Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Конструирование программ

Отчёт по лабораторной работе №7

ЦЕЛОЧИСЛЕННЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ. ОБРАБОТКА МАССИВОВ ЧИСЛОВЫХ ДАННЫХ

Студент Киселёва Е.А.

Принимающий Романюк М.В.

Минск 2022

СОДЕРЖАНИЕ

[1 ЦЕЛЬ РАБОТЫ 3](file:///C:\Users\Acer\Downloads\Telegram%20Desktop\ХрищановичАК_КПрог_Лаб5.docx#_Toc121092272)

[2 ЗАДАНИЕ 4](file:///C:\Users\Acer\Downloads\Telegram%20Desktop\ХрищановичАК_КПрог_Лаб5.docx#_Toc121092273)

[3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ 8](file:///C:\Users\Acer\Downloads\Telegram%20Desktop\ХрищановичАК_КПрог_Лаб5.docx#_Toc121092274)

[4 ВЫВОД 9](file:///C:\Users\Acer\Downloads\Telegram%20Desktop\ХрищановичАК_КПрог_Лаб5.docx#_Toc121092275)

# ЦЕЛЬ РАБОТЫ

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

2 ЗАДАНИЕ

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

Листинг 1 ­– Код программы

; Распечатать строку [Макрос]

Print macro str

push ax

push dx

mov ah,09h ; Номер функции DOS

lea dx,str ; Смещение str в DX

int 21h ; 21 прерывание

pop dx

pop ax

endm

; Распечатать символ [Макрос]

printSymb macro ascii

mov ah,06h

mov dl,ascii

int 21h

endm

; Макрос для вывода элементов массива на консоль

Output\_Array\_In\_Console macro

lea si, Array ; SI смещение массива Array

mov Array\_Counter, 0d ; Занулить переменную

dec Array\_Size ; Для исправления индексов, потом оно восстанавливается

Print New\_Line

Print Output\_Array\_Str

Output:

Print Probel\_Str

Print\_Number [si] ; Вывод всех чисел массива

Print Comma\_Str

add si,2d

mov dx, Array\_Size

cmp dx,Array\_Counter

jz End\_Macro

inc Array\_Counter

jmp Output

End\_Macro:

inc Array\_Size

endm

; Модель памяти (код в 1 сегменте, данные и стек в DGROUP)

.model small

; Сегмент кода

.code

; Начало прогарммы

START:

mov ax,@data ; @data - идентификатор сегмента данных

mov ds,ax ; Адрес сегмента данных — в DS

mov es,ax ; Адрес сегмента данных — в ES

;====Задание максимальной длины для вводимого числа======

lea di,String\_Number ; Смещение строки в DI

mov al,06d ; Максимальное число символов для ввода

stosb ; AL в ES:DI

;========================================================

;=================== Задание ===================

Print Task\_Str

Print New\_Line

;========================================================

;===================Ввод размера массива ( 0 < size < 30) ===================

Enter\_Array\_Size:

Print Enter\_ArraySize\_Str

call Enter\_Number

Print New\_Line

; Array\_Size = Main\_Number

mov dx, Main\_Number

mov Array\_Size, dx

cmp Array\_Size, 30d; Если больше чем 30 элементов

ja Exeption\_Array\_Size\_More\_Than\_30

cmp Array\_Size, 0d; Если 0 элементов

jz Exeption\_Array\_Size\_Less\_Than\_1

;========================================================

;===================Заполнение массива===================

Print Enter\_Numbers\_In\_Array\_Str

mov Array\_Counter, 0d

lea si, Array

Enter\_Numbers\_In\_Array\_Loop\_Start:

mov dx, Array\_Counter

cmp dx, Array\_Size

jz After\_Entering\_Array

Print New\_Line

Print\_Number Array\_Counter

push si

call Enter\_Number ; Число в Main\_Number

pop si

mov dx, Main\_Number ; Число Main\_Number -> Array[index]

mov [si], dx

add si, 2d

inc Array\_Counter ; Index для вывода [0],[1]...

