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.

Студентка гр. 1303 _____ Королева П.А

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Разработать программу обработки символьной информации с использованием строковых команд.

Задание.

Вариант 10.

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;

- преобразование введенных во входной строке шестнадцатиричных цифр в двоичную СС, остальные символы входной строки передаются в выходную строку непосредственно.

- 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).

Выполнение работы.

При написании программы были использованы команды lodsb и stosb.

Lodsb считывает байт из регистра esi в al. При этом, после каждого считывания значение регистра увеличивается на 1. Это свойство позволяет перемещаться по строке, последовательно считывая символы.

Stosb записывает в регистр edi символ, хранящийся в al. После записи, регистр edi увеличивается на 1, что также облегчает заполнение строки.

Для написания программы была использована Visual Studio с встроенным макросом __asm. На языке c++ реализовано считывание строки из консоли и вывод переработанной строки на экран и в файл.

В ассемблерной части реализован цикл, считывающий побайтово строку до тех пор, пока не будет встречен символ конца строки. За счет кодирования ascii каждый байт проверяется, является ли он числом в 16ричной системе или другим символом.

Если байт не является числом, он записывается в output_str без изменений.

Если является числом – сначала обрабатывается в зависимости от того, чем выражено цифрой или буквой. Т.к кодирование символов цифр в ascii начинается не с нуля, надо вычесть из кода символа код нуля. Тогда получится чистое число, которое и требуется перевести в двоичную систему. Аналогично с буквенными числами.

После с помощью маски проверяется, какой бит надо поставить в двоичной записи числа. Берется 4х-значная запись, тк максимальная цифра в 16-ричной системе (F) записывается четырьмя битами – 1111.

Таким образом обрабатывается каждый символ в строке.

Текст исходного файла программы lab4 представлен в приложении А.

Вывод.

Разработана программа на языке Ассемблер, выполняющая обработку символьной информации с использованием строковых команд.

ПРИЛОЖЕНИЕ А

Название файла: lab4.cpp

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

char input_str[81];
char output_str[400];

int main()
{
    std::cout << "Author: Koroleva Polina\n";
    std::cout << "Task: converting numbers from hex to bin\n";
    std::cout << "Hello, print your string:\n";
     fgets(input_str, 81, stdin);
     input_str[strlen(input_str) - 1] = '\0';
     __asm {
        push ds
        pop es
        mov esi, offset input_str
        mov edi, offset output_str
    next:
        lodsb;
        cmp al, '0'
        jl writeSymbol
        cmp al, 'F'
        jg writeSymbol
        cmp al, '9'
        jle digit
        cmp al, 'A'
        jge letter
        jmp writeSymbol
        digit :
            sub al, '0'
            jmp tobin
        letter :
            sub al, 'A'
            add al, 10
            jmp tobin
        tobin :
            mov bl, al
            mov cl, 8
            and cl, bl
            jnz writeOne1
            mov al, '0'
            jmp checkSecondBit
            writeOne1:
            mov al, '1'
            checkSecondBit :
            stosb
            mov cl, 4
            and cl, bl
            jnz writeOne2
            mov al, '0'
            jmp checkThirdBit
            writeOne2 :
            mov al, '1'
```

```
            checkThirdBit :
            stosb
            mov cl, 2
            and cl, bl
            jnz writeOne3
            mov al, '0'
            jmp checkFourthBit
            writeOne3 :
            mov al, '1'
            checkFourthBit :
            stosb
            mov cl, 1
            and cl, bl; 0001 and XXX?
            jnz writeOne4
            mov al, '0'
            stosb
            jmp checkNewSymbol
            writeOne4 :
            mov al, '1'
            stosb
            jmp checkNewSymbol
        writeSymbol :
            stosb
        checkNewSymbol :
            mov  ecx, '\0'
            cmp  ecx, [esi]
            je   Exit
            jmp  next
    Exit:
      };
      std::cout << output_str;
      FILE* f;
      fopen_s(&f, "out.txt", "w");
      fwrite(output_str, sizeof(char), strlen(output_str), f);
      return 0;
}
```