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. 1382 _____ Исайкин Г. И.

Преподаватель _____ Ефремов М. А

Санкт-Петербург
2022

Цель работы.

Создать программу на ЯВУ, основная логика который внутри написана на Ассемблере через специальные инструменты языка.

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 8.

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

Выполнение работы.

Программа выполнена на языке C++ (в ОС Windows).

В программе для ввода и вывода информации созданы два массива char как глобальные переменные — myin и myout соответственно. Первая может содержать до 81 символа (включая «\0»), вторая — до 161 (включая «\0»).

Главной функция main начинается с считывания строчки с консоли. Считываются только первые 80 символов, если их больше 80. После мдёт блок на Ассемблере. В нём мы перемещаем в регистр es значение ds, а в регистры esi и edi отступ от области памяти, на которую указывает регистр ds, до мест хранения myin и myout соответственно. Сделано это для роботы с строками через команды stos и lods. После мы берём байт из myin через команду lodsb (команда берёт байт по отступу esi от адреса в es, копирует значение в al, после чего увеличивает отступ на байт), после чего проревем этот символ через cmp и jne. Если символ равен , то в ax помещаются два символа «01», «11», «21», «31», «41» или «51» (символы хранятся в обратном порядке т. к. байты в памяти, следственно и элементы массива char, в обратном прядке), после чего используется команда stosw (команда берёт значения по отступу edi от адреса в es, и вставляет туда значение ax, после чего увеличивает отступ на 2). Если символ был другим, то мы вставляем символ в строку через stosb (аналогична stosw, но для байта). Далее мы проверем, является ли символ в по отступу esi «\0». Если да, то перемещаем в al значение «\0» и вставляем его в конец через stosb, после чего блок на Ассемблере заканчивается. Если нет, то программа перемещается в начало блока на Ассемблере. Далее мы выводим строку в консоль и программа завершает работу.

Таблица 2 — Тесты программы

№ теста	Ввод	Вывод
1	A B C D E F 124563nkllknppb ABHOUGY	10 11 12 13 14 15 124563nkllknppb 1011HOUGY
2	Davfavdgd a AedkgfbByfho656556	Davfavdgd a 10edkgfb11yfho656556
3	6768AC75B FF865CA 6774	676810C7511 FF8651210 6774
4	3246527856 6454	3246527856 6454

Выводы.

Создана программа на высоком языке C++ с реализацией основной логики на Ассемблере через __asm.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4.cpp

```
#include <iostream>
#include <fstream>
#include <cstdio>
char myout[161];
char myin[81];
int main() {
    fgets(myin, 81, stdin);
    __asm {
        push ds
        pop es
        mov esi, offset myin
        mov edi, offset myout
        take :
        lodsb
        checkA :
        cmp al, 'A'
        jne checkB
        mov ax, '01'
        jmp end
        checkB:
        cmp al, 'B'
        jne checkC
        mov ax, '11'
        jmp end
        checkC:
        cmp al, 'C'
        jne checkD
        mov ax, '21'
        jmp end
        checkD:
        cmp al, 'D'
        jne checkE
        mov ax, '31'
        jmp end
        checkE:
        cmp al, 'E'
        jne checkF
        mov ax, '41'
        jmp end
        checkF:
        cmp al, 'F'
        jne another
        mov ax, '51'
        jmp end
        another:
        stosb
        jmp check
        end:
        stosw
```

```
        check:
        cmp[esi], '\0'
        jne take
        mov al, '\0'
        stosb
    }
    std::cout << myout;
}
```