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 _____ Герасев Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить команды для работы со строками 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).

4 Вариант

4. Преобразование всех заглавных латинских букв входной строки в строчные, а восьмеричных цифр в инверсные, 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 строки с фиксированным количеством выделенной памяти. В одну из них загружается ввод, после чего ассемблерная вставка по символьно обрабатывает одну из строк и результат записывает во вторую. Вторая строка выводится.

Тестирование.

Вводные данные	Результат
1234567890ASDWasdw	6543210897asdwasdw
oH rEaLy11	oh realy66
1234567 IS NOT 7654321	6543210 is not 0123456

Выводы.

В результате выполнения лабораторной работы был разработан код для определенной обработки строк. Освоен один из методов работы ассемблерными вставками.

Приложение.

Текст файла main.cpp

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

int main()
{
    std::cout<<"Вид преобразования: 4. Преобразование всех заглавных
латинских букв входной строки в\n";
    std::cout<<"строчные, а восьмеричных цифр в инверсные, остальные символы
входной строки\n";
    std::cout<<"передаются в выходную строку непосредственно.\n";
    std::cout<<"Работу выполнил: студент группы 9382 Герасев Георгий\n";

    char* str1 = new char[80];
    char* str2 = new char[80];
    std::cout<<"Введите строку: ";

    std::cin.getline(str1,80);
    asm("mov  %0,%%rsi\n\t"     //положили в al 0
        "mov  %1,%%rdi\n\t"
        "mov $80,%%ecx\n\t"     //положили длину строки в ecx

        "get_symbol:"
        "lodsb (%%rsi)\n\t"     //загружаем символ в al
        "cmpb $0x30,%%al\n\t"   //сравниваем символ с кодом цифры 0
        "jl character_case\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 character_case\n\t"      //если больше то не цифра в 8 сс идем к
проверке на буквы

        "number_case:"
        "sub $0x30,%%al\n\t"    //вычитаем 30 чтобы получить цифру
        "xor $0x7,%%al\n\t"     //инвертируем последние 3 бита
        "add $0x30,%%al\n\t"    //прибавляем 30 чтобы получить код цифры
        "jmp add_to_result\n\t"         //переходим к выводу в выходную
строку

        "character_case:"
        "cmpb $0x41,%%al\n\t "  //сравниваем с символом "A"
        "jl add_to_result\n\t"          //если меньше, переходим к выводу в
выходную строку
        "cmpb $0x5a,%%al\n\t"   //сравниваем с символом "Z"
        "jg add_to_result\n\t"          //если больше, то преходим к выводу в
выходную строку
        "add $0x20,%%al\n\t"    //получаем строчную букву

        "add_to_result:"
        "stosb (%%rdi)\n\t"     //записываем символ в выходную строку
        "loop get_symbol\n\t"    //возвращаемся в начало пока ecx!=0
        ::"m"(str1),"m"(str2)
    );
```

```
    std::cout<<"Преобразованная строка:\t"<<str2<<'\n';
    std::ofstream fileOut("Result.txt");
    if(fileOut.is_open())
    {
        fileOut<<str2;
        fileOut.close();
    }

    return 0;
}
```