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 67

|  |
| --- |
| Выполнил студент группы 353503  КОХАН Артём Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 7.** Ввести массив целых чисел размерностью 30 элементов. Найти наиболее часто встречающееся число.

# 2 Выполнение работы

Максимальную длину строки для ввода возьмём равную 7-ми символом, где последний символ будет ‘$’. А остальные 6 будут хранить число с учётом знака минуса или плюса. Тем не менее для строки ввода выделено 9 байт, где 1-й байт будет хранить максимально возможное число символов, которое можно ввести, а 2-й байт число символов, которое было введено. Вывод будет осуществляться с помощью функции AH = 0Ah, прерывания int21h. В программе имеются необходимые проверки на ввод, в случае неправильного ввода в консоль будут выводится соответствующие сообщения. Для поиска наиболее часто встречающегося числа в программе выделим ещё один массив, где будет хранится частота встречаемости каждого числа, исходя из этого в конце программы происходит вывод.

Листинг кода 1 – Текст программы задания 1

.model small

.stack 100h

.data

infoSizeArray db "Input size array from 1 to 99:",0Dh,0Ah,'$'

infoStr db 0Ah, "Input numbers for Array...",0Dh,'$'

input db 09h dup('$') ; input str

output db 08h dup('$') ; output str

maxLen equ 07h

maxSize db 0h ; maxSize array

maxLenArray equ 03h

infoArray db 0Dh,0Ah,"Your array: $"

separator db ", $"

result db 0Dh,0Ah,0Ah,"The most repeated number is $"

enterNum db 0Dh,0Ah,"Enter integer number from -32768 to 32767: $"

error db 0Dh,0Ah,"Incorrect input!!!$"

overflow db 0Dh,0Ah,"Overflow input!!!$"

empty db 0Dh,0Ah,"Empty input!!!$"

array dw 99 dup (?) ; input array

repArray dw 99 dup (0h) ; repeat numbers array

.code

info proc

mov dx, offset infoStr

call outputString

ret

info endp

infoSize proc

mov dx, offset infoSizeArray

call outputString

ret

infoSize endp

clearOutput proc

mov di, offset output

mov cx, 8

clearOutputLoop:

mov [di], '$'

inc di

loop clearOutputLoop

ret

clearOutput endp

outputString proc

mov ah, 09h

int 21h

ret

outputString endp

inputString proc

mov ah, 0Ah

int 21h

ret

inputString endp

inputSizeArray proc

call infoSize

mov dx, offset input

mov [input], maxLenArray

call inputString

xor cx, cx

xor ax, ax

mov si, offset input[2] ; first symbol in str

mov cl, [input+1]

cmp cl, 0

je emptyInput

cmp [si], '0'

je incorrectInput

converseSize:

mov bx, 0Ah

mul bx ; next digit

jo inputOverflow ; OVERFLOW

mov bl, [si] ; symbol from str

sub bx, '0'

; check: 0 <= num <= 9

cmp bx, 9

jg incorrectInput

cmp bx, 0

jl incorrectInput

add ax, bx ; + this digit

jo inputOverflow ; OVERFLOW

inc si ; next symbol

loop converseSize

cmp ax, 0

jl inputOverflow ; OVERFLOW

mov maxSize, al

ret

inputSizeArray endp

strToNum MACRO str

xor cx, cx

xor ax, ax

mov si, offset str[2] ; first symbol in str

mov cl, [str+1] ; numbers symbol in str

cmp cl, 0 ; if input empty

je emptyInput

cmp [si], '+' ; symbol '+' check

jne minus

sub cl, 1

inc si

jmp conversePlus

minus:

cmp [si], '-' ; symbol '-' check

jne conversePlus

sub cl, 1

inc si

converseMinus:

mov bx, 0Ah

imul bx ; next digit

jo inputOverflow ; OVERFLOW

mov bl, [si] ; symbol from str

sub bx, '0'

; check: 0 <= num <= 9

cmp bx, 9

jg incorrectInput

cmp bx, 0

jl incorrectInput

sub ax, bx ; + this digit

jo inputOverflow ; OVERFLOW

inc si ; next symbol

loop converseMinus

jmp strToNumEnd

conversePlus:

mov bx, 0Ah

mul bx ; next digit

jo inputOverflow ; OVERFLOW

mov bl, [si] ; symbol from str

sub bx, '0'

