Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №4

на тему:

**«**ЦЕЛОЧИСЛЕННЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ. ОБРАБОТКА МАССИВОВ ЧИСЛОВЫХ ДАННЫХ**»**

БГУИР 6-05-0612-02 85

|  |
| --- |
| Выполнил студент группы 353503  МАРТИНОВИЧ Андрей Александрович |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 5.** Выполнить набор операций над двумя целыми числами.

# 2 Выполнение работы

Листинг кода 1 – Текст программы задания 1

name "calc2"

PUTC MACRO char

PUSH AX

MOV AL, char

MOV AH, 0Eh

INT 10h

POP AX

ENDM

org 100h

jmp start

msg1 db "enter first number: $"

msg2 db "enter the operator: + - \* / : $"

msg3 db "enter second number: $"

msg4 db 0dh,0ah , 'the approximate result of my calculations is : $'

err1 db "wrong operator!", 0Dh,0Ah , '$'

smth db " and something... $"

msg5 db "overflow",0Dh,0Ah , '$'

ten dw 10

make\_minus DB ?

opr db '?'

num1 dd ?

num2 dd ?

start:

lea dx, msg1

mov ah, 09h ; output string at ds:dx

int 21h

call scan\_num

mov num1, cx

putc 0Dh

putc 0Ah

lea dx, msg2

mov ah, 09h ; output string at ds:dx

int 21h

mov ah, 1 ; single char input to AL.

int 21h

mov opr, al

putc 0Dh

putc 0Ah

cmp opr, 'q' ; q - exit in the middle.

je exit

cmp opr, '\*'

jb wrong\_opr

cmp opr, '/'

ja wrong\_opr

lea dx, msg3

mov ah, 09h

int 21h

call scan\_num

mov num2, cx

lea dx, msg4

mov ah, 09h

int 21h

cmp opr, '+'

je do\_plus

cmp opr, '-'

je do\_minus

cmp opr, '\*'

je do\_mult

cmp opr, '/'

je do\_div

wrong\_opr:

lea dx, err1

mov ah, 09h ; output string at ds:dx

int 21h

exit:

ret

do\_plus:

mov ax, num1

add ax, num2

jc overflow\_message

jo overflow\_message

call print\_num ; print ax value.

jmp exit

do\_minus:

mov ax, num1

sub ax, num2

jc overflow\_message

jo overflow\_message

call print\_num ; print ax value.

jmp exit

do\_mult:

mov ax, num1

imul num2 ; (dx ax) = ax \* num2.

jc overflow\_message

jo overflow\_message

call print\_num ; print ax value.

jmp exit

do\_div:

mov dx, 0

mov ax, num1

idiv num2 ; ax = (dx ax) / num2.

cmp dx, 0

jnz approx

jc overflow\_message

jo overflow\_message

call print\_num ; print ax value.

jmp exit

approx:

call print\_num ; print ax value.

lea dx, smth

mov ah, 09h ; output string at ds:dx

int 21h

jmp exit

overflow\_message:

lea dx, msg5

mov ah, 09h

int 21h

jmp exit

SCAN\_NUM PROC NEAR

push dx

push ax

push si

mov cx, 0

mov cs:make\_minus, 0

next\_digit:

mov ah, 00h ;scan from keyboeard

int 16h ; write to al

mov ah, 0Eh ;print

int 10h

cmp al, '-'

je set\_minus

cmp al, 0Dh ;cr

jne not\_cr

jmp stop\_input

not\_cr:

cmp al, 08h ;back

jne backspace\_checked

mov dx, 0

mov ax, cx

div cs:ten

mov cx, ax

PUTC ' '

PUTC 08h ;back

jmp next\_digit

backspace\_checked:

cmp al, '0' ;check 0-9

jb remove\_not\_digit

cmp al, '9'

ja remove\_not\_digit

jmp ok\_digit

remove\_not\_digit:

PUTC 08h ;remove digit

PUTC ' '

PUTC 08h

jmp next\_digit

ok\_digit:

push ax

mov ax, cx

mul cs:ten ;result in dx|ax

mov cx, ax

pop ax

cmp dx, 0

jne too\_big

sub al, '0'

mov ah, 0 ;save only lower part

mov dx, cx ;reserve copy

add cx, ax

jc too\_big2

jmp next\_digit

set\_minus:

mov cs:make\_minus, 1

jmp next\_digit

too\_big2:

mov cx, dx ; restore cx from dx

MOV dx, 0

too\_big:

mov ax, cx

div cs:ten ; save in ax

mov cx, ax

PUTC 08h ;back

PUTC ' '

PUTC 08h ;back

jmp next\_digit

stop\_input:

cmp cs:make\_minus, 0

je not\_minus

neg cx

not\_minus:

pop si

pop ax

pop dx

ret

SCAN\_NUM ENDP

PRINT\_NUM PROC NEAR

push dx

push ax

cmp ax, 0

jne not\_zero

PUTC '0'

jmp printed

not\_zero:

cmp AX, 0

jns positive

neg AX

PUTC '-'

positive:

call PRINT\_NUM\_UNS

printed:

pop ax

pop dx

ret

PRINT\_NUM ENDP

PRINT\_NUM\_UNS PROC NEAR

push ax

push bx

push cx

push dx

mov cx, 1

mov bx, 10000

cmp ax, 0

jz print\_zero

begin\_print:

cmp bx,0

jz end\_print

cmp cx, 0

je calc

cmp ax, bx

jb skip

calc:

mov cx, 0

mov dx, 0

div bx ; AX = AX / BX (DX=remainder).

add al, 30h ; convert to ASCII code.

PUTC al

mov ax, dx ; get remainder from last div.

skip:

; calculate BX=BX/10

push ax

mov dx, 0

mov ax, bx

div cs:ten

mov bx, ax

pop ax

jmp begin\_print

print\_zero:

PUTC '0'

end\_print:

pop dx

pop cx

pop bx

pop ax

ret

PRINT\_NUM\_UNS ENDP

# Вывод

В ходе выполнения лабораторной работы с программным обеспечением, предназначенным для сборки, отладки и запуска программ на языке ассемблера ознакомились с арифметическими операциями над целочисленными данными, а также с правилами оформления ассемблерных процедур.