**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: Изучение программирования обработки символьной информации с использованием команд пересылки строк**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8382 |  | Мирончик П.Д. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2019

**Формулировка задачи.**

Разработать программу обработки символьной информации, реализующую функции:

- инициализация (вывод титульной таблички с указанием вида преобразования и автора программы) - на языке высокого уровня (Pascal или Си);

- ввода строки символов, длиной не более Nmax (<=80), с клавиатуры в заданную область памяти - на языке высокого уровня;

- выполнение заданного в таблице 1 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 нa Pascal или Си принципу встраивания (in-line).

Вариант 10: Преобразование введенных во входной строке шестнадцатиричных цифр в двоичную СС, остальные символы входной строки передаются в выходную строку непосредственно.

**Ход работы**

Была реализована программа, принимающая на вход строку и заменяющая символы 0-1 и a-f шестнадцатеричной СС на соответствующие им числа в двоичной СС.

Программа считывает строку в коде C и передает адрес считанной строки и адрес строки, куда будет записан результат, в ассемблерный код. Там она посимвольно обрабатывается: если символ – цифра или одна из букв a,b,c,d,e,f, берет соответствующее ему число (отнимая ‘0’ или ‘a’) и переводит его в двоичную систему счисления побитовым сдвигом и умножением на 1.

Для работы со строками были использованы функции *lodsb* и *stosb* – соответственно для чтения и записи строк.

Для компиляции программы используется команда

*gcc -std=c99 -Wall -O0 -masm=intel main.c*

В связи с тем, что программа компилируется в 64-битной системе, для работы с адресами строк используются регистры rsi и rdi. Всего в программе (в ассемблерном коде) изменяется 4 регистра: rsi и rdi – адреса строк, rax используется для обработки символов (например, побитовый сдвиг) и rcx используется как счетчик стека – сколько цифр мы положили в стек при побитовом сдвиге (например, для цифры 10 стек будет содержать 0101).

В результате выполнения программы получаются следующие строки:

|  |  |
| --- | --- |
| **Входная строка** | **Выходная строка** |
| 1234 | 11011100 |
| ab4k | 10101011100k |
| 0 | 0 |

Как видно из результатов тестов, программа успешно справляется с поставленной задачей и преобразует символы из 16 системы счисления в 2-ю

**Выводы**

В ходе выполнения данной работы было изучено использование ассемблерных вставок, особенности работы со строками, перевод чисел из одной системы счисления в другую, реализация циклов на языке ассемблер.

**Приложение А. Код программы**

#include <stdio.h>

int main()

{

    char s\_in[81];

    char s\_out[321];

    puts("Mironchik Pavel, 8382: convert 16ns (0-9,a-h) symbols to 2ns");

    gets(s\_in);

    \_\_asm\_\_ (".intel\_syntax noprefix\n\t"

        "mov rsi, %0\n\t"

        "mov rdi, %1\n\t"

        "cld\n"

        "TRANSFORM:"

        "mov cx,0\n\t" // set up <cx> to 0

        "lodsb\n\t"

        "cmp al,'0'\n" // check 0..9

        "jl NOT\_NUMBER\n"

        "cmp al,'9'\n"

        "jg CHECK\_LOW\n"

        "sub al,'0'\n"

        "jmp TO\_BIN\n"

        "CHECK\_LOW:\n" // check a..f

        "cmp al,'a'\n"

        "jl NOT\_NUMBER\n"

        "cmp al,'f'\n"

        "jg NOT\_NUMBER\n"

        "sub al,'a'\n"

        "add al,10\n"

        "jmp TO\_BIN\n"

        "TO\_BIN:\n"

        "mov ah,al\n"

        "and ah,1\n"

        "add ah,'0'\n"

        "push ax\n"

        "inc cx\n"

        "shr al\n"

        "cmp al,0\n"

        "jg TO\_BIN\n"

        "WRITE:\n"

        "pop ax\n"

        "mov al,ah\n"

        "stosb\n"

        "dec cx\n"

        "cmp cx,0\n"

        "jg WRITE\n"

        "jmp TRANSFORM\n"

        "NOT\_NUMBER:\n"

        "stosb\n"

        "cmp al,0\n"

        "jg TRANSFORM\n"

        :

        : "r" (s\_in), "r" (s\_out)

        : "%rax","%ecx","%rsi","%rdi"

        );

    puts(s\_out);

}