МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра Математического обеспечения электронно-вычислительных машин

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Организация ЭВМ и систем»

ТЕМА: ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА СИМВОЛЬНОЙ ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ СТРОКОВЫХ КОМАНД.

Вариант 13

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить принципы обработки символьной информации с использованием строковых команд на языке Ассемблер. Написать программу на языке высокого уровня со вставкой ассемблерного кода по принципу in-line.

Задание.

Разработать программу обработки символьной информации, реализующую функции:

- инициализация (вывод титульной таблички с указанием вида преобразования и автора программы) - на ЯВУ;

- ввода строки символов, длиной не более Nmax (<=80), с клавиатуры в заданную область памяти - на ЯВУ; если длина строки превышает Nmax, остальные символы следует игнорировать;

- выполнение заданного в таблице 5 преобразования исходной строки с записью результата в выходную строку - на Ассемблере;

- вывода результирующей строки символов на экран и ее запись в файл - на ЯВУ. Ассемблерную часть программы включить в программу на ЯВУ по принципу встраивания (in-line).

Вид преобразования, соответствующий 13 варианту заданий:

Формирование номера введенной русской буквы по алфавиту и номера позиции его первого вхождения во входной строке и выдача их на экран.

Замечания:

1) При выполнении преобразования обязательно использовать команды работы со строками;

2) При выполнении преобразования нельзя портить входную строку. Результат преобразования должен записываться в выходную строку.

Ход выполнения.

В ходе работы была разработана программа на языке *C++* со встраиванием ассемблерного кода по принципу *in-line.*

Объявим, инициализируем необходимые переменные: *char input[81]* под входную строку, *int num = -1* для счетчика номеров букв в алфавите на каждой итерации поиска, *int arr[33]={0}* под массив номеров позиций первого вхождения букв русского алфавита, *int len* – для длины входной строки, понадобится в дальнейшем для *ecx* – счетчика повторений выполнения команды *scasb* с префиксом repne.

Выведем строку «инициализации работы» с указанием автора и вида преобразования строки: *std::cout<<"*Автор: Рубежова Наталия ст.гр.0382 *n*Формирование номера введенной русской буквы по алфавиту и номера позиции его первого вхождения во входной строке и выдача их на экран.*n*Введите строку: *"*

Считываем входную строку *cin.getline(input, 81).* И вычислим ее длину *len = strlen(input)*, она нам понадобится в дальнейшем для счетчика повторений *ecx* в цикле поиска символа в строке.

Преобразование будем осуществлять посредством разработки ассемблерного кода и вставки его блока по принципу in-line: *__asm*{ }.

Будем предполагать, что нумерация букв в алфавите начинается с единицы, также будем считать, что позиции букв в слове тоже начинаются с 1.

Для поиска символа во входной строке в дальнейшем будем использовать строковую команду *SCASB,* которая сравнивает элемент строки, адрес которого задается парой *ES:DI*, со значением регистра *AL* и результат сравнения фиксирует в флагах, после чего увеличивает *DI* на 1. Поэтому мы должны установить *edi* на смещение *input* входной строки. Перед строковой командой *SCASB* поставим префикс повторения *REPNE,* который устанавливает *ZF* в 0 и, постоянно уменьшая *CX,* заставляет многократно повторяться эту команду до момента, когда *ZF* установится в 1 или *CX* не достигнет нуля. В регистр *ECX* записываем значение длины строки, вычисленное раннее, *len* – именно столько раз выполнится *SCASB* – сравнение элемента строки с искомым символом.

Краткое описание реализуемого алгоритма: так как мы будем искать во входной строке буквы русского алфавита, то мы должны организовать цикл

перебора букв русского алфавита в *AL* и на каждой итерации запускать поиск заданного символа во входной строке и увеличивать значение переменной-счётчика *num,* таким образом на каждой итерации *num* будет отвечать за номер буквы в алфавите, которую мы ищем. Если заданный символ найден во входной строке, то обращаемся к *arr[esi*4],* где *esi* предварительно присваивается значение *num,* и в эту ячейку памяти будем записывать индекс первого вхождения буквы в строку(умножаем на 4, так как *arr* – целочисленный массив и каждый элемент занимает 4 байта). Если заданный символ не найден, переходим к следующей букве алфавита. После выполнения алгоритма получим массив *arr[],* в котором каждому элементу с индексом-номером буквы русского алфавита соответствует число - индекс первого вхождения этой буквы во входной строке. Если массив заполнен нулями, т.е. русских букв не встретилось, то выводим «Буквы кириллицы отсутствуют в введенной строке»

