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 19

Студент гр. 0382 _____ Тюленев Т.В.

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.

Заменить введенные во входной строке латинские буквы на десятичные числа, соответствующие их номеру по алфавиту, остальные символы входной строки передать 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’. Сравнение происходит при помощи cmp и если символ в AL оказался не равен одной букв, то сравнение продолжается с другими буквами, 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 букв, 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.	abcdef	010203040506	passed
2.	ABCDEF	ABCDEF	passed
3.	абвгде	абвгде	passed
4.	АБВГДЕ	АБВГДЕ	passed
5.	123456	123456	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

lb4.cpp:

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

using namespace
std;

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

int main()
{
    cout <<
"Tulenev Tumofey,
group 0382, task:
Replace Latin
letters with
decimal numbers" <<
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,'10'
        stosw
        jmp
loop_start
    if_b:
        cmp al, 'b'
        jne if_c
        mov ax,'20'
        stosw
        jmp
loop_start
    if_c:
```

```
        cmp al, 'c'
        jne if_d
        mov ax,'30'
        stosw
        jmp
loop_start
    if_d:
        cmp al, 'd'
        jne if_e
        mov ax,'40'
        stosw
        jmp
loop_start
    if_e:
        cmp al, 'e'
        jne if_f
        mov ax,'50'
        stosw
        jmp
loop_start
    if_f:
        cmp al, 'f'
        jne if_g
        mov ax,'60'
        stosw
        jmp
loop_start
    if_g:
        cmp al, 'g'
        jne if_h
        mov ax,'70'
        stosw
        jmp
loop_start
    if_h:
        cmp al, 'h'
        jne if_i
        mov ax,'80'
        stosw
        jmp
loop_start
    if_i:
        cmp al, 'i'
        jne if_j
        mov ax,'80'
        stosw
        jmp
loop_start
    if_j:
        cmp al, 'j'
        jne if_k
        mov ax,'90'
        stosw
        jmp
loop_start
    if_k:
        cmp al, 'k'
        jne if_l
        mov ax,'01'
```

```
        stosw
        jmp
loop_start
    if_l:
        cmp al, 'l'
        jne if_m
        mov ax,'11'
        stosw
        jmp
loop_start
    if_m:
        cmp al, 'm'
        jne if_n
        mov ax,'21'
        stosw
        jmp
loop_start
    if_n:
        cmp al, 'n'
        jne if_o
        mov ax,'31'
        stosw
        jmp
loop_start
    if_o:
        cmp al, 'o'
        jne if_p
        mov ax,'41'
        stosw
        jmp
loop_start
    if_p:
        cmp al, 'p'
        jne if_q
  mov ax, '51'
        stosw
        jmp
loop_start
    if_q:
        cmp al, 'q'
        jne if_r
        mov ax,'61'
        stosw
        jmp
loop_start
    if_r:
        cmp al, 'r'
        jne if_s
        mov ax,'71'
        stosw
        jmp
loop_start
    if_s:
        cmp al, 's'
        jne if_t
        mov ax,'81'
        stosw
        jmp
loop_start
```

```
     if_t:
        cmp al, 't'
        jne if_u
        mov ax,'91'
        stosw
        jmp
loop_start
     if_u:
        cmp al, 'u'
        jne if_v
        mov ax,'02'
        stosw
        jmp
loop_start
     if_v:
        cmp al, 'v'
        jne if_x
        mov ax,'12'
        stosw
        jmp
loop_start
     if_x:
        cmp al, 'x'
        jne if_y
        mov ax,'22'
        stosw
        jmp
loop_start
     if_y:
        cmp al, 'y'
        jne if_z
        mov ax,'32'
        stosw
        jmp
loop_start
     if_z:
        cmp al, 'z'
        jne other
        mov ax,'42'
        stosw
        jmp
loop_start
    other:
        stosb
        jmp
loop_start
    loop_final:
        stosb
    };
    std::cout <<
output;
    file << output;
    file.close();
    return 0;
}
```