«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_____

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по домашней работе № 1

Дисциплина: Машинно-зависимые языки и основы компиляции

Название лабораторной работы: Обработка символьной информации

Вариант: 17

Студент гр. ИУ6-43Б **27.03.2022** **М.А. Мяделец**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ **М.В. Широкова**
(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Обработка символьной информации

Цель работы:

Задание

Рассмотрим задание для домашнего задания 1.

Дан текст 8 слов, разделенных пробелом. Определить количество повторений буквы E в каждом слове.

Краткие моменты решения:

1) Для начала работы необходимо определиться с максимальной длиной вводимой строки (InBuf) и количеством символов строчки, в которую будем временно записывать количество символов 'E' в i-том слове (countE).
2) Просим пользователя ввести строчку из 8 слов, если слов больше – лишние будут проигнорированы, если меньше – количество 'E' в недостающих будет нулевым.
3) После ввода строчки, программа найдет первый пробел с помощью команд обработки строк repne scasb. По завершении выполнения регистр EDI будет указывать на следующее слово.
4) Посимвольно итерируемая по текущему слову через регистр ESI и при нахождении символа 'E' увеличиваем EAX на 1 (цикл-пока count).
5) Когда регистр ESI станет равным EDI, т.е. произойдет переход на следующее слово. Программа выйдет из цикла count и выведет количество символов 'E' в текущем слове.
6) Повторяем пункты 3-5 в счетном цикле cycle для каждого из 8-ми слов.

Иллюстрация алгоритма

Рассмотрим для примера строчку из 28 символов. Изначально регистры ESI и EDI указывает на ее начало (смотри рисунок 1).

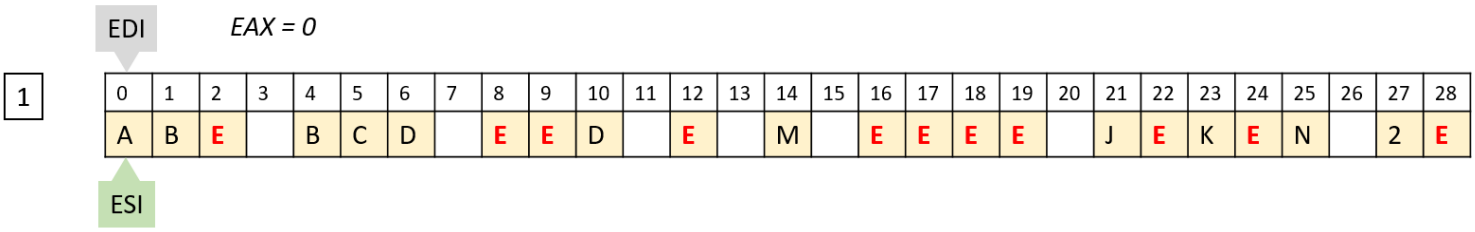

Рисунок 1 – Начальная позиция регистров

Далее занесем в регистр EAX символ пробела и запустим команду repne scasb, после которой регистр EDI будет указывать на следующее слово (смотри рисунок 2).

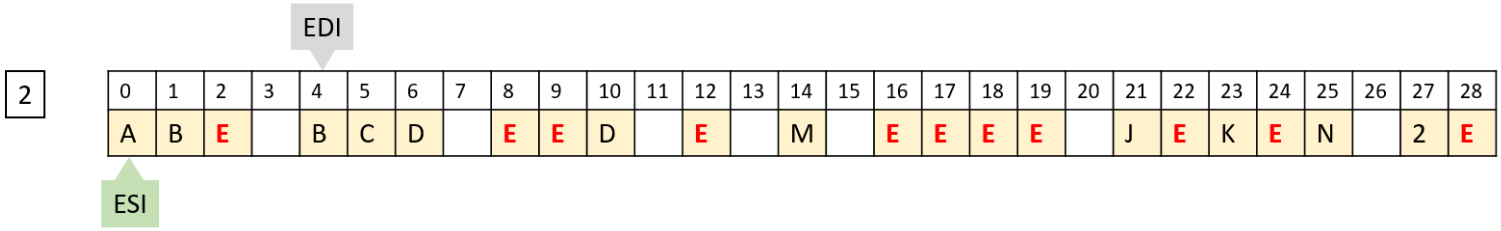

Рисунок 2 – Команда repne scasb

Далее посимвольно пройдемся по строке регистром ESI. Если текущий символ 'E', увеличим EAX на 1 (смотри рисунок 3).

