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»
Тема: Представление и обработка символьной информации с использованием строковых команд.

Студент гр. 9383 _____ Чумак М.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Научиться обрабатывать строку на языке Ассемблера. Написать программу, включая ассемблерную часть в ЯВУ, по принципу in-line.

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 №23:

Преобразование всех строчных латинских букв входной строки в заглавные, а шестнадцатеричных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.

Выполнение работы.

В ходе работы была реализована программа на языке C++ с in-line вставкой языка Ассемблера. Функция start_message() – функция для вывода титульной таблички с указанием вида преобразования и автора программы.

Функция main() – функция считывания строки, а также её обработки при помощи языка Ассемблера. Если нашлась латинская буква нижнего регистра, то она заменяется на такую же букву верхнего регистра. Если нашлась шестнадцатеричная цифра, то заменяем ее на инверсионную. В конце выводим обработанную строку в консоль и в файл.

Тестирование.

Входные данные	Выходные данные
<пусто>	<пусто>
abcdefABCDEF	ABCDEF543210
ABrACADABrA0123456789	54R535254R5FEDCBA9876

Выводы.

Получены знания об обработке строк на языке ассемблера. Написана программа, в которой ассемблерная часть включена в ЯВУ.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Файл lab4.asm

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

using namespace std;

void start_message() {
    cout << "Вариант №23." << endl
        << "Вид преобразования следующий:" << endl
        << "Преобразование всех строчных латинских букв входной строки в заглавные, а" << endl
            << "шестнадцатиричных цифр в инверсные, остальные символы входной строки передаются в" << endl
        << "выходную строку непосредственно." << endl
        << "Работу выполнил студент группы 9383 Чумак Михаил." << endl;
}

int main() {
    setlocale(0, "");
    start_message();
    char str1[80], str2[80];
    cout << "Введите строку: ";
    cin.getline(str1, 80);
    __asm {
        sub eax, eax; eax = 0
        sub esi, esi; esi = 0
        lea edi, str1; edi указывает на начало str1

        start :
```

```
mov al, [edi]; считываем текущий символ str1 по индексу edi в al
    cmp al, 'z'; если больше чем z,
    jg write; то записываем данные в выходную строку
    cmp al, 'a'; если меньше чем a,
    jl check_hex; то проверяем, является ли число шестнадцатеричным,
        sub al, 0x20; иначе переводим латинскую букву к верхнему регистру
    jmp write; и записываем символ в выходную строку

    check_hex :
cmp al, 0x30; если меньше чем 0,
    jl write; то записываем символ в выходную строку
    cmp al, 0x46; если больше чем F,
    jg write; то записываем символ в выходную строку
    cmp al, 0x3A; если входим в диапазон цифр 0 - 9,
    jl hex_inverse; то выполняем преобразование
    cmp al, 0x40; если входим в диапазон цифр A - F,
    jg hex_inverse; то выполняем преобразование
      jmp write; иначе выводим символ, который не попал ни под одно условие

    hex_inverse :
cmp al, 0x30; если 0,
    je hex_0; то переходим к инвертированию
    cmp al, 0x31; если 1,
    je hex_1; то переходим к инвертированию
    cmp al, 0x32; если 2,
    je hex_2; то переходим к инвертированию
    cmp al, 0x33; если 3,
    je hex_3; то переходим к инвертированию
```

```
    cmp al, 0x34; если 4,
    je hex_4; то переходим к инвертированию
    cmp al, 0x35; если 5,
    je hex_5; то переходим к инвертированию
    cmp al, 0x36; если 6,
    je hex_6; то переходим к инвертированию
    cmp al, 0x37; если 7,
    je hex_7; то переходим к инвертированию
    cmp al, 0x38; если 8,
    je hex_8; то переходим к инвертированию
    cmp al, 0x39; если 9,
    je hex_9; то переходим к инвертированию
    cmp al, 0x41; если A,
    je hex_A; то переходим к инвертированию
    cmp al, 0x42; если B,
    je hex_B; то переходим к инвертированию
    cmp al, 0x43; если C,
    je hex_C; то переходим к инвертированию
    cmp al, 0x44; если D,
    je hex_D; то переходим к инвертированию
    cmp al, 0x45; если E,
    je hex_E; то переходим к инвертированию
    cmp al, 0x46; если F,
    je hex_F; то переходим к инвертированию

    hex_0 :
mov al, 0x46
    jmp write
    hex_1 :
mov al, 0x45
```

```
    jmp write
    hex_2 :
mov al, 0x44
    jmp write
    hex_3 :
mov al, 0x43
    jmp write
    hex_4 :
mov al, 0x42
    jmp write
    hex_5 :
mov al, 0x41
    jmp write
    hex_6 :
mov al, 0x39
    jmp write
    hex_7 :
mov al, 0x38
    jmp write
    hex_8 :
mov al, 0x37
    jmp write
    hex_9 :
mov al, 0x36
    jmp write
    hex_A :
mov al, 0x35
    jmp write
    hex_B :
mov al, 0x34
```

```
            jmp write
            hex_C :
        mov al, 0x33
            jmp write
            hex_D :
        mov al, 0x32
            jmp write
            hex_E :
        mov al, 0x31
            jmp write
            hex_F :
        mov al, 0x30
            jmp write

            write :
        mov str2[esi], al; помещаем текущий символ в выходную строку
            cmp al, 0; если был встречен нулевой символ,
            je end_func; то заканчиваем
            inc edi; иначе увеличиваем индексы
            inc esi
            jmp start; и возвращаемся к считыванию строки
            end_func :
    }
    cout << str2;
    ofstream output;
    output.open("out.txt");
    output << str2;
    output.close();
    return 0;
}
```