|  |  |
| --- | --- |
| 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: Программирование обработки массивов и матриц**

Студент гр. ИУ6-41Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. А. Самодурова**

(Подпись, дата) (И.О. Фамилия)

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

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

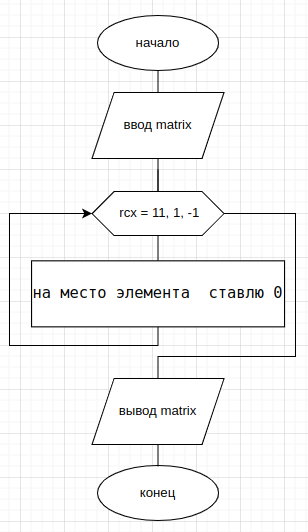
**Вариант 16**

**Цель:** изучение приемов моделирования обработки массивов и матриц в языке ассемблера.

**Текст задания:**

Дана матрица 7х3. Обнулить элементы с четной суммой индексов. Организовать ввод матрицы и вывод результатов.

Схема алгоритма программы представлена на рисунке 1.

Рисунок 1 — схема алгоритма

**Код программы:**

%include "io64.asm"

section .data

InputMsg dq "Input 7\*3 matrix: ",10

lenInput equ $-InputMsg

DevMsg dq "",10

lenDev equ $-DevMsg

err\_line dq "Each line should have exactly 3 numbers divided by spaces", 10

err\_line\_len equ $-err\_line

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

err\_num\_len equ $-err\_num

input times 32 db 0

input\_len equ $-input

matrix times 21 dq 0

A dq 2,3,1

dq 6,-8,5

dq 6,3,1

dq 8,4,7

dq 3,-1,8

dq 3,1,6

dq 8,6,7

section .bss

InBuf resb 10

lenIn equ $-InBuf

X resd 1

res resq 1

section .text

global \_start

\_start:

mov rax, 1

mov rdi, 1

mov rsi, InputMsg

mov rdx, lenInput

syscall

mov rcx,7

xor rdi, rdi

read\_line:

push rcx

push rdi

mov rax, 0 ; системная функция 0 (read)

mov rdi, 0 ; дескриптор файла stdin=0

mov rsi, input ; адрес вводимой строки

mov rdx, input\_len ; длина строки

syscall ; вызов системной функции

pop rdi

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 ;проверка на букву

jne error\_num

cmp rdi, 21

jge error\_line ;;проверка на лишнее число

mov [matrix + 8\*rdi], rax

inc rdi

pop rdx

mov r10, rdx

inc r10

lea rsi, [input + r10]

next:

inc rdx

loop process\_line

pop rcx

mov rax, 3

sub rax, rcx

inc rax

push rdx

mov rdx, 3

mul rdx

pop rdx

dec rcx

cmp rcx, 0

jne read\_line

;основная программа

mov rbx, 0

mov rcx,11

xor rdi, rdi

read:

mov qword[rbx\*8+matrix], 0

add rbx, 2

loop read

;вывод матрицы

mov rcx,7

mov rbx,0

xor rdi, rdi

out\_line:

push rcx

mov rcx,3

out\_num:

mov rax,[rbx\*8+matrix]

mov rsi, InBuf

call IntToStr64

push rcx

; write

mov rdx, rax

mov rax, 1 ; системная функция 1 (write)

mov rdi, 1 ; дескриптор файла stdout=1

syscall

pop rcx

mov rdi, 0

cmp rcx,1

inc rbx

loop out\_num

push rcx

mov rax,1

mov rdi,1

mov rsi, DevMsg ; адрес выводимой строки

mov rdx, lenDev ; длина строки

syscall

pop rcx

pop rcx

loop out\_line

exit:

xor rdi, rdi

mov rax, 60

syscall

error\_line:

mov rax, 1

mov rdi, 1

mov rsi, err\_line

mov rdx, err\_line\_len

syscall

jmp exit

error\_num:

mov rax, 1

mov rdi, 1

mov rsi, err\_num

mov rdx, err\_num\_len

syscall

jmp exit

**Результат работы программы:**

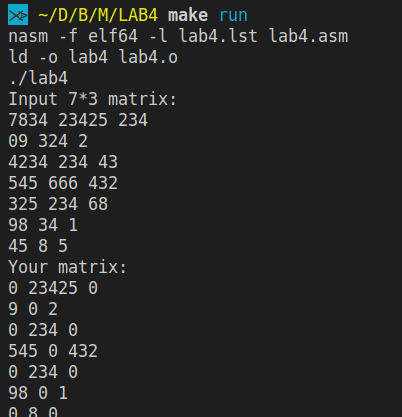
****

Рисунок 2 — результат выполнения

Таблица 1 — проведенные тесты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Ожидаемый результат | Полученный результат |
| 1 | 7834 23425 234  09 324 2  4234 234 43  545 666 432  325 234 68  98 34 1  45 8 5 | 0 23425 0  09 0 2  0 234 0  545 0 432  0 234 0  98 0 1  0 8 0 | 0 23425 0  09 0 2  0 234 0  545 0 432  0 234 0  98 0 1  0 8 0 |
| 2 | 1 1 1  2 2 2  3 3 3  4 4 4  5 5 5  6 6 6  7 7 7 | 0 1 0  2 0 2  0 3 0  4 0 4  0 5 0  6 0 6  0 7 0 | 0 1 0  2 0 2  0 3 0  4 0 4  0 5 0  6 0 6  0 7 0 |
| 3 | 1 2 3  1 2 3  1 2 3  1 2 3  1 2 3  1 2 3  1 2 3 | 0 2 0  1 0 3  0 2 0  1 0 3  0 2 0  1 0 3  0 2 0 | 0 2 0  1 0 3  0 2 0  1 0 3  0 2 0  1 0 3  0 2 0 |

**Контрольные вопросы**

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

Массив и матрица во внутреннем представлении – это последовательность элементов в памяти. В ассемблере понятия «массив» и «матрица» не определены, так как они имеют одинаковое внутреннее представление, не принадлежащие к структуре, полностью контролируются программистом.

2. Как в ассемблере моделируются массивы?

Массив во внутреннем представлении – это последовательность элементов в памяти. В ассемблере такую последовательность можно определить, например, так:

A dw 10,13,28,67,0,-1 ; массив из 6 чисел размером «слово».

Программирование обработки выполняется с использованием адресного регистра, в котором хранится либо смещение текущего элемента относительно начала сегмента данных, либо его смещение относительно начала массива. При переходе к следующему элементу и то, и то смещение увеличивают на длину элемента.

3. Поясните фрагмент последовательной адресации элементов массива?

Почему при этом для хранения частей адреса используют регистры?

Элементы массива размещены в памяти друг за другом. Начало массива выражается смещением. Имея адрес одного, можно получить адрес следующего, добавив размер элемента в байтах. Так, если элементы нумерованы с единицы, то их адрес определяется как:

Aисп = Aначала+(<Номер>-1)\*<Длина элемента>.

Исполнительный адрес считается по формуле:

EA = (База) + (Индекс)\*<Масштаб>+<Смещение>

Полученный по формуле адрес записывается в регистр.

4. Как в памяти компьютера размещаются элементы матриц?

Элементы матриц в памяти компьютера размещаются по строкам или по столбцам. Матрицу можно развернуть в одномерный массив.

5. Чем моделирование матриц отличается от моделирования массивов? В

каких случаях при выполнении операций для адресации матриц используется

один регистр, а в каких – два?

Для инициализации матрицы необходим двойной цикл, в котором будет последовательное увеличение числа строк и изменения индекса в строке, в то время как для инициализации массива достаточно одного цикла и нет зависимости от количества строк.

Для адресации используется один регистр, когда матрица расположена в памяти по строке и просмотр идет по строке. Обработка будет как у массива, один регистр хранит смещение от первого элемента. В другом случае нужно использовать два регистра, в которых будут храниться адрес текущей строки и смещение от начала.

**Вывод:** в ходе выполнения данной лабораторной работы я изучила приемы моделирования обработки массивов и матриц в языке ассемблера.