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: Обработка символьной информации
Вариант 8

Студент гр. 0382 _____ Кондратов Ю.А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

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).

Задание.

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

Выполнение работы.

В качестве ЯВУ используется язык C++, компилируемый при помощи MSVC.

Входная строка считывается в массив input. Выходная строка записывается в массив output, который далее выводится на консоль и в файл out.txt.

Так как команды работы со строками в качестве источника используют адрес DS:ESI а в качестве адреса назначение ES:EDI, регистру ES присваивается значение DS. Регистру ESI присваивается значение смещения input. Регистру EDI присваивается значение смещения output.

Далее начиная с сметки loop_start организована структуру цикла. Сначала при помощи lodsb в AL записывается символ из input, после чего он сравнивается с символом конца строки. Если символ в AL оказался равен символу конца строки, значит цикл необходимо завершить, поэтому производится условный переход на метку loop_final, в которой символ конца строки записывается в выходной массив.

Если же символ в AL не равен символу конца строки, то он последовательно сравнивается с символами ‘A’, ‘B’, …, ‘F’, которые являются шестнадцатеричными цифрами. Сравнение происходит при помощи cmp и если символ в AL оказался не равен одной из шестнадцатеричный цифр, то сравнение продолжается с другими цифрами, иначе в ax при помощи stows помещается его значение в десятичной системе счисления (теперь это два символа => два байта).

Если символ не равен ни одной из десятичных цифр, то он просто копируется в выходной массив.

После обработки происходит вывод строки с cout и в поток выходного файла file.

Тестирование.

Для проверки работоспособности программы разработаны тесты, представленные в таблице 2.

Таблица 2 – Тесты для проверки работоспособности программы.

Номер теста	input	output	Вердикт
1.	i have already lived ABCDEF lifes	i have already lived 101112131415 lifes	passed
2.	1234567	1234567	passed
3.	рашн текст энд рашн лэттерс АВСДЕФ	рашн текст энд рашн лэттерс АВСДЕФ	passed
4.	And there is nothing ElsE to tEst, EvErything works	10nd there is nothing 14ls14 to t14st, 14v14rything works	passed

Выводы.

В ходе работы были изучены основные принципы представления и обработки символьной информации с использованием строковых команд на языке ассемблера. Была разработана программа преобразующая введённые шестнадцатеричные цифры в десятичную СС.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp:

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

using namespace std;

char input[85];
char output[170];

int main()
{
    cout << "Kondratov Yurii, group 0382, task: hexadecimal -> decimal" << endl;
    ofstream file;
    file.open(R"(C:\Coding\Assembly\lab4\out.txt)");
    cin.getline(input, 80);
    __asm {
        mov ax, ds
        mov es, ax
        mov esi, offset input
        mov edi, offset output
    loop_start:
        lodsb
        cmp al, '\0'
        je loop_final
    if_a:
        cmp al, 'A'
        jne if_b
        mov ax, '01'
        stosw
        jmp loop_start
    if_b:
        cmp al, 'B'
        jne if_c
        mov ax, '11'
        stosw
        jmp loop_start
    if_c:
        cmp al, 'C'
        jne if_d
        mov ax, '21'
        stosw
        jmp loop_start
    if_d:
        cmp al, 'D'
        jne if_e
        mov ax, '31'
        stosw
        jmp loop_start
    if_e:
        cmp al, 'E'
        jne if_f
        mov ax, '41'
        stosw
```

```
        jmp loop_start
    if_f:
        cmp al, 'F'
        jne other
        mov ax, '51'
        stosw
        jmp loop_start
    other:
        stosb
        jmp loop_start
    loop_final:
        stosb
    };
    std::cout << output;
    file << output;
    file.close();
    return 0;
}
```