|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 4**

**Дисциплина: Машинно-зависимые языки и основы компиляции**

**Название лабораторной работы: Обработка массивов и матриц**

Студент гр. ИУ6-41Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тарасова М.А.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Данилюк С.С.**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель работы:** изучение приёмов моделирования обработки массивов и матриц на языке ассемблера.

**Задание:** Дан массив 25 элементов. Определить количество различных элементов и поместить их в начале массива. Организовать ввод массива и вывод результатов.

**Код программы:**

%include "lib.asm"

section .data

; MAS dw 3 9 2 2 5 8 1 5 -1 8 10 15 9 5 -1 90 90 44 7 7 60 7 90 22 8

ExitMsg db "",10

lenExit equ $-ExitMsg

EnterMsg db "Enter the array which is 25 or smaller numbers: ", 10

lenEnter equ $-EnterMsg

err\_num db "Only numbers and spaces can be entered", 10

err\_num\_len equ $-err\_num

ResultMsg db "New array is: ", 10

lenResult equ $-ResultMsg

input times 255 db 0

len\_input equ $-input

section .bss

NewUnicMAS resw 25

NewMAS resw 25

OutBuf resb 50

MAS resw 25

section .text

global \_start

\_start:

mov rax, 1

mov rdi, 1

mov rsi, EnterMsg

mov rdx, lenEnter

syscall

mov rax, 0

mov rdi, 0

mov rsi, input

mov rdx, len\_input

syscall

mov rcx, rax

xor rdx, rdx

xor r10, r10

process\_line:

cmp byte[input + rdx], 10

je process\_number

cmp byte[input + rdx], ' '

jne next

mov byte[input + rdx], 10

cmp r10, rdx

jne process\_number

jmp next

process\_number:

push rdx

call StrToInt64

cmp rbx, 0

jne error\_num

mov [MAS + 2 \* rdi], rax

inc rdi

pop rdx

mov r10, rdx

inc r10

lea rsi, [input + r10]

next:

inc rdx

loop process\_line

;logic

mov ebx, 0

mov ecx, edi

mov r10, rdi

xor rsi, rsi ; Unic counter

xor rdi, rdi ; Replicated counter

cycle1: ; цикл прохода по массиву MAS в поиске уникальных элементов

push rcx

mov dx, 0

mov rcx, r10

mov ax, [ebx\*2 + MAS]

push rbx

mov ebx, 0

cycle2: ; цикл в котором происходит сравнение

; текущего элемента цикла1 со всеми

cmp ax, [ebx\*2 + MAS] ; элементами массива MAS для выявления уникального элемента

je plus

jmp next1

plus:

inc dx ; при нахождении повторения увеличиваем счётчик на 1

next1:

inc ebx

loop cycle2

cmp dx, 2 ; если счётчик равен 1, то это уникальный элемент

jl dobU

jge dobR

jmp next2

dobU:

mov [rsi\*2 + NewUnicMAS], ax ; добавляем уникальные элементы в NewUnicMAS

inc rsi

jmp next2

dobR:

mov [rdi\*2 + NewMAS], ax ; добавляем повторяющиеся элменеты в NewMAS

inc rdi

next2:

pop rbx

inc ebx

pop rcx

loop cycle1

mov ebx, 0

mov rcx, rsi

new\_array1:

mov ax, [ebx\*2 + NewUnicMAS] ; добавляем в массив сначала неповторяющиеся элементы

mov [ebx\*2 + MAS], ax

inc ebx

loop new\_array1

mov edx, ebx

mov ebx, 0

mov rcx, rdi

new\_array2:

mov ax, [ebx\*2 + NewMAS] ; потом добавляем в массив повторяющиеся элементы

mov [edx\*2 + MAS], ax

inc ebx

inc edx

loop new\_array2

; Result Message

mov rax, 1

mov rdi, 1

mov rsi, ResultMsg

mov rdx, lenResult

syscall

mov ebx, 0

mov rcx, r10

output\_cycle:

mov ax, [ebx\*2 + MAS]

inc ebx

push rcx

; конвертируем частное из целого в строку

mov esi, OutBuf

cwde

call IntToStr64

; write

mov edx, eax

mov eax, 1

mov edi, 1

syscall

pop rcx

loop output\_cycle

; write ExitMsg

mov rax, 1

mov rdi, 1

mov rsi, ExitMsg

mov rdx, lenExit

syscall

exit: ;exit

mov rax, 60

xor rdi, rdi

syscall

error\_num:

mov rax, 1

mov rdi, 1

mov rsi, err\_num

mov rdx, err\_num\_len

syscall

jmp exit

**Результат программы:**

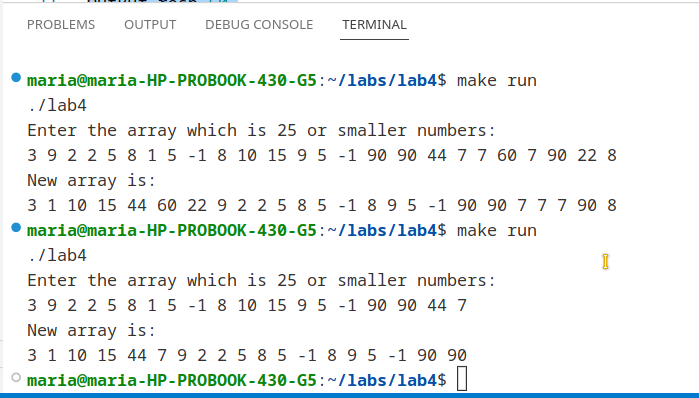
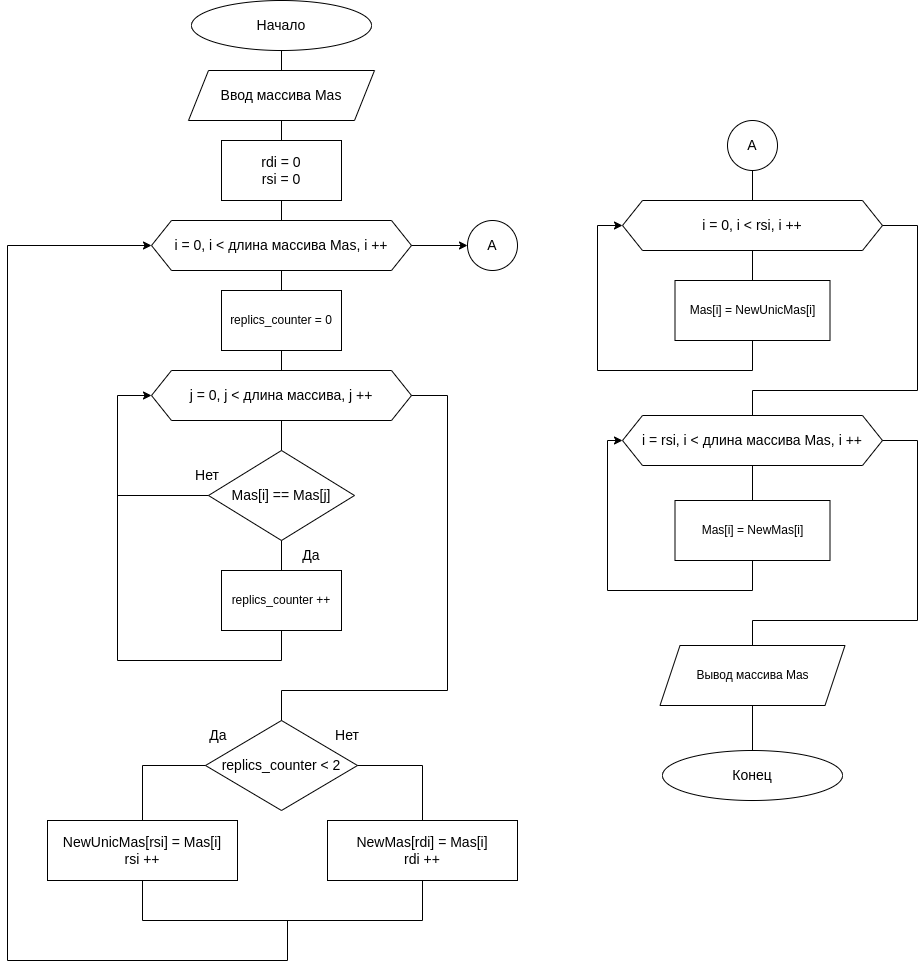


Рисунок 1 — Результат программы

**Схема алгоритма:**



**Вывод:** в данной лабораторной работе с помощью ассемблера обрабатывался массив на 25 и менее элементов. Программа разделяла массив на повторяющиеся и уникальные элементы и делала из них новый массив.

**Контрольные вопросы:**

1. **Почему в ассемблере не определены понятия «массив», «матрица»?**

Они неопределены, так как матрица и массив — это последовательность элементов памяти, лежащая по конкретному адресу.

1. **Как в ассемблере моделируются массивы?**

В ассемблере есть специальная команда LEA (r32, mem), для загрузки исполнительного адреса. Последовательность чисел в переменной можно назвать массивом, пример: A WORD 1,2,3,4,5,6 – что будет являться массивом из 6 чисел.

1. **Поясните фрагмент последовательной адресации элементов массива Почему при этом для хранения частей адреса используют регистры?**

Используется поочередный доступ к адресам элементов, а указатель на конкретный элемент последовательно смещается.

mov ebx, 0 - EBX присваивает 0, чтобы указатель был на первый элемент.

mov [ebx\*2 + MAS], ax - [ebx\*2 + MAS] присваивает EAX, EBХ умножается на 2 - так как WORD.

inc ebx - EBX = EBX + 1 – смещение указателя на следующий элемент.

Для хранения адреса используют регистры так как они предоставляют полный адрес текущего элемента.

1. **Как в памяти компьютера размещаются элементы матриц?**

Они размещаются последовательно друг за другом, строчка за строчкой.

1. **Чем моделирование матриц отличается от моделирования массивов? В каких случаях при выполнении операций для адресации матриц используется один регистр, а в каких – два?**

Моделирование матриц отличается от моделирования массивов только тем, что надо помимо движения по строкам надо еще отслеживать движения по столбцам. Один регистр используется в случае, если движение идет по строкам, а два - если одновременно еще и по столбцам.