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 домашней работе № 1

Дисциплина: МЗЯиОК

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

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

Москва, 2024

Цель работы

Изучение команд обработки цепочек и приемов обработки символьной информации.

Задание

Дан текст 26 символов. Определить количество различных символов и частоту их повторений.

Схема алгоритма

Схема алгоритма работы программы представлена на рисунке 1.

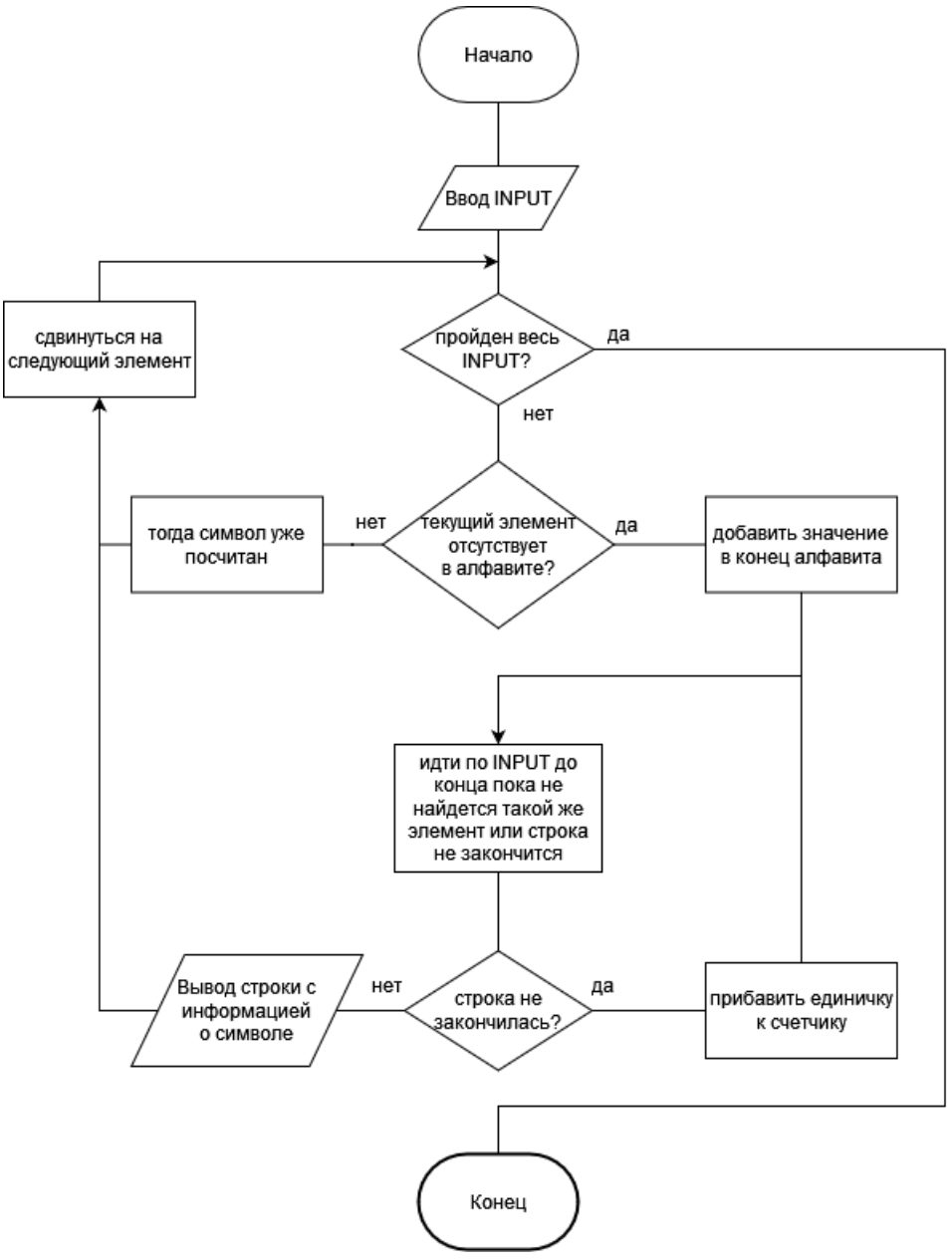

Рисунок 1 – схема алгоритма

Код программы

Ниже приведен листинг программы.

```
        section .data
ALPH_LEN dd 0    ; количество символов, записанных в буфер
OUTPUT_CHAR_INFO db "  -"
OUTPUT_CHAR_INFO_LEN equ $-OUTPUT_CHAR_INFO
INVITE db "Enter the text (26 characters):",10
INVITE_LEN equ $-INVITE
INT_TO_STR_DIST times 10 db " "
RESULT_CHAR_AMOUNT_INFO db " different charaters ",10
RESULT_CHAR_AMOUNT_INFO_LEN equ
$-RESULT_CHAR_AMOUNT_INFO
   section .bss
INPUT resb 26    ; буфер для входной строки
ALPH resb 26    ; буфер для символов, которые встретились входной
строке
   section .text
global _start
_start:
   ; приглашение на ввод
   mov        rax, 1                ; fun=write
   mov        rdi, 1                ; stdout=1
   mov        rsi, INVITE           ; string
   mov        rdx, INVITE_LEN    ; string length
   syscall                          ; system call
   ; ввод входной строки
   mov        rax, 0                ; fun=read
   mov        rdi, 0                ; stdin=0
```

```
    mov         rsi, INPUT          ; dist
    mov         rdx, 26             ; dist length
    syscall
    ; преамбула внешнего цикла
    cld                             ; направление прямое
    mov         rcx, 26             ; количество символов в буфере
    mov         rsi, INPUT          ; указываем буфер с входной строкой
cycle1: ; внешний цикл. обход всех элементов
    cmp         rcx, 0              ; проверяем, что еще не закончились
символы
    je          ed                  ; если закончились, то выходим
    dec         rcx                 ; уменьшаем счетчик оставшихся
элементов
    lodsb                           ; загружаем байт из последовательности
    ; проверка, что элемент в аккумуляторе не встречался раньше
    push        rcx                 ; запоманием значение для внешнего
цикла
    mov         ecx, [ALPH_LEN]     ; сколько уже было запомнено символов
    lea         rdi, [ALPH]         ; указываем на буфер алфавита
    repne scasb                     ; ищем такой же элемент
    pop         rcx                 ; возврат значение для внешнего цикла
    je          cycle1              ; символ уже есть в алфавите
    ; найден новый символ
    ; записываем его в алфавит
    push        rdx                 ; занимаем регистр для вычисления адреса
