МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Архитектура ЭВМ и систем»
Тема: Представление и обработка символьной информации с использованием строковых команд.

Студент гр. 9383 _____ Моисейченко К.А.

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 11.

Преобразование введенных во входной строке десятичных 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. Реализован вывод в файл, также на ЯВУ. Перевод из 10-ичной СС в двоичную осуществлялся с помощью следующих инструкций:

- 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:

```
1 2 3 4 5 6 7 8 9 0
```

Выходные данные:

```
0001 0010 0011 0100 0101 0110 0111 1000 1001 0000
```

2. Входные данные:

```
0123456789 the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS
```

Выходные данные:

```
00000001001000110100010101100111100010001 the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS
```

Выводы.

Была реализована программа на языке Ассемблер, обрабатывающая символьную информацию с использованием строковых команд.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lr4.cpp:

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

int N = 81;

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

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

    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 rcx, 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"
        " jmp check\n\t"
```

```
    "digit:\n\t"
    " sub ah, 0x30\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;
}
```

Файл Makefile:

```
all:
	g++ -g -masm=intel lr4.cpp -o lr4
```