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: Изучение программирования обработки символьной информации с использованием команд пересылки строк.

Студентка гр. 9382 _____ Сорокумов С. В.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2020

Цель работы.

Изучить особенности работы с строками на языке ассемблера.

Задание.

Разработать программу обработки символьной информации, реализующую функции:

1. инициализация (вывод титульной таблички с указанием вида преобразования и автора программы) - на языке высокого уровня (Pascal или Си);
2. ввода строки символов, длиной не более Nmax (<=80), с клавиатуры в заданную область памяти - на языке высокого уровня;
3. выполнение заданного в таблице 1 преобразования исходной строки с записью результата в выходную строку - на Ассемблере;
4. вывода результирующей строки символов на экран и ее запись в файл - на ЯВУ.
5. Ассемблерную часть программы включить в программу на Pascal или Си по принципу встраивания (in-line).

19. Заменить введенные во входной строке латинские буквы на десятичные числа, соответствующие их номеру по алфавиту, остальные символы входной строки передать в выходную строку непосредственно.

Теоретические положения.

Для работы со строками, или цепочками символов или чисел (т.е. попросту говоря, с массивами произвольных данных) на языке ассемблера предусмотрен ряд специальных команд:

lea в Ассемблере вычисляет эффективный адрес источника и помещает его в приёмник.

repne - повторить следующую строковую операцию, если не равно

Ход работы.

Программа на вход получает строку, в которой проходит по всей строке циклом, с помощью команды loop. Каждый элемент строки проверяется на латинский символ, и переводит его в десятичное представление этого числа из латинского алфавита в том порядке, как они представлены. Остальные остаются неизменными.

Таблица 1 – Результаты тестирования программы

Входная строка	Выходная строка
Aa 77	0101 77
Bc проект	0203 проект
Hello world	0805121215 2315181204
пустая строка	*пустая строка*

Вывод.

В ходе данной работы были изучены основы работы со строками на языке ассемблера, использован метод ассемблерной вставки в программе преобразования латинских букв исходной строки в порядковый номер в алфавите.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <stdio.h>

#define N 80
using namespace std;

int main()
{
    system("chcp 1251 > nul");
    char _str[N + 1];
    cout << "ЛР4. Сорокумов Сергей 9382. \n 19. Заменить введенные во входной строке латинские буквы на числа, соответствующие их номеру по алфавиту, представленному в десятичной СС, остальные символы входной строки передать в выходную строку непосредственно.\n";
    char str_out[N * 2 + 1];
    int i = 0;
    cin.getline(_str, N);
    _asm {
        sub eax, eax;
        mov al, 0;            in al code of str ending symbol
            mov ecx, N;     ecx = N
            lea edi, _str;    edi now points at start of _str
            repne scas;       ecx now contains N - str.length
            sub ecx, N;       ecx = -str.length
            neg ecx;          ecx = str.length
            mov edx, ecx;     edx = ecx
            sub edi, edi;     edi == 0
            sub esi, esi;     esi == 0

            traverse:
         mov edi, edx;  edi = edx
             sub edi, ecx;    edi - points at last element in str, when we subtracting ecx we pointing to currentIdx, as ecx decreasing every iteration

            mov al, _str[edi];      al contains currentElement

            cmp al, 'a'
            jge small
            cmp al, 'A'
            jge big
            jmp writeSymbol


        small:
            cmp al , 'z'
            jle  number_small
            jmp writeSymbol

        big:
            cmp al, 'Z'
            jle number_big
            jmp writeSymbol
```

```
    number_big:
        sub al, 'A'
            inc al
            cmp al, 10
            jl startAlf
            cmp al, 20
            jl midleAlf
            jmp endAlf

    number_small:
        sub al, 'a'
          inc al
        cmp al, 10
        jl startAlf
        cmp al, 20
        jl midleAlf
        jmp endAlf


    startAlf:
        mov str_out[esi], '0'
        inc esi
            add al, 48
        jmp writeSymbol

    midleAlf:
        mov str_out[esi], '1'
            inc esi
            sub al, 10
            add al, 48
            jmp writeSymbol

    endAlf :
        mov str_out[esi], '2'
            inc esi
            sub al, 20
            add al, 48
            jmp writeSymbol

       writeSymbol :

        mov str_out[esi], al
        inc esi
        loop traverse;

         mov str_out[esi], 0
}
cout << str_out;
return 0;
  }
```