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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Организация ЭВМ и систем»
Тема: Представление и обработка символьной информации с использованием строковых команд

Студент гр. 1381 _____ Луценко Д. А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить представление и обработку строк информацией на языке Ассемблера.

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 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).

Вариант 13:

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

Выполнение работы.

Создаем глобальные переменные:

char input[81] - массив элементов \ для входной строки.

int letters[66] = {0} – массив элементов, в котором будут хранится номер первого вхождения буквы.

int counter – счетчик букв для letters.

Считывание строки происходит на языке C++ . Ассемблерная часть Просто включается в программу. Для нахождения вхождения буквы в строку, было реализована несколько меток. Метка start каждый раз запускается строку заново и добавляет к счётчику(который отвечает на какой букве мы находимся) единицу и

также добавляет 1 к переменной counter. Метка new_symvol отвечает за загрузку символа в регистр al при помощи lodsb и также увеличивает на 1 счётчик кол-ва считанных символов, если мы встречаем конец строки то мы отправляемся на метке check_last_symvol . Метка check_last_symvol проверяет не дошли ли мы до конечного символа(‘я’) и в случае чего завершает программу если мы дошли до конца строки то начинаем заново в любом другом случае переходим к метке new_symvol. Метке check_symvol отвечает за проверку буквы и встреченного символа если мы прошли все заглавные символы, то при помощи метки Jump мы переходи к прописным буквам. Последняя метка write_index записывает в массив letters на каком моменте мы встречаем нашу букву и после чего опять переходим в start.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Тестирование программы

№	Входные данные	Результат	Комментарий
1	абфу	1 1 2 2 20 4 21 3	Верный результат
2	АБВУЖ	1 1 2 2 3 3 7 5 20 4	Верный результат
3	123 qwe ЕРУ!!	6 9 17 10 20 11	Верный результат

Выводы.

Изучены представление и обработка строк на языке Ассемблера.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

Название файла: lab4.cpp

```
#include <iostream>
#include <fstream>

char input[81];
int letters[65] = {0};
int counter = -1; // индекс массива letters
int main() {
    setlocale(LC_ALL, "Russian");
    std::cout << "Lutsenko Dmitry 1381.\nForming the number of the entered
Russian letter alphabetically and the position number of its first
occurrence in the input string and displaying them on the screen.\n";
    std::cin.getline(input, 81);

    __asm {
        push ds
        pop es
        sub ax, ax
        mov ah, 127 // Символ перед 'A'
        start:
            mov esi, offset input
            inc ah // счётчик (на какой мы букве)
            inc counter // увеличение индекса числа
            sub ecx, ecx
        new_symvol :
            sub al, al
            lodsb // загружаем символ в al
            inc ecx // увеличиваем кол-во считанных символов
            cmp al, '\0'
            je check_last_symvol
        check_symvol :
            cmp al, ah
            je write_index // найден прописной символ
            cmp ah, 176
            je jump
            jmp check_last_symvol
        jump :
            add ah, 48
            jmp check_symvol
        s_e:
            mov ES : letters[edi * 4], 666
            jmp start
        write_index :
            mov edi, counter
            mov ES : letters[edi * 4], ecx
            //cmp ah, 240
            //je s_e
            jmp start // начинаем проверку новой буквы
        check_last_symvol :
            cmp ah, 240 // если дошли до э завершаем
            je final
            cmp al, '\0' // строка закончилась начинаем новый цикл
```

```
            je start // метка на начало цикла
            jmp new_symvol // продолжаем считывание строки

            final:
    };
    std::fstream file;
    file.open("output.txt");
    for (int i = 0; i < 65; i++) {
        if (letters[i] != 0) {
            if (i > 31) {
                std::cout << i - 31 << " " << letters[i] << std::endl;
                file << i - 31 << " " << letters[i] << std::endl;
            }
            else {
                std::cout << i + 1 << " " << letters[i] << std::endl;
                file << i + 1 << " " << letters[i] << std::endl;
            }
        }
    }
    file.close();
    return 0;
}
```