    lea         rdx, [ALPH]         ; вычисляем место в алфавите для символа
    add         edx, [ALPH_LEN] ; начало буфера + смещение внутри
    mov         [rdx], al           ; запоминаем символ в алфавит
```

```
    pop         rdx                 ; освобождаем
    inc         DWORD[ALPH_LEN]     ; увеличиваем счетчик запомненных букв
    ; считаем, сколько раз он встречается в последовательности
    push        rcx                 ; запоманием значение для внешнего цикла
    push        rsi                 ; запоминаем значение для внешнего цикла
    mov         rdx, 0              ; в регистр rdx запоминаем, сколько раз встречался текущий элемент
    mov         rdi, rsi        ;       двигаем в регистр для команды scasb
cycle2: ; цикл поиска одинаковых элементов
    repne   scasb                   ; ищем элемент такой же элемент
    inc         rdx                 ; увеличиваем счетчик найденных одинаковых элементов
    ; 2 джампа понадобились, тк нет прыжка по отрицанию пустоты ecx
    jecxz       print_char_info     ; если всю последовательность обошли, то уходим
    jmp         cycle2              ; иначе повторяем, пока не дойдем до конца
print_char_info: ; выводим информацию о количестве вхождений символа
    ; подготовка буфера OUTPUT_CHAR_INFO
    mov         [OUTPUT_CHAR_INFO], al  ; на первое место записываем текущий символ
    mov         rax, rdx            ; записываем количество вхождений в требуемый регистр (затирается текущий символ!)
    lea         rsi, [INT_TO_STR_DIST]  ; указываем буфер, куда будем переводить число
```

```
    call IntToStr64
    inc         rsi                 ; без первого символа
    dec         rax                 ; --
    mov         rcx, rax            ; количество символов, которые нужно
записать
    lea         rdi, [OUTPUT_CHAR_INFO+4] ; будем копировать цифры в
буфер с 4 позиции (после тире и пробела)
    rep movsb
    ; вывод буфера
    add         rax, 4              ; 4 символа еще занимают начало буфера
    mov         rdx, rax            ; муваем в требуемый регистр
    lea         rsi, [OUTPUT_CHAR_INFO]    ; буфер для вывода
    mov         rax, 1              ; fun=write
    mov         rdi, 1              ; stdout=1
    syscall
    ; уходим на начало цикла
    pop         rsi                 ; восстанавливаем значение для внешнего
цикла
    pop         rcx                 ; восстанавливаем значение для внешнего
цикла
    cmp         rcx, 0              ; а не равняется ли счетчик нулю
    jne         cycle1              ; возврат на начало цикла, если неверно
; вывод количества различных символов
ed:    mov      eax, [ALPH_LEN]
    lea         rsi, [INT_TO_STR_DIST]
    call IntToStr64
    sub         eax, 1              ; на один символ меньше (\n не берем)
```

```
    xor         rdx, rdx                ; чистим регистр, чтобы в старших
