

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

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Организация ЭВМ и систем»
Тема: Представление и обработка символьной информации с
использованием строковых команд
Вариант 11

Студентка гр.1381

Рымарь М.И.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Изучить представление символьной информации с использованием строковых команд. Разработать программу обработки символьной информации на языке Ассемблер и включить в программу на языке высокого уровня – C++ по принципу встраивания in-line.

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. По заданию из таблицы *11 варианта*: преобразование введенных во входной строке десятичных 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.

Для реализации задачи, поставленной в лабораторной работе, был написан программный код на языке C++ с использованием принципа встраивания ассемблерной части. Программа была разработана в Visual Studio 2022.

С помощью функции *fgets* входная строка записывается в массив символов *input_string*, который по условию должен состоять из 80 символов. Выходная

строка записывается в массив из 320 символов *output_string*. Этот массив состоит из 320 символов для того, чтобы была выделена память на случай того, если введутся десятичные цифры 8 или 9, которые в двоичной системе счисления занимают 4 символа.

После ключевого слова *__asm* находится блок ассемблерного кода. Регистру ES присваиваем значение DS, так как при работе со строками чтение происходит из памяти по адресу ES:ESI, а запись в ячейку памяти происходит по адресу ES:EDI. Далее присваиваем смещение *input_string* в ESI, а смещение *output_string* в EDI.

Строка начинает обрабатываться с метки *line*. Команда *lodsb* отвечает за чтение байта из строки (копирует один байт из памяти по адресу DS:SI в регистр AL). С помощью команды *cmp* последовательно сравниваются считанный символ с другими десятичными символами: «2», «3», «4», «5», «6», «7», «8», «9». Если символ равен какому-либо из вышеуказанных, то он заменяется, соответственно, на «10», «11», «100», «101», «110», «111», «1000», «1001».

Поскольку десятичные цифры «2» и «3» заменяются на два символа каждый, то их запись происходит следующим образом: в регистр AX помещается в обратном порядке соответствующая запись, а затем отправляется в выходной массив *output_string* с помощью *stosw* (команда отвечает за запись слова в строку, после выполнения команды регистр DI увеличивается на 2).

Десятичные цифры с 4 по 7 заменяются на три символа каждый, поэтому замена происходит следующим образом: первые два символа соответствующей записи помещаются в обратном порядке в регистр AX и отправляются в выходной массив с помощью команды *stosw*, далее оставшийся символ помещается в регистр AL с помощью команды *stosb* (отвечает за запись байта в строку, после выполнения команды регистр DI увеличивается на 1) и также записывается в массив *output_string*.

Десятичные цифры 8 и 9, занимающие четыре символа в двоичной системе счисления заменяются следующим образом: в регистр EAX помещается соответствующая запись в обратном порядке и отправляется в выходной массив

с помощью команды *stosd* (команда отвечает за запись двойного слова в строку, после выполнения команды регистр DI увеличивается на 4).

Если после сравнения символ не оказался равен ни одному из предыдущих, то с помощью команды *stosb* записываем символ в *output_string*.

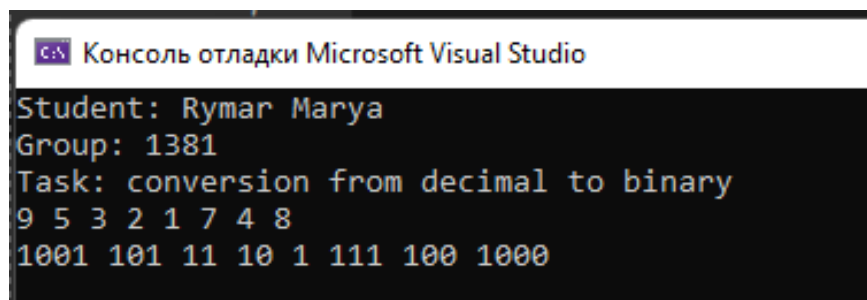
После всех замен и внесений исходных символов в выходной массив переходим к метке *final*. Если по смещению ESI находится символ конца строки, то работа ассемблерного блока заканчивается. Выходной массив *output_string* будет выведен в консоль и записан в файл *output.txt*.

Исходный код программы см. в приложении А.

Тестирование.

