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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по лабораторной работе № 4

Дисциплина: МЗЯиОК

Название: Программирование обработки массивов и матриц (14 вариант)

Студент ИУ6-43Б (Группа) 14.05.2024 (Подпись, дата) В.К. Залыгин (И.О. Фамилия)

Преподаватель (Подпись, дата) (И.О. Фамилия)

Москва, 2024

Цель работы

Изучение приемов моделирования обработки массивов и матриц в языке ассемблера.

Задание

Задание для 14-го варианта: дана матрица 7x3. Обнулить элементы с четной суммой индексов. Организовать ввод матрицы и вывод результатов.

Схема алгоритма

Схема алгоритма представлена на рисунках 1 и 2.

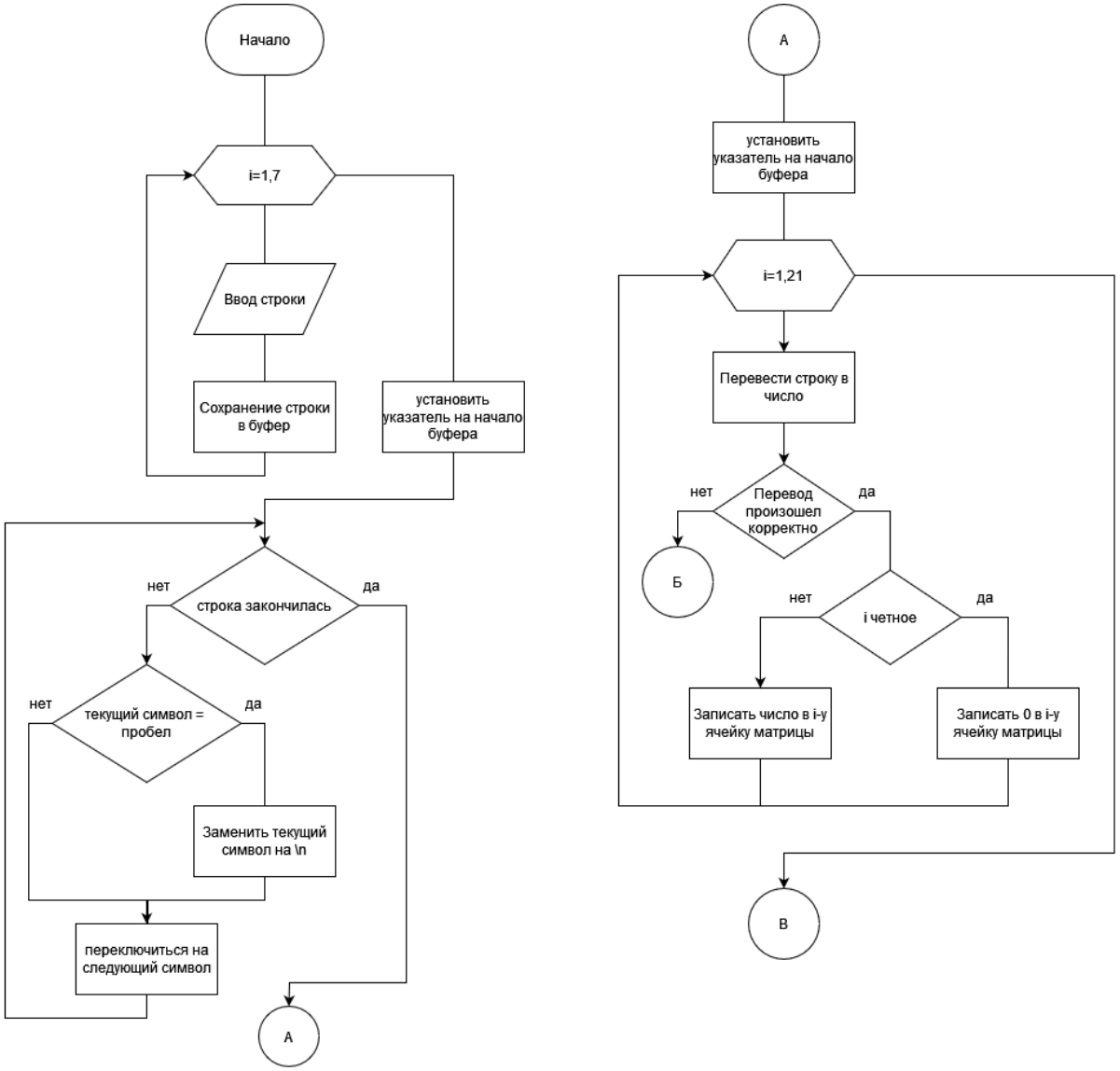

Рисунок 1 – Схема алгоритма

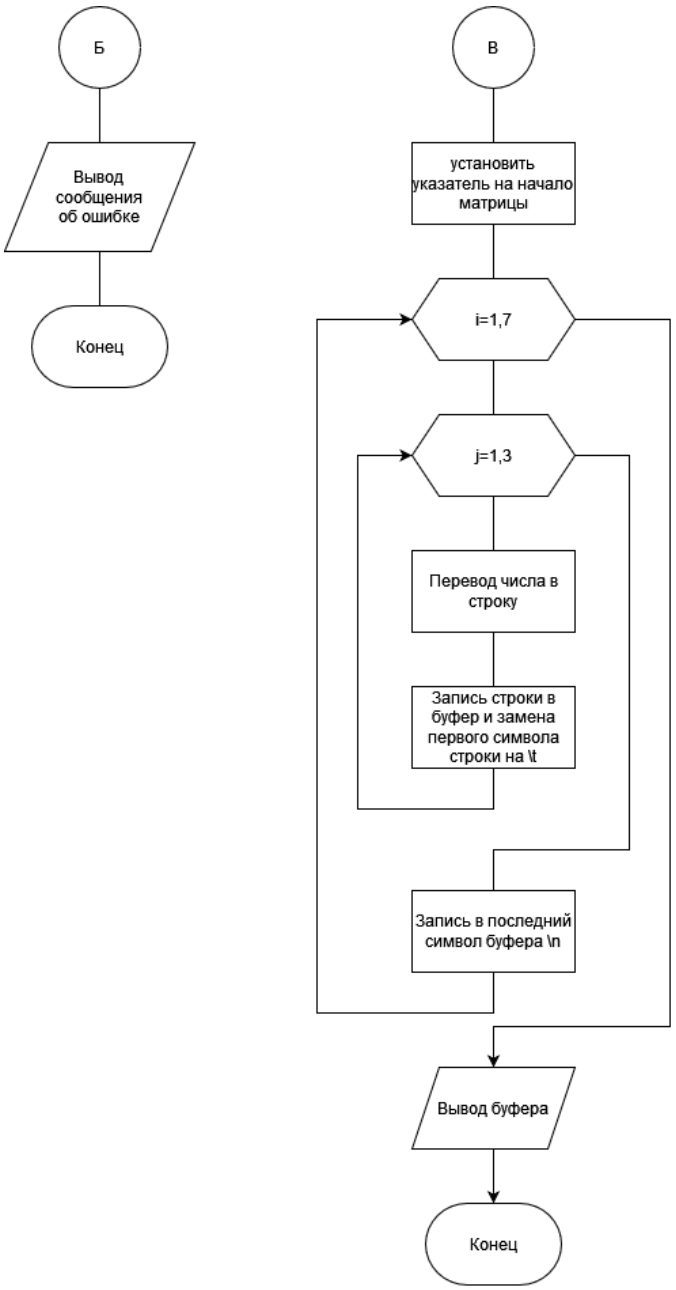

Рисунок 2 – Схема алгоритма

Код программы

Код программы представлен в листинге 1.

