Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №4

на тему:

**«Целочисленные арифметические операции. Обработка массивов**

**числовых данных.»**

БГУИР 6-05-0612-02 005

|  |
| --- |
| Выполнила студент группы 353504  АНТОНОВА Лидия Сергеевна |
|  |
| (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

В данной лабораторной работе была разработана программа на ассемблере, которая выполняет набор арифметических и логических операций над двумя целыми числами. Программа написана для использования в эмуляторе Emu8086 и включает в себя операции сложения, вычитания, умножения, деления, а также логические операции AND, OR, XOR и NOT.

.model small

.stack 100h

.data

num1 dw 0

num2 dw 0

minus db 0

res dw 11 dup(0)

remainder dw 6 dup(0)

; arithmetic

msgFirst db 'Input first number: $'

msgSecond db 'Input second number: $'

msgAddition db 'Addition result: $'

msgSubtraction db 'Subtraction result: $'

msgMultiplication db 'Multiplication result: $'

msgDivision db 'Division result: $'

msgRemainder db 'Remainder: $'

; logical

msgAND db 'AND result: $'

msgOR db 'OR result: $'

msgXOR db 'XOR result: $'

msgNOT1 db 'NOT result for first number: $'

msgNOT2 db 'NOT result for second number: $'

msgDivByZero db 'Division by zero. $'

msgOverflow db 'Overflow detected. $'

Enter db 0Dh, 0Ah, '$'

msgMinus db '-$'

msgEmpty db 'String is empty.$'

.code

start:

mov ax, @data

mov ds, ax

lea dx, msgFirst

call output

lea si, num1

call input

lea dx, Enter

call output

lea dx, msgSecond

call output

lea si, num2

call input

lea dx, Enter

call output

;start arithm operations

; add

lea dx, msgAddition

call output

mov si, [num1]

mov di, [num2]

adc si, di

call checkOverflow

mov ax, si

call showResult

; sub

lea dx, msgSubtraction

call output

mov si, [num1]

mov di, [num2]

sub si, di

mov ax, si

call showResult

; mul

lea dx, msgMultiplication

call output

mov ax, [num1]

mov di, [num2]

imul di

call checkOverflow

call showResult

; div

lea dx, msgDivision

call output

mov dx, 0

mov ax, [num1]

mov di, [num2]

call divide

lea dx, Enter

call output

;start arithm operations

; AND

lea dx, msgAND

call output

mov ax, [num1]

mov di, [num2]

and ax, di

call showResult

; OR

lea dx, msgOR

call output

mov ax, [num1]

mov di, [num2]

or ax, di

call showResult

; XOR

lea dx, msgXOR

call output

mov ax, [num1]

mov di, [num2]

xor ax, di

call showResult

; NOT

lea dx, msgNOT1

call output

mov ax, [num1]

not ax

call showResult

lea dx, msgNOT2

call output

mov ax, [num2]

not ax

call showResult

exit:

mov ax, 4C00h

int 21h

divide proc

test di, di

jz zero

mov ax, [num1]

mov bx, [num2]

mov cx, 0

test ax, ax

jns skipNeg1

neg ax

inc cx

skipNeg1:

test bx, bx

jns skipNeg2

neg bx

inc cx

skipNeg2:

cwd

idiv bx

cmp cx, 1

jne positiveResult

neg ax

positiveResult:

mov remainder, dx

lea di, res

call convertToStr

mov dx, di

call output

lea dx, Enter

call output

lea dx, msgRemainder

call output

mov ax, remainder

lea di, remainder

call convertToStr

mov dx, di

call output

ret

zero:

lea dx, msgDivByZero

call output

ret

divide endp

output:

mov ah, 09h

int 21h

ret

input proc

mov ax, 0

mov bx, 0

mov cx, 0

mov dx, 0

mov bl, 0

mov minus, 0

inputLoop:

mov ah, 01h

int 21h

cmp al, '-'

je checkNegative

cmp al, 0Dh

je inputEnd

cmp al, '0'

jb invalidInput

cmp al, '9'

ja invalidInput

sub al, '0'

mov dl, al

mov al, bl

mov cl, 10

mul cl

add ax, dx

inc cx

mov bx, ax

jmp inputLoop

invalidInput:

mov ah, 0Eh

mov al, 08h

int 10h

mov al, ' '

int 10h

mov al, 08h

int 10h

jmp inputLoop

checkNegative:

cmp cx, 0

jne invalidInput

mov minus, 1

inc cx

jmp inputLoop

inputEnd:

cmp minus, 1

je setNeg

mov [si], bx

cmp cx, 0

je empty

ret

setNeg:

neg bx

mov [si], bx

ret

input endp

empty:

lea dx, Enter

call output

lea dx, msgEmpty

call output

jmp exit

convertToStr proc

mov bx, 10

add di, 5

mov byte ptr [di], '$'

mov cx, ax

test cx, cx

jns positiveNumber

neg cx

lea dx, msgMinus

call output

positiveNumber:

mov ax, cx

convertLoop:

dec di

mov dx, 0

div bx

add dl, '0'

mov [di], dl

test ax, ax

jnz convertLoop

ret

convertToStr endp

showResult:

lea di, res

call convertToStr

mov dx, di

call output

lea dx, Enter

call output

ret

checkOverflow:

jo overflowDetected

ret

overflowDetected:

lea dx, msgOverflow

call output

ret

end start

Программа начинается с инициализации сегмента данных, где определяются переменные для хранения двух целых чисел и сообщения для ввода и вывода результатов. Пользователь вводит первое и второе числа, которые сохраняются в переменных num1 и num2. Ввод осуществляется по символам с помощью прерывания 21h, и программа проверяет, что введенные символы являются допустимыми цифрами. Также предусмотрена обработка отрицательных чисел.

После ввода чисел программа выполняет арифметические операции. Сначала производится сложение двух чисел с использованием команды ADC. Результат выводится на экран с помощью отдельной процедуры. Затем выполняются вычитание и умножение, при этом каждая операция также включает проверку на переполнение.

Для деления реализована отдельная процедура, которая обрабатывает возможные ошибки, такие как деление на ноль. В случае успешного деления программа выводит как частное, так и остаток.

После выполнения арифметических операций программа переходит к логическим операциям. Каждая логическая операция (AND, OR, XOR и NOT) выполняется последовательно, и результаты выводятся на экран аналогичным образом.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 2.1 — Пример выполнения задания

# Вывод

В ходе выполнения лабораторной работы была изучена работа с ассемблером, а также освоены основные арифметические и логические операции. Программа успешно выполнила все поставленные задачи, включая ввод данных, обработку ошибок и вывод результатов. Это позволило глубже понять принципы работы с низкоуровневыми языками программирования и их возможности для манипуляции с числами.