Для проверки корректности работы программы было создано четыре теста:

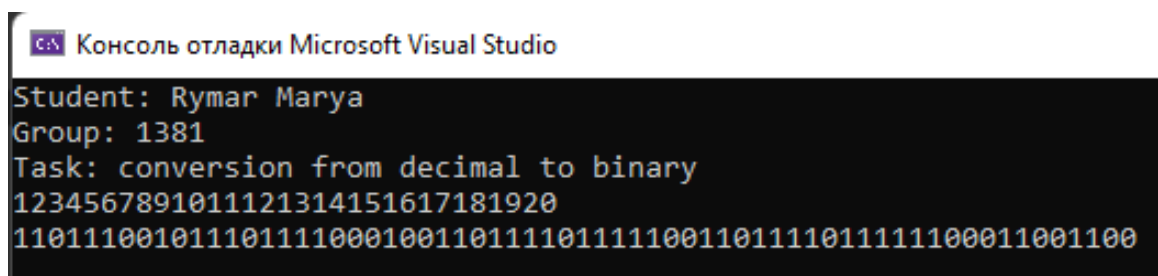
1. Работа программы при вводе строки «9 5 3 2 1 7 4 8» в файл представлена на рисунке 1.



```
Консоль отладки Microsoft Visual Studio
Student: Rymar Marya
Group: 1381
Task: conversion from decimal to binary
9 5 3 2 1 7 4 8
1001 101 11 10 1 111 100 1000
```

Рисунок 1 – Работа программы при тесте 1

2. Работа программы при вводе строки «1234567891011121314151617181920» в файл представлена на рисунке 2.



```
Консоль отладки Microsoft Visual Studio
Student: Rymar Marya
Group: 1381
Task: conversion from decimal to binary
1234567891011121314151617181920
110111001011101111000100110111101111100110111101111100011001100
```

Рисунок 2 – Работа программы при тесте 2

«qwerty12345uiop6789asdfghjklzx0987654321cvbnmqwertyqwertyqwertyqwertyqwertyqwertyuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu» в файл представлена на рисунке 3. Было обработано ровно 80 символов (с первой буквы «у» в конце начинается 81 символ).

[illegible]

В ходе выполнения лабораторной работы были изучены представление и обработка символьной информации с использованием строковых команд. Для реализации поставленной задачи было написано консольное приложение на высокоуровневом языке программирования с использованием принципа встраивания ассемблерной части.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *lab4.cpp*

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

char input_string[81];
char output_string[321];

int main() {
    std::cout << "Student: Rymar Marya\nGroup: 1381\nTask: conversion
from decimal to binary\n";

    fgets(input_string, 81, stdin);
    input_string[strlen(input_string)] = '\\0';

    __asm {
        push ds
        pop es
        mov esi, offset input_string
        mov edi, offset output_string

    line:
        lodsb
        cmp al, '2'
        jne digit3
        mov ax, '01'
        stosw
        jmp final

    digit3:
        cmp al, '3'
        jne digit4
        mov ax, '11'
        stosw
        jmp final

    digit4:
        cmp al, '4'
        jne digit5
        mov ax, '01'
        stosw
        mov al, '0'
        stosb
        jmp final

    digit5:
        cmp al, '5'
        jne digit6
        mov ax, '01'
        stosw
```

```

        mov al, '1'
        stosb
        jmp final

digit6:
        cmp al, '6'
        jne digit7
        mov ax, '11'
        stosw
        mov al, '0'
        stosb
        jmp final

digit7:
        cmp al, '7'
        jne digit8
        mov ax, '11'
        stosw
        mov al, '1'
        stosb
        jmp final

digit8:
        cmp al, '8'
        jne digit9
        mov eax, '0001'
        stosd
        jmp final

digit9:
        cmp al, '9'
        jne last
        mov eax, '1001'
        stosd
        jmp final

last:
        stosb

final:
        mov ecx, '\0'
        cmp ecx, [esi]
        ; если был найден конец, то выход
        je lineEnd
        jmp line
lineEnd:
};

std::cout << output_string;
FILE* f;
fopen_s(&f, "output.txt", "w");
fwrite(output_string, sizeof(char), strlen(output_string), f);

return 0;
}

```