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. 9383 _____ Лысова А.М,

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться с представлением символьной информации. Создать программу на языке Ассемблер, реализовывающую обработку символьной информации с использованием строковых команд.

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 10.

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

Ход работы.

В ходе работы была реализована программа на языке Ассемблер, которая обрабатывает символьную информацию, поступающую в виде строки. Ввод и вывод строки реализован на ЯВУ, обработка информации на Ассемблере. Кроме того, реализован вывод в файл также на ЯВУ. Перевод из 16-ричной СС в двоичную осуществлялся с помощью таких инструкций как:

- jl, jle, je — условный переход по метке, если первых аргумент <, <=, == соответственно.

- jmp — безусловный переход по метке.
- inc — инкремент, добавление 1 к заданному аргументу.
- shr — побитовый сдвиг вправо (деление на 2).
- loop — позволяет зациклить какие-то действия, пока счетчик ecx не равен 0.

Для декодировки символов использовалась таблица ASCII.

Тестирование.

1. Входные данные:

   0 1 2 3 4 5 6 7 8 9

   Результат:

   0000 0001 0010 0011 0100 0101 0110 0111 1000 1001
2. Входные данные:

   Hello, one 1, two 2, three 3, four 4, …, a A, b B, c C, d D, e E, f F.

   Результат:

   Hello, one 0001, two 0010, three 0011, fore 0100, …, a 1010, b 1011, c 1100, d 1101, e 1110, f 1111

Исходный код см. в приложении А.

Выводы.

Была реализована программа на языке Ассемблер, реализовывающая обработку символьной информации с использованием строковых комманд.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: lab4.cpp

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

int N = 81;

void printHello(){
      std::cout <<
"_____
__________\n";
    std::cout << "|
|\n";
    std::cout << "|Работа Лысовой Александры
|\n";
    std::cout << "|10. Преобразование введенных во входной строке
шестнадцатиричных цифр в двоичную СС,|\n";
      std::cout << "| остальные символы входной строки передаются в выходную
строку непосредственно.     |\n";
    std::cout << "|
_____
_______|\n";
}

int main(){
      std::ofstream output;
      output.open("./output.txt");
      printHello();

      char* source = new char[N];
      std::cout << "Input string:\n";
      std::cin.getline(source, N);
      char* target = new char[4*N];// = func(source);

      asm(
            ".intel_syntax noprefix\n\t"
```

```
" mov rsi, %0\n\t" //rsi = *target
" mov rdi, %1\n\t"      //rdi = *source

"input:\n\t"
" mov ah, [rdi]\n\t"
" inc rdi\n\t"
" mov bh, 0x8\n\t"
" mov ecx, 4\n\t"
" cmp ah, 0\n\t"
" je end\n\t"

" cmp ah, 0x30\n\t"
" jl check\n\t"
" cmp ah, 0x39\n\t"
" jle digit\n\t"
" cmp ah, 'A'\n\t"
" jl check\n\t"
" cmp ah, 'F'\n\t"
" jle letter\n\t"
" jmp check\n\t"

"digit:\n\t"
" sub ah, 0x30\n\t"
" jmp bin8\n\t"

"letter:\n\t"
" sub ah, 0x37\n\t"
" jmp bin8\n\t"

"bin8:\n\t"
" cmp ah, bh\n\t"
" jl print0\n\t"
" mov al, '1'\n\t"
" mov [rsi], al\n\t"
" inc rsi\n\t"
" sub ah, bh\n\t"
" shr bh\n\t"
" loop bin8\n\t"
" jmp input\n\t"

"print0:\n\t"
```

```
            " mov al, '0'\n\t"
            " mov [rsi], al\n\t"
            " inc rsi\n\t"
            " shr bh\n\t"
            " loop bin8\n\t"
            " jmp input\n\t"

            "check:\n\t"
            " mov [rsi], ah\n\t"
            " inc rsi\n\t"
            " jmp input\n\t"
            "end:\n\t"

            : "=m" (target)
            : "m" (source)
        );

        std::cout << "Output string:\n" << target << '\n';
        output << "Output string:\n" << target << '\n';

        output.close();
        delete[] source;
        delete[] target;
        return 0;
}
```