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)**

О т ч е т

по лабораторной работе № 4

Название: Программирование целочисленных вычислений

Дисциплина: Машинно-зависимые языки и основы компиляции

Студент гр. ИУ6-41Б 01.03.2023 Т. Е. Старжевский
(Подпись, дата) (И.О. Фамилия)

Преподаватель 01.03.2023 С. С. Данилюк
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Введение

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

Задачи работы:

1) Разработать схему алгоритма решения задачи.

2) Написать программу на языке ассемблера, которая вычисляет заданное выражение.

3) Протестировать.

Ход работы

Задание 1

Дана матрица 5x7. Каждый столбец с четным номером матрицы перевернуть так, чтобы первый элемент стал последним, второй — предпоследним и т.д. Организовать ввод матрицы и вывод результатов.

Схема алгоритма показана на рисунке 1:

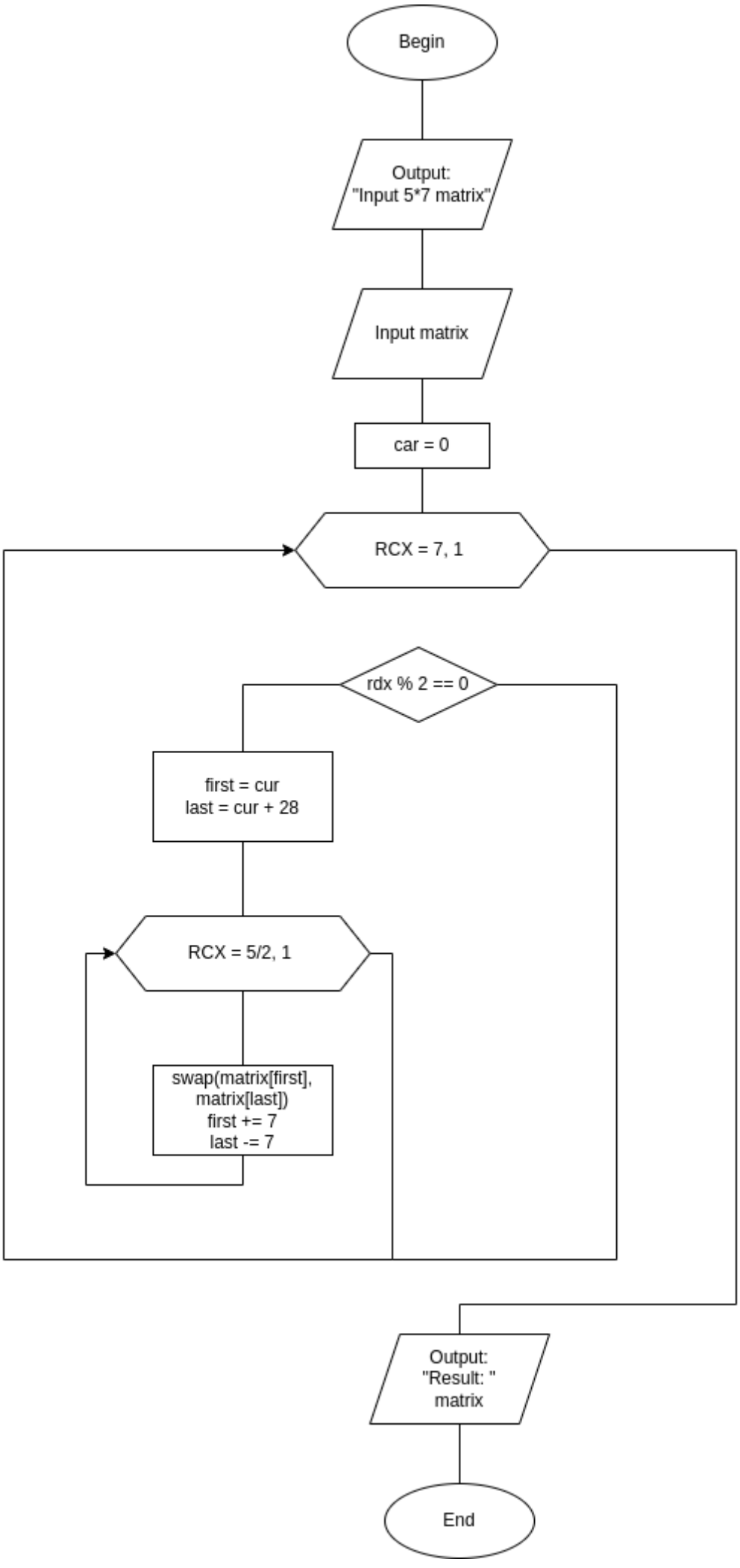
Begin
Output:
"Input 5*7 matrix"
Input matrix
car = 0
RCX = 7, 1
rdx % 2 == 0
first = cur
last = cur + 28
RCX = 5/2, 1
swap(matrix[first],
matrix[last])
first += 7
last -= 7
Output:
"Result: "
matrix
End