jmp Enter\_Numbers\_In\_Array\_Loop\_Start

;========================================================

After\_Entering\_Array: ; Продолжение программы после заполнения массива

; Алгоритм подсчета наиболее встречающегося элемента в массиве

;====================Главный алгоритм====================

lea si, Array ; SI смещение массива Array, SI = Начало массива

mov Array\_Counter, 0d; Счетчик массива для всего массива

mov Pod\_Array\_Counter, 0d; Счетчик для прохода в подмассиве

mov ax, Array\_Size ; Размер массива в AX

mov dx, 0d

While:

cmp ax, Array\_Counter

jz Out\_Of\_Algoritm

mov Curr\_Address, si; В текущий адрес помещаем элемент массива, сохраняем

; inc Counter ; Как минимум всегда один такой элемент

push si ; Сохраняем адрес текущего элемента

lea si, Array

Inside\_While:

cmp ax, Pod\_Array\_Counter

jz After\_Inside\_While

inc Pod\_Array\_Counter

;add si,2d ; Переход к след. элементу массива

mov dx,[si] ; В DX = след. элемент массива

mov bx, Curr\_Address

mov bx, [bx]

cmp dx, bx ; Если такой же элемент, то надо Counter++

jz If\_Same\_Element

Continue: ; В Counter у нас кол-во одинаковых элементов по адресу Curr\_Address

add si,2d ; Переход к след. элементу массива

jmp Inside\_While

After\_Inside\_While:

pop si

; В Max\_Counter заносим максимальное кол-во элементов

mov dx, Max\_Counter

cmp dx, Counter

jl Counter\_In\_MaxCounter ; Если Max\_Counter < Counter => Max\_Counter = Counter

Posle\_Proverki:

add si, 02d

inc Array\_Counter

mov Counter, 0d

mov Pod\_Array\_Counter, 0d

jmp While

;========================================================

Out\_Of\_Algoritm:

; В Next\_Address хранится адрес нашего самого часто втречающегося элемента

mov bx, Next\_Address

mov dx, [bx]

; Вывод результата

Print New\_Line

Print Answer\_Str

Print\_Number dx

Print New\_Line

Output\_Array\_In\_Console ; Макрос для вывода элементов массива на консоль

Print New\_Line

Print End\_Programm\_Str

jmp end

;==================== Прыжки для алгоритма ====================

Counter\_In\_MaxCounter:

mov dx, Counter ; Max\_Counter = Counter

mov Max\_Counter, dx

mov dx, Curr\_Address

mov Next\_Address, dx ; в Next\_Address хранится самый встречабщийся элемент

jmp Posle\_Proverki

If\_Same\_Element: ; Если такой же элемент

inc Counter ; Увеличить счетчик таких элементов

jmp Continue

;========================================================

;====================Процедура ввода числа====================

; Работает правильно, в Main\_Number = Введеное число

Enter\_Number PROC near

pop bp; Загрузить в стек адрес возвращения

Enter\_Number\_Loop\_Start:

; Ввод числа в строку

mov ah,0Ah ; Функция DOS (считать строку символов из STDIN в буфер)

lea dx,String\_Number; Смещение строки в DX

int 21h

; Вычисление длины введеного числа

lea si, String\_Number ; Смещение в SI наше число String\_Number

inc si ; SI++ ; Возможно не надо !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

mov ax, 0d ; Занулить AX

lodsb ; DS:SI в AL

mov cx,ax ; CX = длина введеного числа

cmp cx, 0d ; Если длина = 0, то Exeption\_NumberIsZero ->

jz Exeption\_NumberIsZero

mov Main\_Number, 0d; Занулим главное число

; Будем в AX хранить степент 10 (TEN) ->

; 1234 -> 4 \* 10^0 + 3 \* 10^1 + 2 \* 10^2 + 1 \* 10^3 = 1234

mov ax, 01d ; Будем в AX хранить степень 10

NumberStr\_In\_Number\_Loop:

mov bx, 0d; Занулим BX

mov si, cx; SI = CX -> идем с конца слова, можно было так же поменять флаг DF(направления)

inc si;

mov bl, String\_Number[si]