разрядах ничего не было
    mov         edx, eax                ; муваем в требуемый регистр
    mov         rax, 1                  ; fun=write
    mov         rdi, 1                  ; stdout=1
    syscall
    ; вывод заверщающего сообщения
    mov         rsi, RESULT_CHAR_AMOUNT_INFO       ; string
    mov         rdx, RESULT_CHAR_AMOUNT_INFO_LEN    ; string length
    mov         rax, 1                  ; fun=write
    mov         rdi, 1                  ; stdout=1
    syscall
    ; завершение программы
    mov         rax, 60                 ; fun=exit
    xor         rdi, rdi                ; return code 0
    syscall

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

Тестовые данные

Тестовые наборы данных, а также выходные наборы данных и вердикт представлены в таблице 1.

Таблица 1 – тестирование программы

№	Входная строка	Результат	Вердикт
1	abcdefghijklmnopqrstuvwxyz	a - 1 b - 1 c - 1 d - 1 e - 1 f - 1 g - 1	Верно

		h - 1 i - 1 j - 1 k - 1 l - 1 m - 1 n - 1 o - 1 p - 1 q - 1 r - 1 s - 1 t - 1 u - 1 v - 1 w - 1 x - 1 y - 1 z - 1 26 different charaters	
2	abcdefghijklmnopqrstuvzzzz	a - 1 b - 1 c - 1 d - 1 e - 1 f - 1 g - 1 h - 1 i - 1 j - 1 k - 1 l - 1 m - 1 n - 1 o - 1 p - 1 q - 1 r - 1 s - 1 t - 1 u - 1	Верно

		v - 1 z - 4 23 different charaters	
3	aaaaaaaaaaaaaaaaaaaaaaaaa	a - 26 1 different charaters	Верно
4	aaaaAAAAaaaaAAAAaaaaAAA Aaa	a - 14 A - 12 2 different charaters	Верно

Вывод

Были изучены команды обработки цепочек и приемы обработки символьной информации. Написана программа, реализующая обработку строки при помощи специализированных команд. Проведено тестирование и отладка написанной программы.

Ответы на контрольные вопросы

1. Дайте определение символьной строки.

Символьная строка – непрерывная последовательность байтов.

2. Назовите основные команды обработки цепочек?

movs, cmps, lods, scas, stos и их производные для различных типов, ret и ее производные для различных условий.

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

Данная команда производит перезапись 1 элемента цепочки из источника в приемник. Команде movs не сущесвует машинного аналога, вместо нее при ассемблировании подставляется команда для работы с элементами строки конкретного размера (байт, слово, двойное слово).

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

cmps – сравнение 1 элемента строк, scas – сканирование цепочки на поиск элемента, равного значению в аккумуляторе.

5. Как обеспечить циклическую обработку строк?

С помощью команд rep, repe, repne.

6. Какова роль флага DF во флажковом регистре при выполнении команд обработки строк?

Флаг DF обозначает направление обработки. DF=0 – прямой порядок (от младших байт к старшим), DF=1 – наоборот.

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

Необходимо взять такие наборы данных, чтобы был протестирован каждый оператор программы.