Листинг 1 – Код программы

```
; Дана матрица 7x3. Обнулить элементы с четной суммой
индексов. Организовать ввод матрицы и вывод результатов.
    section .data
invite          db     "Enter the 7x3 matrix:",10
invite_len      equ    $-invite
err             db    "Something went wrong due execution",10
```

```
err_len             equ    $-err
matrix_size             equ    21
buffer_len     equ    256
two            dd    2
    section .bss
matrix          resd     matrix_size
buffer          resb    buffer_len
    section .text
global _start
_start:
    ; input invite
    mov    rax,    1
    mov    rdi,    1
    mov    rsi,    invite
    mov    rdx,    invite_len
    syscall
    ; input matrix
    mov    r8,    0    ; input str length
    mov    rcx,    7
    ; input str
    mov    rdi,    0
.inp:    mov    rax,    0
    lea    rsi,    [buffer+r8]
    mov    rdx,    buffer_len
    push    rcx
    syscall
    pop    rcx
    add    r8,    rax
    loop    .inp
    ; replace spaces to 10
    mov    rdi,    buffer
    mov    rcx,    r8
    mov    al,    " "
.rep:    repne    scasb
    mov    byte[rdi-1], 10
    inc    rcx
    loop .rep
    ; fill matrix
    mov    r9,    0    ; index of the current element
    mov    rcx,    r8
    mov    rdi,    buffer
.fill:    mov    rsi,    rdi
    call    StrToInt64
    cmp    rbx,    0
    jne    .err
    mov    dword[matrix+r9*4], 0
```

```
    push    rax
    mov    rax,    r9
    and    rax,    0x1
    cmp    rax,    0
    pop    rax
    je    .skip
    mov    [matrix+r9*4], eax
.skip:    inc    r9
    mov    al,    10
    repne    scasb
    inc    rcx
    loop    .fill
    ; print matrix
    mov    rcx,    7
    mov    rsi,    buffer
    mov    r8,    0
    mov    r9,    0
.exl:    push    rcx
    mov    rcx,    3
.inl:    mov    eax,    [matrix+r9*4]
    call    IntToStr64
    mov    byte[rsi],9
    dec    rax
    add    rsi,    rax
    add    r8,    rax
    inc    r9
    loop    .inl
    mov    byte[rsi],10
    inc    r8
    inc    rsi
    pop    rcx
    loop    .exl
    ; syscall print
    mov    rsi,    buffer
    mov    rdx,    r8
    mov    rax,    1
    mov    rdi,    1
    syscall
    ; exit
    mov    rdi,    0    ; success
.ext:    mov    rax,    60
    syscall
.err:    mov    rax,    1
    mov    rdi,    1
    mov    rsi,    err
    mov    rdx,    err_len
```

```
    syscall
    mov    rdi,    1    ; failure
    jmp    .ext

%include "../lib.asm"
```

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Тестирование

№	Поток ввода	Ожидаемый результат	Поток вывода	Вердикт
1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	0 2 0 4 0 6 0 8 0 10 0 12 0 14 0	0 2 0 4 0 6 0 8 0 10 0 12 0 14 0	Верно
2	-1 -1 -1 0 0 0 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5	0 -1 0 0 0 0 0 1 0 2 0 2 0 3 0 4 0 4 0 5 0	0 -1 0 0 0 0 0 1 0 2 0 2 0 3 0 4 0 4 0 5 0	Верно

Выводы

Изучены механизмы моделирования обработки массивов и матриц в языке ассемблера. Реализована программа ввода, обработки, вывода матрицы.

Контрольные вопросы

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

Понятия массив и матрица в ассемблере не определены, так как и то, и другое – это, по сути, последовательность элементов в памяти.

2) 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.

3) 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 X[EBX*4],EAX” - X[EBX*4] присваивает EAX, EAX умножается на 4 - так как DWORD. “add EBX,1” - EBX = EBX + 1 – смещение указателя на следующий элемент.

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

4) Как в памяти компьютера размещаются элементы матриц?

Элементы матриц в памяти, как и одномерный массив, расположены последовательно.

5) 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.