip\_letter

cycle1:

cmp [di], ' '

je end

mov bx, [di]

xor bh, bh

cmp bx, '0'

jl error

cmp bx, '9'

jg error

sub bx, '0'

mul si

jc error

add ax, bx

jc error

inc di

loop cycle1

mov bx, ax

success:

pop cx

pop si

pop di

jmp end

error:

output EX\_INCORRECT\_INPUT

endl

jmp begin

end:

ret

endp

macro input buffer

push ax

push dx

mov [buffer], 200

mov [buffer + 1], 0

lea dx, buffer

mov ah, 0ah

int 21h

mov ax, [buffer + 1]

add dx, ax

mov dx, '$'

pop dx

pop ax

endm

macro output buffer

push dx

push ax

lea dx, buffer+2

mov ah, 9

int 21h

pop ax

pop dx

endm

macro endl

output otst

endm

proc wait\_for\_key

push ax

output press\_key

endl

mov ah, 1

int 21h

pop ax

ret

endp

proc task

\_task:

endl

output write\_left

call get\_number ;left

cmp [si+1], '-'

jne cont1

neg bx

cont1:

push bx

endl

output write\_right

call get\_number

cmp [si+1], '-'

jne cont2

neg bx

cont2:

push bx

endl

lea di, mas

mov cx, [di+2]

add di, 4

pop bx

pop ax

cmp ax, 0

js step1

jmp step2

step1:

cmp bx, 0

jns c

cmp ax, bx

jg error2

jmp go

step2:

cmp bx, 0

js error2

cmp ax, bx

jg error2

jmp go

go:

output your\_elements

endl

c:

;call wait\_for\_key

cmp [di], 0

js step3

jmp step4

step3:

cmp ax, 0

jns next\_step

cmp [di], ax

jb next\_step

cmp bx, 0

jns success2

cmp [di], bx

jg next\_step

jmp success2

step4:

cmp ax, 0

js step41

cmp [di], ax

jb next\_step

cmp [di], bx

jg next\_step

jmp success2

step41:

cmp bx, 0

js next\_step

cmp [di], bx

jg next\_step

jmp success2

success2:

call output\_mas\_el

next\_step:

add di, 2

loop c

ret

error2:

output EX\_BORDER

endl

jmp \_task

ret

endp

start:

mov ax, data

mov ds, ax

mov es, ax

output start\_mes

endl

call inputmas

endl

output your\_massive

endl

call outputmas

call task

mov ah, 1

int 21h

mov ax, 4c00h

int 21h

ends

end start

**Вывод в консоли после компиляции**