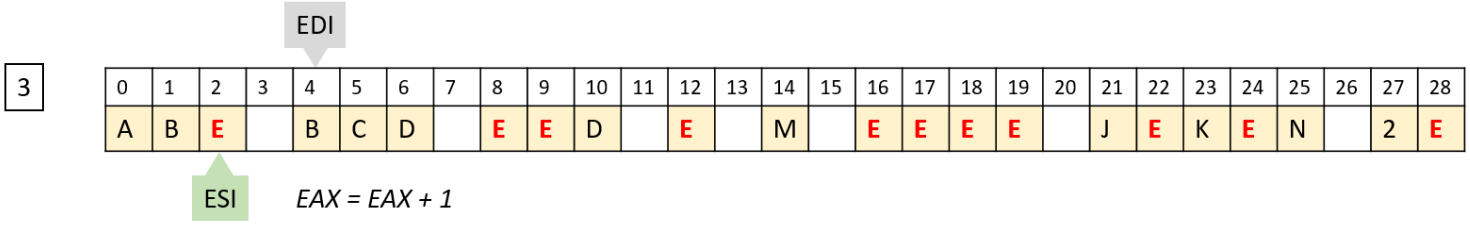

Рисунок 3 – Обнаружение символа 'E'

Условием завершения прохода по слову будет равенство регистров ESI и EDI. Тогда происходит выход из цикла и вывод в консоль значения EAX (смотри рисунок 4).

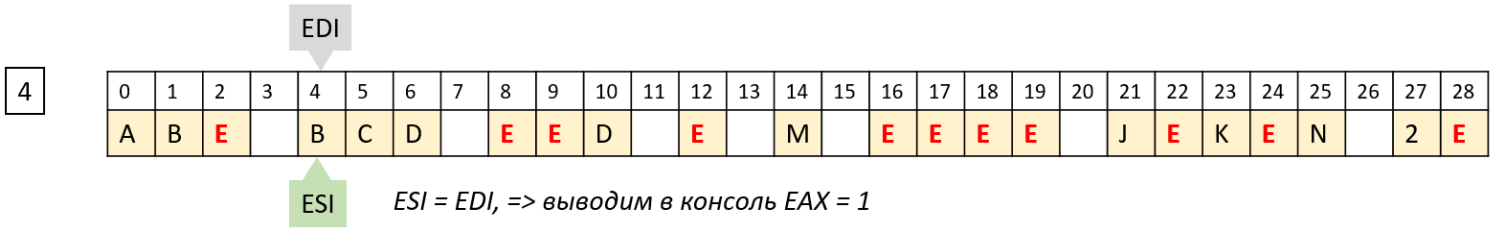

Рисунок 4 – Завершение прохода по слову

Блок-схема алгоритма

Для визуализации работы программы была спроектирована блок-схема алгоритма. Схема наглядно и упрощенно показывает все этапы работы алгоритма (смотри рисунок 5).