; check: 0 <= num <= 9

cmp bx, 9

jg incorrectInput

cmp bx, 0

jl incorrectInput

add ax, bx ; + this digit

jo inputOverflow ; OVERFLOW

inc si ; next symbol

loop conversePlus

cmp ax, 0

jl inputOverflow ; OVERFLOW

strToNumEnd:

ENDM

getArray proc

mov di, offset array

xor ch,ch

mov cl, maxSize

jmp arrayInputLoop

emptyInput: ; EMPTY input

mov dx, offset empty

call outputString

cmp [input], maxLenArray

je inputSizeArray

pop cx

jmp arrayInputLoop

inputOverflow: ; OVERFLOW

mov dx, offset overflow

call outputString

pop cx

jmp arrayInputLoop

incorrectInput: ; NO Number

mov dx, offset error

call outputString

cmp [input], maxLenArray

je inputSizeArray

pop cx

arrayInputLoop:

mov dx, offset enterNum

call outputString

mov dx, offset input

call inputString

push cx ; save cx to STACK

strToNum input

pop cx ; load cx from STACK

mov [di], ax ; num from ax to array after strToNum

add di, 2

loop arrayInputLoop

ret

getArray endp

outputArray proc

xor ch,ch

mov cl, maxSize

mov si, offset array

mov dx, offset infoArray

call outputString

outputArrayLoop:

push cx ; save cx to STACK

mov ax, [si] ; load num to ax

call outputNum ; input num from ax

pop cx

cmp cx, 1 ; last num

je finishOutput

mov dx, offset separator

call outputString

add si, 2 ; next num

loop outputArrayLoop

finishOutput:

ret

outputArray endp

setRepArray proc

xor ch, ch

mov cl, maxSize

mov si, offset array

arrayLoop:

mov ax, [si] ; load from ARRAY

push si

push cx

xor ch, ch

mov cl, maxSize

mov si, offset array

searchLoop:

cmp [si], ax ; found element in repeat Array

je found

add si, 2 ; next num in repeat Array

loop searchLoop

found:

sub si, offset array

add si, offset repArray

inc [si] ; add 1 repeat

pop cx

pop si

add si, 2 ; next num in Array

loop arrayLoop

ret

setRepArray endp

getMaxRep proc

xor ch, ch

mov cl, maxSize

mov si, offset repArray ; Temp Adress

mov di, offset repArray ; Max Adress

mov ax, [di] ; Max element

add si, 2

repSearchLoop:

cmp ax, [si] ; cmp with next element

jge skip

mov ax, [si] ; New Max element

mov di, si ; New Max Adress

skip:

add si, 2 ; Next element

loop repSearchLoop

sub di, offset repArray

add di, offset Array

mov ax, [di]

ret

getMaxRep endp

outputNum proc ; output number from ax

call clearOutput

xor cx, cx ; 0 to cx

mov bx, 0Ah ; 10 to bx(DIV)

mov di, offset output

cmp ax, 0 ; greater then or equal 0?

jge pos

mov [di], '-' ; set '-' to output string

inc di

neg ax

pos:

cmp ax, 0

jne toStack

mov [di], '0' ; output if 0

jmp printNum

toStack: ; digit to Stack

cmp ax, 0

je toStr

xor dx, dx ; 0 to dx

div bx ; ax/10 - ost to dl

add dl, '0' ; ascii code get

push dx

inc cx

jmp toStack

toStr:

pop dx

mov [di], dl

inc di

loop toStr

printNum:

mov dx, offset output ; print output string

call outputString

ret

outputNum endp

start:

mov ax, @data

mov ds, ax

mov es, ax

call inputSizeArray

mov [input], maxLen ; set len buffer

call info

call getArray ; create array

call outputArray

mov dx, offset result

call outputString

call setRepArray ; creat Repeat Array

call getMaxRep ; Max Repeted load to ax

call outputNum ; Output Max Repeted Num

mov ax, 4ch

int 21h

end start

# 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, обработкой массивов чисел, а также с правилами оформления ассемблерных процедур.