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. 9382 _____ Докукин В.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить команды для работы с массивами данных на языке ассемблера, написать программу, обрабатывающую вводимую строку определенным способом и познакомиться с принципом встраивания in-line на примере ЯВУ C++.

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).

Вариант 7.

7. Инвертирование введенных во входной строке цифр в восьмеричной СС и преобразование заглавных русских букв в строчные, остальные символы входной строки передаются в выходную строку непосредственно.

Ход работы:

При разработке программы были использованы следующие команды:

LODSB - копирует один байт из памяти по адресу DS:SI в регистр AL. После выполнения команды, регистр SI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1.

STOSB - сохраняет регистр AL в ячейке памяти по адресу ES:DI. После выполнения команды, регистр DI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1.

Для выполнения работы создается 2 статических массива символов с размера Nmax. В один из них загружается вводимая пользователем строка, после чего ассемблерная вставка посимвольно обрабатывает первый массив и записывает результат во второй. Второй массив выводится на экран и в файл.

Тестирование.

Вводные данные	Результат
РУССКИЕ ВПЕРЕД	русские вперед
01234567АБВГДЕЙКА89	76543210абвгдейка89
SPBETU - Лучший ВУЗ России!	SPBETU - лучший вуз россии!

Выводы.

В результате выполнения данной лабораторной работы была разработана программа для обработки строк способом, указанным в задании. Кроме того, был освоен один из методов работы ассемблерными вставками на языке C++.

Приложение.

Имя файла: main.cpp

```
#include <fstream>
#include <iostream>
#include <windows.h>

int main(){
        SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

        const int Nmax = 80;

        std::cout<<"Program for inverting octal numbers\n"
                         "and converting russian letters into lower case.\n"
                         "Made by ETU student Dokukin V.M., 9382.\n";

        char* source_str = new char[Nmax+1]; // source_str[80] = '\0'
        char* dest_str = new char[Nmax+1];   // dest_str[80] = '\0'

        std::cin.getline(source_str, Nmax);
        source_str[Nmax] = '\0';
        dest_str[Nmax] = '\0';
        // Symbol codes: 'А' = -128, 'Б' = -127, ..., 'Я' = -97, 'а' = -96,
'б' = -95, ..., 'п' = -81,'р' = -32, ..., 'я' = -17, 'Ё' = -16, 'ё' = -15

        asm("mov  %0, %%rsi\n\t"     // SI = source_str
        "mov  %1, %%rdi\n\t"     // DI = dest_str
        "mov $80, %%ecx\n\t"     // ECX = Nmax

        "get_symbol:"
        "lodsb (%%rsi)\n\t"        // Загружаем символ в AL
        "cmpb $0x30, %%al\n\t"     // Сравниваем символ с кодом цифры 0
        "jl is_character\n\t"      // Если меньше, то не цифра, идем дальше к
проверке на буквы
        "cmpb $0x37, %%al\n\t"     // Сравниваем символ с кодом цифры 7
        "jg is_character\n\t"      // Если больше, то не восьмеричная цифра,
идем к проверке на буквы

        "is_number:"
        "sub $0x30, %%al\n\t"    // Вычитаем 48, чтобы получить цифру
        "xor $0x7, %%al\n\t"     // Инвертируем последние 3 бита
        "add $0x30, %%al\n\t"    // Прибавляем 48, чтобы получить код новой
цифры
        "jmp final\n\t"          // Переходим к выводу в выходную строку

        "is_character:"
        "cmpb $0xc0, %%al\n\t "                // Сравниваем с символом 'А'
        "jl final\n\t"                         // Тсли меньше, переходим к
выводу в выходную строку
        "cmpb $0xdf, %%al\n\t"                 // Сравниваем с символом 'Я'
        "jg final\n\t"                                 // Тсли больше, то
преходим к выводу в выходную строку
        "add $0x20, %%al\n\t"                  // Получаем строчную букву

        "final:"
        "stosb (%%rdi)\n\t"      // Записываем символ в выходную строку
        "loop get_symbol\n\t"    // Возвращаемся в начало пока ecx!=0
        ::"m"(source_str),"m"(dest_str)
    );

        std::cout<<"\n---------------------------------\n";
        std::cout<<"Source string: " << source_str << '\n';
```

```
        std::cout<<"Destination string: " << dest_str << '\n';

        std::ofstream f("output.txt");
        if (f.is_open()){
                f << dest_str;
                f.close();
        }

        return 0;
}
```