

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

Студент гр. 1303

Коренев Д.А.

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.

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`. Длина данного массива обусловлена тем, что в двоичной записи цифр 8 и 9 число символов равно 4.

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 смещение `input` присваивается в `esi`, а смещение `output` в `edi`.

Строка начинает обрабатываться с метки `line`. Команда `lodsb` отвечает за чтение байта из строки (копирует один байт из памяти по адресу `es:esi` в регистр `al`). С помощью команды `cmp` последовательно сравниваются считанный символ с другими десятичными символами: «2», «3», «4», «5», «6», «7», «8», «9» (переводить «1» в двоичную запись нет смысла, т.к. ее запись совпадает с записью в десятичной СС). 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` 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`.

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` она записывается в `output`.

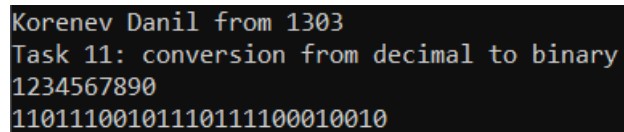
После всех замен и внесений исходных символов в выходной массив переходим к метке `next`. Если по смещению `esi` находится символ конца строки, то работа ассемблерного блока заканчивается. Выходной массив `output` будет выведен в консоль и записан в файл `output.txt`.

Исходный код программы см. в приложении А.

Тестирование.

Для проверки корректности работы программы было создано 4 теста:

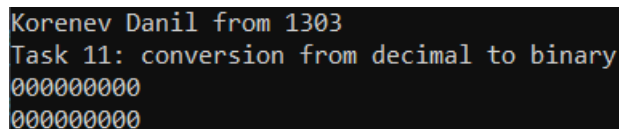
1. Вводится строка «1234567890». Результат в файле соответствует требуемому. Представлен на рисунке 1.



```
Korenev Danil from 1303
Task 11: conversion from decimal to binary
1234567890
11011100101110111100010010
```

Рисунок 1 тест номер 1

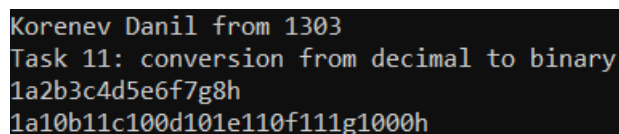
2. Вводится строка «000000000». Результат в файле соответствует требуемому. Представлен на рисунке 2.



```
Korenev Danil from 1303
Task 11: conversion from decimal to binary
000000000
000000000
```

Рисунок 2 тест номер 2

3. Вводится строка «1a2b3c4d5e6f7g8h». Результат в файле соответствует требуемому. Представлен на рисунке 3.



```
Korenev Danil from 1303
Task 11: conversion from decimal to binary
1a2b3c4d5e6f7g8h
1a10b11c100d101e110f111g1000h
```

Рисунок 3 тест номер 3

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lab4.cpp

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

char input[81];
char output[321];

int main() {
    std::cout << "Korenev Danil from 1303\nTask 11: conversion from
decimal to binary\n";
    fgets(input, 81, stdin);
    input[strlen(input)] = '\0';

    __asm {
        push ds
        pop es
        mov esi, offset input
        mov edi, offset output

line:
        lodsb
        cmp al, '2'
        jne symbol3
        mov ax, '01'
        stosw
        jmp next

symbol3:
        cmp al, '3'
        jne symbol4
        mov ax, '11'
        stosw
        jmp next

symbol4 :
        cmp al, '4'
        jne symbol5
        mov ax, '01'
        stosw
        mov al, '0'
        stosb
        jmp next

symbol5:
        cmp al, '5'
        jne symbol6
        mov ax, '01'
        stosw
        mov al, '1'
        stosb
```

```

        jmp next

symbol6:
    cmp al, '6'
    jne symbol7
    mov ax, '11'
    stosw
    mov al, '0'
    stosb
    jmp next

symbol7:
    cmp al, '7'
    jne symbol8
    mov ax, '11'
    stosw
    mov al, '1'
    stosb
    jmp next

symbol8:
    cmp al, '8'
    jne symbol9
    mov eax, '0001'
    stosd
    jmp next

symbol9 :
    cmp al, '9'
    jne letter
    mov eax, '1001'
    stosd
    jmp next

letter :
    stosb

next :
    mov ecx, '\0'
    cmp ecx, [esi]
    je end
    jmp line
end :
};

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

return 0;
}

```