Рисунок 1 – схема алгоритма

Задание 2. Код программы:

```
        %include "../lib64.asm"

%define STDIN 0
%define READ 0
%define STDOUT 1
%define WRITE 1
%define EXIT 60

%define ROWS 5
%define COLUMNS 7
%define MATRIX_SIZE 35

section .data
StartMsg db "Enter 5*7 matrix:", 10
StartLen equ $-StartMsg
NewLine: db 0xA
ResultMsg db "Result:", 10
ResultLen equ $-ResultMsg

IncorrectLineMsg db "Each line should have exactly 7 numbers divided by spaces", 10
IncorrectLineLen equ $-IncorrectLineMsg

ErrorSTIMsg dq "Error while transform str to int", 10
ErrorSTILen equ $-ErrorSTIMsg

Space db " "

section .bss
matrix times MATRIX_SIZE resq 1

OutBuf resw 1
lenOut equ $-OutBuf
InBuf resq 10
lenIn equ $-InBuf


section .text
global _start

_start:
mov rax, WRITE
mov rdi, STDOUT
mov rsi, StartMsg
mov rdx, StartLen
```

```
syscall

mov rcx, ROWS
xor rdi, rdi

read_line:
push rcx
push rdi

mov rax, READ
mov rdi, STDIN
mov rsi, InBuf
mov rdx, lenIn
syscall

pop rdi
mov rcx, rax ; Сохраням длину строки
xor rdx, rdx ; Обнуляем регистр
xor r8, r8 ; Обнуляем регистр

process_line:
cmp byte[InBuf + rdx], 10; Если конец строки то обрабатываем число
je process_number

cmp byte[InBuf + rdx], ' '; Если был не конец, и следующий символ
jne next; не пробел, то продолжаем считывание

mov byte[InBuf + rdx], 10; Помещаем вместо проблеа \n
cmp r8, rdx; Если длина строки не совпадает с предыдущей
jne process_number
jmp next; ???

process_number:
push rdx

call StrToInt64; Вход: RSI Выход: RAX, RBX содержит 0 if errors = 0
cmp rbx, 0
jne STIError; Вывод ошибки

mov [matrix + 8 * rdi], rax; Помещаем результат в матрицу
inc rdi; увеличиваем счетчик введенных чисел

pop rdx
mov r8, rdx; Теперь считывать следующее число надо начинать с
inc r8; окончания длины предыдущего
lea rsi, [InBuf + r8]; Передаем указатель на смещенный буфер

next:
inc rdx; Увеличиваем длину числа
loop process_line
```

```
pop rcx
mov rax, ROWS; Проверим количество введеных чисел < 7 в текущей строке
sub rax, rcx
inc rax
push rdx
mov rdx, COLUMNS
imul rdx
pop rdx

cmp rdi, rax; Если введено чисел больше чем длинна строки матрицы
jne IncorrectLine

; loop read_line; Увы здесь не подойдет шорт прыжок
dec rcx
cmp rcx, 0
jnz read_line

; ; logic starts here
mov rcx, ROWS
xor rdx, rdx; Обнуляем подсчет текущей строки

change_row:
test rdx, 1
jz next_row
push rcx
mov rax, COLUMNS; Разделим текущий счетчик на количество столбцов
mov ebx, 2
push rdx
cwd
idiv ebx
pop rdx
mov rcx, rax; Итерировать должны лишь до половины текущей длины строки
mov rax, COLUMNS
imul rax, rdx
add rax, rcx
dec rax; получаем индекс первого элемента, который надо менять
mov rbx, rax
add rbx, 2; Индекс первого с конца

change_column:
mov r8, [matrix + 8 * rax]; Запомнили первый элемент в регистре r8
mov r9, [matrix + 8 * rbx]; Запомнили второй элемент в регистре r9
mov [matrix + 8 * rbx], r8; Поместили первый элемент во второй
mov [matrix + 8 * rax], r9; Поместили второй элемент в первый

dec rax
inc rbx
loop change_column
pop rcx
```

```
next_row:
inc rdx
loop change_row

; end of logic
output:
mov rax, WRITE
mov rdi, STDOUT
mov rsi, ResultMsg
mov rdx, ResultLen
syscall

mov rcx, ROWS
xor rbx, rbx; Обнуляем регистр
output_row:
push rcx
mov rcx, COLUMNS
output_column:
push rcx
mov rsi, OutBuf
mov rax, [matrix + 8 * rbx]
inc rbx
call IntToStr64

mov rax, WRITE; системная функция 1 (write)
mov rdi, STDOUT; дескриптор файла stdout=1
mov rsi, OutBuf ; адрес выводимой строки
mov rdx, lenOut ; длина строки
syscall; вызов системной функции

call PrintSpace

pop rcx
loop output_column

mov rax, WRITE; системная функция 1 (write)
mov rdi, STDOUT; дескриптор файла stdout=1
mov rsi, NewLine ; адрес выводимой строки
mov rdx, 1 ; длина строки
syscall; вызов системной функции

pop rcx
loop output_row

exit:
xor rdi, rdi
mov rax, EXIT
syscall

IncorrectLine:
```

```
mov rax, WRITE
mov rdi, STDOUT
mov rsi, IncorrectLineMsg
mov rdx, IncorrectLineLen
syscall
jmp exit

STIError:
mov rax, 1; системная функция 1 (write)
mov rdi, 1; дескриптор файла stdout=1
mov rsi, ErrorSTIMsg ; адрес выводимой строки
mov rdx, ErrorSTILen ; длина строки
syscall; вызов системной функции
jmp exit
;end

PrintSpace:
mov rax, 1
mov rdi, 1
mov rsi, Space
mov rdx, 1
syscall
ret
```

