**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 домашней работе № 1

**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 команд обработки цепочек и приемов обработки символьной информации.

**Задачи работы**:

1) Разработать схему алгоритма решения задачи.

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

3) Протестировать.

## Ход работы

Задание 1

Дан текст, состоящий из 7 слов по 5 символов. Удалить слова, содержащие более 3-х букв «О».

Схема алгоритма показана на рисунке 1:

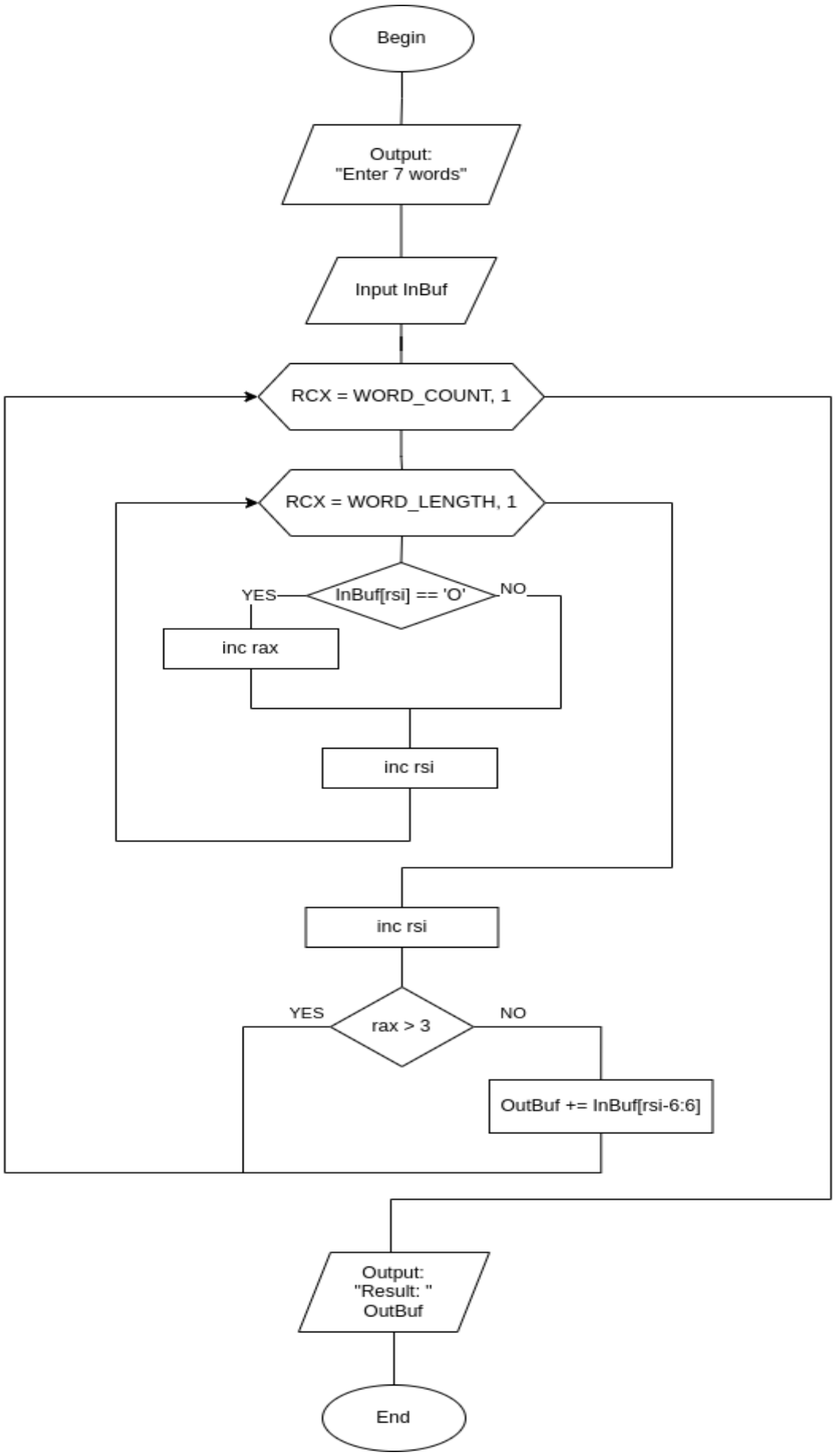


Рисунок 1 – схема алгоритма

Задание 2. Код программы:

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

%define STDIN 0
%define READ 0
%define STDOUT 1
%define WRITE 1
%define EXIT 60
%define WORD_COUNT 7
%define WORD_LENGTH 5

section .data
WRONG_CHAR db "O"

StartMsg db "Enter 7 words: "
StartLen equ $-StartMsg
NewLine db 0xA

ResultMsg db "Result: "
ResultLen equ $-ResultMsg

section .bss
char resb 1

OutBuf resb 41
lenOut equ $-OutBuf

InBuf resb 41
lenIn equ $-InBuf


section .text
global _start

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

; OOOOO aOOOa bOOOs OadOO OOOOO asdcd asdac
read_line:
mov rax, READ
mov rdi, STDIN
mov rsi, InBuf
mov rdx, lenIn
syscall
; logic
lea rdi, OutBuf; Сохранять будем в буфер
```