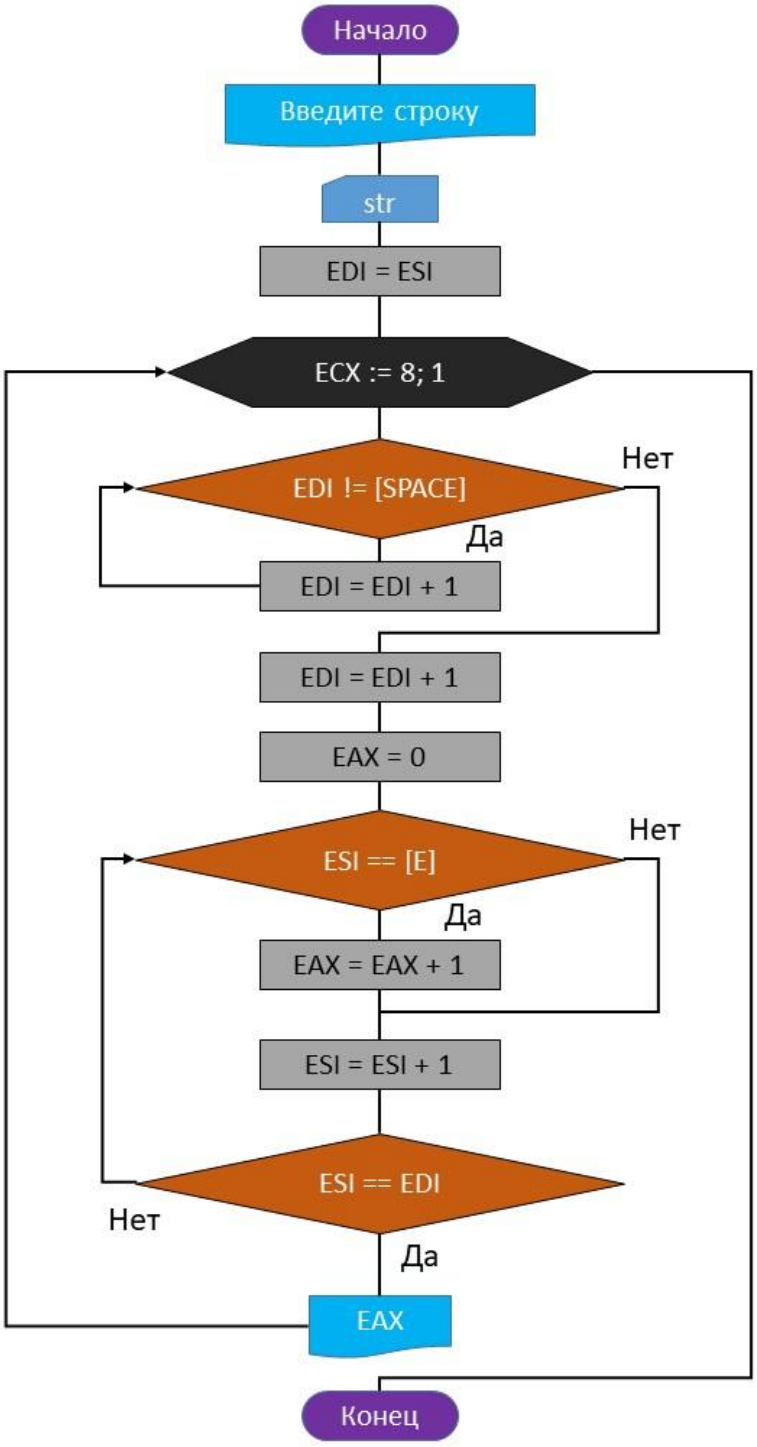

Рисунок 5 – Блок-схема алгоритма

Тестирование

Тест 1. Пример из раздела «Иллюстрация алгоритма» (смотри рисунок 6).

Рисунок 6 – Корректный ввод строчки

Тест 2. Введено больше 8-ми слов (смотри рисунок 7).

Рисунок 7 – Некорректный ввод строчки.

Тест 3. Введено меньше 8-ми слов (смотри рисунок 8).

Рисунок 8 – Некорректный ввод строчки

Вывод: было сделано домашнее задание 1, которое было направлено на изучение команд обработки строк. В ходе выполнения задания были применены различные приемы работы со строками: через операцию repne scasb для поиска следующего пробела и через «ручной» посимвольный проход, каждый раз сравнивая текущий символ с символом 'E' для увеличения счетчика и с началом нового слова для вывода общего количества символов 'E' текущего слова и перехода на следующее слово. Также были обработаны все граничные случаи, чтобы программа работала корректно при любых входных данных. Результаты тестирования корректны. Работа проведена успешно!