Результат работы программы представлен на рисунке 2:

```
./lab4
Enter 5*7 matrix:
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
Result:
1 2 3 4 5 6 7
7 6 5 4 3 2 1
1 2 3 4 5 6 7
7 6 5 4 3 2 1
1 2 3 4 5 6 7
timofey@timofey-ASUS:~/Projects/Study/BMSTU_Assembly/lab4$
```

Рисунок 2 - результат работы программы

Задание 3

Тестирование программы показаны на таблице 1:

Таблица 1 – Таблица тестирования

Исходные данные	Ожидаемый результат	Полученный результат
1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7	1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7	1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7
7 6 5 4 3 2 1 7 6 5 4 3 2 1 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7	7 6 5 4 3 2 1 1 2 3 4 5 6 7 1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7	7 6 5 4 3 2 1 1 2 3 4 5 6 7 1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7
7 6 5 4 3 2 1 7 6 5 4 3 2 1 7 6 5 4 3 2 1 7 6 5 4 3 2 1 7 6 5 4 3 2 1	7 6 5 4 3 2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1	7 6 5 4 3 2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1

Вывод: Научился работать с вводом строки произвольной длины и обработкой из нее чисел, разбитием по пробелу. Использовал двойные ветвления для ввода, обработки, вывода матрицы.

Контрольные вопросы:

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

Любые данные в ассемблере — последовательность байтов. В таком случае работаем с ними как хотим, называем как хотим.

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: var word 7,1,2,3,4,5,6 – 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 SDWORD.

“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 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?

Матрицы — последовательно хранящиеся массивы. Сложность лишь в обработке и записи в них, нужно использовать два регистра — один для движения по строкам, второй - по столбцам.