```
mov rcx, WORD_COUNT
check_word:
push rcx
xor rax, rax; Обнуляем счетчик ошибок
mov rcx, WORD_LENGTH
check_char:
cmp byte[rsi], 'O'
jne next_char
inc rax
next_char:
inc rsi
loop check_char

inc rsi; Скипаем пробел
cmp rax, 3; Если ошибок больше трех
jg next_word; То скипаем слово, иначе копирование
sub rsi, 6; Возвращаемся на 6 символов назад
mov rcx, 6; Считаем 6 раз
rep movsb; Скопируем слово в буфер
add rdx, 6; Увеличиваем сдвиг текущего буфера
next_word:
pop rcx
loop check_word
; end logic

output:
mov rax, WRITE; системная функция 1 (write)
mov rdi, STDOUT; дескриптор файла stdout=1
mov rsi, ResultMsg
mov rdx, ResultLen ; длина строки
syscall; вызов системной функции

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


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

exit:
xor rdi, rdi
mov rax, EXIT
syscall
```

Результат работы программы представлен на рисунке 2:

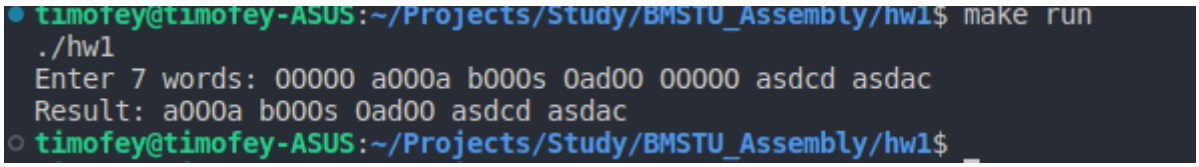


Рисунок 2 - результат работы программы

Задание 3

Тестирование программы показаны на таблице 1:

Таблица 1 – Таблица тестирования

| Исходные данные | Ожидаемый результат | Полученный результат |
|---|---|---|
| OOOOO aOOOa bOOOs OadOO OOOOO asdcd asdac | aOOOa bOOOs OadOO asdcd asdac | aOOOa bOOOs OadOO asdcd asdac |
| OOOOO aOOOa bOOOs OadOO OvOkO asdcd asOac | aOOOa bOOOs OadOO OvOkO asdcd asOac | aOOOa bOOOs OadOO OvOkO asdcd asOac |
| Abcde abcde abcde abcde abcde abcde | Abcde abcde abcde abcde abcde abcde | Abcde abcde abcde abcde abcde abcde |

**Вывод**: изучил функции работы со строками, такие как movsb, научился обрабатывать элементы в строке, легко писать циклические участки кода с несколькими условиями.

**Контрольные вопросы:**

**1. Дайте определение символьной строки.**

Символьная строка – последовательность байт.

**2. Назовите основные команды обработки цепочек?**

Команды обработки цепочек:

– movs – для пересылки из одной области памяти в другую;

– cmps – сравнение элементов цепочки-источника с элементами цепочки-приемника;

– scas – поиск некоторого значения в области памяти;

– lods – позволяет извлечь элемент из цепочки и загрузить регистр-аккумулятор eax/al/ax;

– stos – позволяет сохранить элемент из регистра-аккумулятора al/ax/eax в цепочке.

**3. Какие операции выполняют строковые команды MOVS? Какие особенности характерны для этих команд?**

Транслятор преобразует команду movs в одну из трех машинных команд movsb, movsw и movsd. Та или иная команды выбирается от того, данные какого размера перемещала команды movs. Сама команда перемещает один элемент, но если нужно больше, то используются префиксы, такие как rep (работает до тех пор, пока ECX не станет равно 0).

**4. Какие операции выполняют строковые команды CMPS, SCAS? Какие особенности характерны для этих команд?**

CMPS – команда сравнения элементов цепочек-источника с подстроками, то есть цепочки-приемника. После выхода из цикла, регистры содержат адреса элементов после тех, которые стали причиной выхода

SCAS – команда поиска некоторого значения в области памяти. Имеет один операнд, обозначающий местонахождение цепочки, который и нужно найти.

**5. Как обеспечить циклическую обработку строк?**

С помощью префиксов повторения:

– rep – работает до тех пор, пока EAX не станет равной 0, используется с movs и stos;

– repe и repz – работает до тех пор, пока EAX не станет равной 0 или zf не станет равным “1”, используется с cmps и scas;

– repne и repnz – работает до тех пор, пока EAX не станет равной 0 или zf не станет равным “0”, используется с cmps и scas.

**6. Какова роль DF во флаговом регистре при выполнении команд обработки строк?**

Он будет указывать откуда начнется обработка строк. Если DF = 1, то с начала, если DF = 0, то с конца. Командами “cld” можно установить DF равный 0, а “std” DF равный 1

**7. Какие макрокоманды используются в среде RADASM для вводов и вывода строк?**

StdOut - вывод, StdIn - ввод, StripLF – замена символа конца строки нуляем, atol – преобразования строки в число, dwtoa – преобразования числа в строку.

**8. Как правильно выбрать тестовые данные для проверки алгоритма обработки строки?**

Надо выбрать те данные, которые проходили бы через весь код программы. Также можно ввести несколько частных случаев проверяющие конкретные мета программы на правильность её выполнения.