После каждого выполнения *SCASB* значения *ECX* и *ES:DI* будут «искажаться», отличаться от первоначальных, указывающих на начало строки и длину строки. Поэтому на каждой итерации цикла нужно их «восстанавливать» с помощью *move di, offset input* и *move ecx, len.*

Перебирать символы алфавита будем, инкрементируя *INC AL(*для символов а-е), а так как ASCII-код символа ‘ё’ «обособлен» от остальных, то после обработки и поиска буквы е, мы должны вручную установить в *AL* символ ‘ё’, а после его обработки вернуться к символу ‘ж’, который в таблице *ASCII* идет сразу за ‘е’. Для этого используем команды условного перехода и метки.

Не забываем на каждой итерации увеличивать на 1 значение *num,* которое отвечает за номер буквы алфавита, которая ищется в строке на данной итерации.

Выполняем поиск символа из *AL* в строке с помощью *SCASB(*принцип работы команды описан на четыре абзаца выше*).* Если *ecx != 0*, значит, не все символы строки перебрались и где-то встретилась искомая буква. Используя

команду условного перехода, перейдем к метке *get_index,* где перейдем к вычислению индекса первого вхождения. Если же *ecx* стал равен 0, значит, все символы в строке перебрались. Но возможны два варианта: все символы перебрались и не нашли искомого символа, все символы перебрались и последний символ оказался искомым. Поэтому мы должны сравнить последний символ с искомым с помощью *cmp.* В случае эквивалентности – нашли искомый символ и переходим к вычислению индекса первого вхождения по метке *get_index.* В ином случае – искомый символ в строке отсутствует, можем перейти к поиску следующей буквы. Всегда перед тем, как перейти к поиску следующей буквы, необходимо проверить, если текущая буква поиска оказалась символом ‘я’ – значит, все буквы русского алфавита перебраны, входная строка обработана, можем выводить результат.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	вfghгwsqдnvzж	3 1 4 5 5 9 8 13	Результаты корректны
2.	абвгд	1 1 2 2 3 3 4 4 5 5	Результаты корректны
3.	fhdjs	Буквы кириллицы отсутствуют в введенной строке.	Результаты корректны

Выводы.

В результате работы была изучена обработка символьной информации с использованием строковых команд на языке Ассемблер посредством разработки программы, формирующей номер введенной русской буквы по алфавиту и номера позиции ее первого вхождения во входной строке и выводящей их на экран.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
ASSUME CS:CODE, DS:DATA, SS:AStack
#include <iostream>

using namespace std;

char input[81];
int num = -1; //счётчик-номер буквы в алфавите
int arr[33]={0}; //массив номеров позиций первого вхождения буквы
русского алфавита во входной строке
int len;

int main() {
    system("chcp 1251 > nul");
    setlocale(LC_CTYPE, "rus");
    cout << "Автор: Рубежова Наталия ст.гр.0382 \nФормирование номера
введенной русской буквы по алфавиту и номера позиции его первого
вхождения во входной строке и выдача их на экран.\nВведите строку: ";

    cin.getline(input, 81);
    len = strlen(input);
     __asm {
            push ds
            pop es

            mov al,'a'
            dec al
        cycle:
            mov edi, offset input
            mov ecx, len
            cmp al, 'ё'
            je ret_to_zh
            cmp al, 'e'
            je symbol_yo
            inc al
        label :
            inc num
            repne scasb
        has_symbol_check :
            cmp ecx,0
            jne get_index
            dec edi
            cmp ES:[edi],al
            je get_index
            jmp last_iteration_check
        get_index:
            mov ebx, len
            sub ebx, ecx //ebx теперь содержит индекс первого
вхождения
            mov esi, num
            mov ES:arr[esi*4],ebx
            jmp last_iteration_check
        last_iteration_check:
```

```
            cmp al, 'я'
            je the_end
            jmp cycle

        symbol_yo:
            mov al,'ё'
            jmp label
        ret_to_zh:
            mov al, 'ж'
            jmp label

        the_end:
     };
    int flag = 0;
    for (int i = 0; i < 33; i++) {
        if (arr[i] != 0) {
            cout << i + 1 << ' ' << arr[i] << endl;
            flag = 1;
        }
    }
    if (flag == 0)
        cout << "Буквы кириллицы отсутствуют в введенной строке.";
    return 0;
}
```