; Проверка на то, что введеное число(str) является числом(int)

Proverka\_Na\_SymbolASCII:

cmp bl, 48d; Сверка с нулем [ASCII] 48d = 0

jl Exeption\_Invalid\_Number; bl < 48d

cmp bl, 57d; Сверка с 9 [ASCII] 57d = 9

jg Exeption\_Invalid\_Number; bl > 57d

sub bl, 48d; Получить реальное число, "N" - 48 = N

push ax; Сохранение AX

clc; Очистить CF

mul bx; bl = bl \* AX -> Реальное число \* 10^ax ; BX !!!!!

jc Exeption\_OverFlow ; Если есть флаг переноса от mul

add Main\_Number, ax ; Main\_Number + ax = Полноценное число

jc Exeption\_OverFlow ; Если есть флаг переноса от add

pop ax; Восстановление AX

mul TEN; Увеличиваем AX \* 10

dec cx; Уменьшаем счетчик

cmp cx,0d

jne NumberStr\_In\_Number\_Loop

push bp ; Загрузить их стека адерс для ret

ret ;Вернуться туда, откуда была вызвана PROC

Enter\_Number endp

;=======================================================

;====================Процедура вывода числа====================

Print\_Number\_Proc PROC near

mov cx,0h ; Количество цифр у числа

; Добавление цифр в тек

pushDigitsWhile:

mov dx,0h

mov ax,Number\_General ; AX = число

div ten ; DX:AX / ten = DX:AX

mov Number\_General,ax ; numberGeneral уже /= 10

add dx,48d ; DX += 48 (перевод в ASCII)

push dx ; DX = текущая цифра числа

inc cx ; CX ++

; Условие выхода из цикла

cmp Number\_General,0h

jne pushDigitsWhile

; Доставание из стека и печать цифр

printDigitsWhile:

mov dx,0h

mov ax,Number\_General ; AX = число

div ten ; DX:AX / ten = DX:AX

mov Number\_General,ax ; numberGeneral уже /= 10

; Цифра числа DX

; прерывание берет ASCII символа из DL

pop dx

mov ah,06h

int 21h

; Условие выхода из цикла

dec cx

cmp cx,0h

jne printDigitsWhile

ret

Print\_Number\_Proc endp

; Макрос - передает число в процедуру [Number\_General]

Print\_Number macro digit

Print Index\_Array\_Start\_Str

push ax

mov ax,digit

mov Number\_General,ax

call Print\_Number\_Proc

Print Index\_Array\_End\_Str

pop ax

endm

;=================================================================

;====================Exeption\_Array\_Size\_More\_Than\_30==============

Exeption\_Array\_Size\_More\_Than\_30:

Print New\_Line

Print Exeption\_Array\_Size\_More\_Than\_30\_Str

Print New\_Line

jmp Enter\_Array\_Size

;=======================================================

;====================Exeption\_Array\_Size\_Less\_Than\_1==============

Exeption\_Array\_Size\_Less\_Than\_1:

Print New\_Line

Print Exeption\_Array\_Size\_Less\_Than\_1\_Str

Print New\_Line

jmp Enter\_Array\_Size

;=======================================================

;====================Exeption\_NumberIsZero==============

Exeption\_NumberIsZero:

Print New\_Line

Print Exeption\_NumberIsZero\_Str

Print New\_Line

jmp Enter\_Number\_Loop\_Start

;=======================================================

;====================Exeption\_OverFlow==============

Exeption\_OverFlow:

pop ax; Восстановление AX

Print New\_Line

Print Exeption\_OverFlow\_Str

Print New\_Line

jmp Enter\_Number\_Loop\_Start

;=======================================================

;====================Exeption\_Invalid\_Number==============

Exeption\_Invalid\_Number:

Print New\_Line

Print Exeption\_Invalid\_Number\_Str

Print New\_Line

jmp Enter\_Number\_Loop\_Start

;=======================================================

;====================Конец программы====================

end:

mov ax,4C00h ; 4C (выход из программы) в AH

; 00 (успешное звершение) в AL

int 21h ; Функция DOS "Завершить программу"