Контрольные вопросы

1) *Дайте определение символьной строки.*

Символьная строка – последовательность байт.

2) *Назовите основные команды обработки цепочек?*

Основные операции работы с цепочками: movs, movsb, movsw, movsd, cmps, cmpsb, cmpsw, cmpsd, scas, scasb, scasw, scasd, lods, lodsb, lodsw, lodsd, stos, stosb, stosw, stosd.

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

MOVS – команды пересылки цепочки. Особенность команд: сами по себе команды пересылают только один элемент, исходя из его типа, и модифицирует значения регистров esi/si и edi/di. А если перед командой написать префикс rep, то одной командой можно передавать до 4 ГБ данных.

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

CMPS – команды сравнения цепочек, SCAS – команды сканирования цепочки. Их общая особенность заключается в том, что перед ними так же используются префиксы повторения.

5) *Как обеспечить циклическую обработку строк?*

Для работы с циклической обработкой строк необходимо использовать префиксы повторения: rep, repe/repz, repne/repnz.

6) *Какова роль флага DF во флажковом регистре при выполнении*

команд обработки строк?

Ответ: Флаг DF (флаг направления) необходим для управления направлением обработки цепочек (в направлении возрастания или убывания адресов).

7) *Как правильно выбрать тестовые данные для проверки алгоритма обработки строки?*

Тесты должны проверять все граничные случаи и потенциальные ошибки.

ПРИЛОЖЕНИЕ

Текст программы

```
        section .data
message db "Enter the string", 10
lenMsg  equ $-message
SPACE   db " "
E       db "E"

        section .bss
InBuf   resb    100             ; максимальная длина строки
lenIn   equ     $-InBuf
countE  resb    3

        section .text
        global  _start

_start:

        ; вывод сообщения "Enter the string"

        mov     eax, 4
        mov     ebx, 1
        mov     ecx, message
        mov     edx, lenMsg
        int     80h

        ; ввод строчки

        mov eax, 3
        mov ebx, 0
        mov ecx, InBuf
        mov edx, lenIn
        int 80h



        mov esi, ecx
        mov edi, ecx    ; пересылаем строчку в esi и edi
```

```
        mov ecx, 8      ; счетчик слов


cycle:
                push ecx
                mov ecx, lenIn

                movzx eax, byte[SPACE]
                repne scasb             ; в edi - адрес следующего пробела

                pop ecx
                mov eax, 0              ; счетчик буквы 'E'

                movzx ebx, byte[E]      ; ebx содержит символ 'E'
count:
                        cmp byte[esi], bl       ; сравниваем текущий символ
с 'E'
                        jne not_e

                        inc eax                 ; если символ = 'E', то
счетчик увеличиваем на 1

not_e:
                        inc esi                 ; переходим на следующий
символ
                        cmp esi, edi            ; сравниваем с edi, который
указывает на конец слова (пробел)
                        jne count               ; если не конец слова,
продолжаем считать "E"


                ; если пробел

                push ecx                ; stack <
                push edi                ; stack <

                ; выводим количество букв 'E' в слове под номером ECX
                ; в eax содержится кол-во 'E' в слове
```

```
                mov esi, countE
                call IntToStr
                mov edi, eax    ; количество символов в строке "<EAX>"

                mov eax, 4
                mov ebx, 1
                mov ecx, esi
                mov edx, edi
                int 80h

                pop edi                 ; stack >
                pop ecx                 ; stack >

                ; inc edi              ; был на пробеле, переходим на
следующий символ (на след. слово)
                mov esi, edi        ; оба указывают на новое слово

                loop cycle

        ; exit
        mov     eax, 1          ; системная функция 1 (exit)
        xor     ebx, ebx        ; код возврата 0
        int     80h             ; вызов системной функции

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