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

ОТЧЕТ

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

Студент гр. 1303 _____ Иевлев Е.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить представление и обработку символьной информации с использованием строковых команд на языке Ассемблера. Разработать программу, которая обрабатывает строку.

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.

Ход работы.

Преобразование введенных во входной строке шестнадцатиричных цифр в десятичную СС, остальные символы входной строки передаются в выходную строку непосредственно.

Выполнение работы.

В начале выполнения программы в консоль выводится сообщение с именем, номером группы и заданием, а также с запросом на ввод строки для обработки. С помощью getline считывается не более 81 символа с учетом символа окончания строки ‘\0’. Setlocale и system дают нам возможность работать с кириллицей.

Далее объявляется ассемблерный блок через ключевое слово __asm, в котором происходит считывание каждого символа введенной строки с помощью команды lodsb. В процессе выполнения программа проверяет каждый символ на вхождения в промежутки от ‘A’ до ‘F’, при помощи флагов jl и jg, и если тот входит, то преобразует число из шестнадцатиричной сс в десятиричную, определяя конкретный символ при помощи флага je, перезаписывет значение в регистре ax на десятичное число от 10 до 15 в обратном порядке и командой stosw сохраняет ax по адресу es:edi. Если символ не входит в этот промежуток, то программа выводит его без преобразований, для этого используются метки и команды перехода к меткам: jl, jg. Команда stosb записывает символ, лежащий в al, в es:edi. Если встречается символ конца строки, ассемблерный блок заканчивается.

В конце, полученная строка на ЯВУ выводится на экран и записывается в текстовый файл.

Исходный код программы см. в приложении А.

Результаты тестирования программы main.exe представлены в табл. 1.

Таблица 1 – Тестирование программы main.exe.

№ Теста	Вввод	Вывод	Результат
1	кпфф!НёJAOB1095^7	кпфф!НёJ10O111095^7	Верно
2	ФOIDC прив UGIFT13jg	ФOI1312 прив UGI15T13jg	Верно
3	fgashjGJSACVKL	fgashjGJS1012VKL	Верно
4	FCADBE	151210131114	Верно
5	E194716F	1419471615	Верно

Вывод.

В результате лабораторной работы была изучена обработка символьной информации с использованием ассемблерного блока в коде на ЯВУ.

Приложение А

Исходный код программы

Название файла: main.cpp

```
#include <iostream>
using namespace std;
char input_str[81];
char output_str[81];


int main() {
      system("chcp 1251 > nul");
      setlocale(LC_CTYPE, "rus");
      cout << "Иевлев Егор 1303\n";
    cout << "Преобразование введённых во входной строке шестнадцатиричных
чисел в десятичную СС, \n";
    cout << "остальные символы входной строки передеются в выходную строку
непосредственно.\n";
    cout << "Введите строку:\n";
      cin.getline(input_str, 81);

    __asm {
        push ds
        pop es
        mov esi, offset input_str
        mov edi, offset output_str

        reading:
            lodsb
            cmp al, '\0'
            je ending
            cmp al, 'A'
            jl write
            cmp al, 'F'
            jg write

            cmp al, 'A'
            je converting_A
            cmp al, 'B'
            je converting_B
            cmp al, 'C'
            je converting_C
            cmp al, 'D'
            je converting_D
            cmp al, 'E'
            je converting_E
            cmp al, 'F'
            je converting_F

        converting_A:
            mov ax, '01'
            stosw
            jmp reading

        converting_B:
            mov ax, '11'
            stosw
            jmp reading
```

```
        converting_C:
            mov ax, '21'
            stosw
            jmp reading

        converting_D:
            mov ax, '31'
            stosw
            jmp reading

        converting_E :
            mov ax, '41'
            stosw
            jmp reading

        converting_F :
            mov ax, '51'
            stosw
            jmp reading

        write:
            stosb
            jmp reading

        ending:
    };

      cout << output_str;

      return 0;
}
```