;=======================================================

; Сегмент данных

.data

;Exceptions

Exeption\_NumberIsZero\_Str db "You didn't enter a number, please repeat the input!$"

Exeption\_Invalid\_Number\_Str db "The number is entered in the wrong format, repeat the input!$"

Exeption\_OverFlow\_Str db "Oops, you've caught an overflow(65535), repeat the input!$"

Exeption\_Array\_Size\_More\_Than\_30\_Str db "The size of the array exceeds the allowed size (30)!$"

Exeption\_Array\_Size\_Less\_Than\_1\_Str db "Array size is less than allowed size(1) !$"

; Strings

New\_Line db 0Ah, 0Dh, '$'; Переход на новую строку

Enter\_ArraySize\_Str db "Enter array size ( 1 <= size <= 30):$"

Enter\_Numbers\_In\_Array\_Str db "Enter numbers in array:$"

Index\_Array\_Start\_Str db "[$"

Index\_Array\_End\_Str db "]$"

End\_Programm\_Str db "Programm END.$"

Task\_Str db "Task : Find the most frequently occurring number in the array.$"

Answer\_Str db "Answer: $"

; Строки для красивого вывода массива

Output\_Array\_Str db "Array: $"

Probel\_Str db ' $'

Comma\_Str db ',$'

; ВВОД ЧИСЕЛ

String\_Number db 10 DUP(?) ; Строка, в которую вводим число

Main\_Number dw 0d ; Введенное число (высчитывается)

; Число для передачи в процедуру

Number\_General dw 0

; Счетчик для ввода чисел в массив

Array\_Counter dw 0

Pod\_Array\_Counter dw 0

; Для алгоритма перевода числа

TEN dw 10d

; МАССИВ

Array\_Size dw 10 ; Заполняемое число элементов массива

Array dw 40 DUP(?) ; Массив чисел

; Основной алгоритм

Counter dw 0

Max\_Counter dw 0

Curr\_Address dw 0

Next\_Address dw 0

; Символы перехода на новую строку (v,<)

toNewLine db 0Ah, 0Dh, '$'

; Строка для передачи в процедуру

strGeneral\_offset dw 0

strGeneral\_size db 0

; Сегмент стека (256 байт)

.stack 100h

; Конец программы

end START

3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

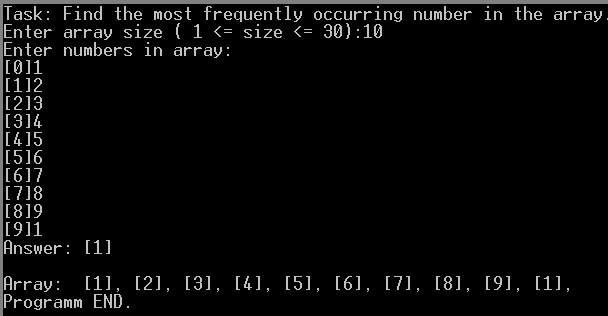


Рисунок 1 – результат выполнения программы с корректным вводом

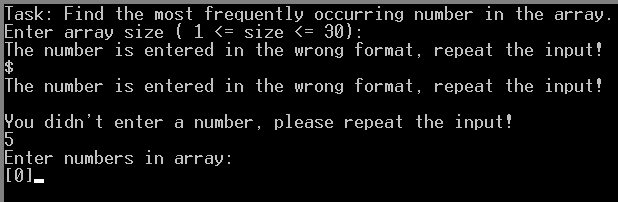


Рисунок 2 – результат выполнения программы с некорректным вводом

4 ВЫВОД

Была создана ассемблерная программа, которая дает возможность ввести массив целых чисел размерностью 30 (или менее) элементов, а после находит в этом массиве наиболее часто встречающееся число.

Также в ходе работы я ознакомилась с арифметическими операциями над целочисленными данными, обработкой массивов чисел и с правилами оформления ассемблерных процедур.