**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. 1303 |  | Самохин К.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.

## Выполнение работы.

Разработана программа на языке C++ с использованием ассемблерных вставок. При ее запуске в консоль выводится строка, содержащая имя, фамилию, номер группы, а также задание. Затем выводится сообщение с просьбой ввести входную строку. С помощью метода *getline()* считывается не более 81 символа с учетом нуля-терминатора. Setlocale и system позволяют программе работать с кириллицей.

Далее объявляется ассемблерная вставка через ключевое слово asm. Настраиваем расширенные сегменты ESI и EDI на входную и выходную строки соответственно. Затем создается метка *checking*, по которой будем переходить при проверке очередного символа исходной строки. С помощью

команды lodsb выгружается очередной символ в нижний байт регистра-аккумулятора(AX). В процессе выполнения программы поступивший на вход символ проверяется на вхождение в промежутки ‘0’ – ‘9’ и ‘A’-‘Z’. Если символ является цифрой, производится переход по метке *inverse*, где цифра инвертируется (0 на 9, 1 на 8, 2 на 7 и т.д.) после завершения инвертирования вызывается переход по метке *writing*, где символ записывается в выходную строку с помощью команды stosb, которая выгружает символ из регистра-аккумулятора в память. Если же символ является заглавной буквой латинского алфавита, производится переход по метке *swap*, где символ приводится к нижнему регистру, также после этого производится запись символа с помощью перехода по метке *writing*.

Для перехода по меткам используются следующие команды условного перехода: je, jg, jl, gle, а также команды безусловного перехода jmp. Если встречается символ конца строки, то совершается переход по метке *end*, после чего ассемблерная вставка оканчивается.

В конце, полученная строка выводится на экран и записывается в текстовый файл с помощью языка ВУ.

Таблица 1. Тестирование программы EVM.exe

|  |  |
| --- | --- |
| Входные данные | Результат |
| 123 !@# qwe QWE ёйцу ЁЙЦУ | 876 !@# qwe qwe ёйцу ЁЙЦУ |
| Абв Abc 123 YUvvs | Абв abc 876 yuvvs |
| !345 wjh UU абвгд | !654 wjh uu абвгд |

## Выводы.

В ходе лабораторной работы была изучена обработка символьной

информации с использованием языка ассемблера, а также разработана программа на языке ВУ, использующая вставку на языке ассемблера.

# Приложение А

Название файла: EVM.cpp

#include <iostream>

#include <fstream>

#include <windows.h>

char input\_str[81];

char output\_str[81];

int main() {

system("chcp 1251 > nul");

setlocale(LC\_CTYPE, "rus");

std::cout << "Самохин Кирилл 1303.\nВариант 22. Преобразование всех заглавных латинских букв входной строки в строчные, а десятичных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.\n";

std::cout << "Введите строку: ";

std::cin.getline(input\_str, 81);

std::ofstream file("res.txt");

\_\_asm {

push ds

pop es

mov esi, offset input\_str

mov edi, offset output\_str

check :

lodsb

cmp al, '\0'

je end

cmp al, '0'

jb writing

cmp al, '9'

jbe inverse

cmp al, 'A'

jb writing

cmp al, 'Z'

jbe swap

jmp writing

swap :

add al, 32

jmp writing

inverse :

neg al

add al, 105

jmp writing

writing :

stosb

jmp check

end :

};

std::cout << "Результат: " << output\_str;

file << output\_str;

file.close();

return 0;

}