**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. 0383 |  | Сабанов П.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**Цель работы.**

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 нa ЯВУ по принципу встраивания (in-line).

Вариант 10.

Задание:

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

**Ход работы.**

Было реализовано чтение строки из стандартного потока ввода, преобразование строки и вывод её из стандартного потока вывода.

Была реализована функция

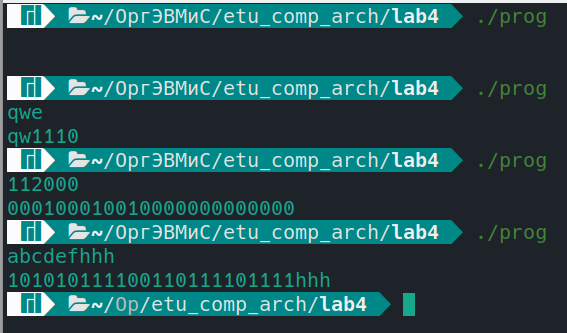
void convert(char\* in, char\* out, int\* len\_out), принимающая исходную строку, выходную строку и указатель на длину выходной строки, которую функция выставит по окончании работы.

Внутри функции была реализована ассемблерная вставка, осуществляющая обработку строки.

Внутри ассемблерной вставки программа пробегает по исходной строке и проверяет каждый её символ. Если символ является шестнадцатиричной цифрой, то программа вычисляет число, соответствующее символу этой цифры и последовательными двоичными сдвигами и битовой операцией and вычисляет и устанавливает во входной строке двоичное представление этой цифры.

Символы, не удовлетворяющие шестнадцатиричным обозначениям, программа пропускает.

**Пример работы программы.**



**Вывод.**

Была написана программа, читающая строку из стандартного потока ввода, преобразующая её и выводящая её в стандартный поток вывода.

Было реализовано преобразование строки с помощью ассемблерной вставки.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define max\_in\_len 80

#define max\_out\_len 1000

// 10 вариант

/\*

Преобразование введенных во входной строке шестнадцатиричных цифр в двоичную СС,

остальные символы входной строки передаются в выходную строку непосредственно.

\*/

void convert(const char\* in, char\* out) {

int i\_out = 0, i\_in = 0;

\_\_asm\_\_ \_\_volatile\_\_ (

".intel\_syntax noprefix\n"

"for:\n"

// char c = in[i\_in]

// compare c with 0

"mov eax, %0\n" // eax <- in

"add eax, %3\n" // eax = in + i\_in

"mov dl, [eax]\n" // char c = in[i\_in]

"cmp dl, 0\n" // c cmp 0

"je exit\_for\n" // in[i\_in] == 0

// if (cur >= '0' && cur <= '9' || cur >= 'A' && cur <= 'F' || cur >= 'a' && cur <= 'f')

"check\_if\_digit:\n"

"cmp dl, '0'\n" // check if >= '0'

"jl check\_if\_big\_alpha\n" // if < '0', then check if is big alpha

"cmp dl, '9'\n" // check if <= '9'

"jg check\_if\_big\_alpha\n" // if > '9', then check if is big alpha

// cur -= '0'

"sub dl, '0'\n"

// check passed, is xdigit

"jmp is\_xdigit\n"

"check\_if\_big\_alpha:\n"

"cmp dl, 'A'\n" // check if >= 'A'

"jl check\_if\_small\_alpha\n" // if < 'A', then check if is small alpha

"cmp dl, 'F'\n" // check if <= 'F'

"jg check\_if\_small\_alpha\n" // if > 'F', then check if is small alphg

// cur += 10 - 'A'

"sub dl, 'A'\n"

"add dl, 10\n"

// check passed, is xdigit

"jmp is\_xdigit\n"

"check\_if\_small\_alpha:\n"

"cmp dl, 'a'\n" // check if >= 'a'

"jl not\_is\_xdigit\n" // if < 'a', then not an xdigit

"cmp dl, 'f'\n" // check if <= 'f'

"jg not\_is\_xdigit\n" // if > 'f', then not an xdigit

// cur += 10 - 'a'

"sub dl, 'a'\n"

"add dl, 10\n"

// check passed, is xdigit

// c is an xdigit

"is\_xdigit:\n"

"add dword ptr %2, 4\n" // i\_out += 4

// eax <- out + i\_out - 1

"mov eax, %1\n"

"add eax, %2\n"

"dec eax\n"

// for(int j = 1; j <= 4; ++j)

// j in ecx

"mov ecx, 1\n"

"for\_j:\n"

// dh = '0' + (c & 1)

"mov dh, dl\n"

"and dh, 1\n"

"add dh, '0'\n"

// out[i\_out-j] = '0' + (cur & 1)

"mov [eax], dh\n"

// go to prev char

"dec eax\n"

// go to next bit in char

"shr dl, 1\n"

// increace counter

"inc ecx\n"

// if (j > 5) exit;

"cmp ecx, 4\n"

"jle for\_j\n"

"jmp end\_iteration\n"

// c is not an xdigit

"not\_is\_xdigit:\n"

// just copy symbol

// out[i\_out++] = c

"mov eax, %1\n" // eax <- out

"add eax, %2\n" // eax = out + i\_out

"mov [eax], dl\n" // out[i\_out] = c

"inc dword ptr %2\n" // ++i\_out

"end\_iteration:\n"

// go to next index in input string

"inc dword ptr %3\n" // ++i\_in

"jmp for\n"

"exit\_for:\n"

// setting \0 char at the end of the output string

"mov eax, %1\n"

"add eax, %2\n"

"mov byte ptr [eax], 0\n"

: // no outputs

:"m" (in), "m" (out), "m" (i\_out), "m" (i\_in)

:"eax", "ecx", "edx"

);

// int i\_out = 0;

// for (int i\_in = 0; in[i\_in] != '\0'; ++i\_in) {

// char cur = in[i\_in];

// if (cur >= '0' && cur <= '9' || cur >= 'A' && cur <= 'F' || cur >= 'a' && cur <= 'f') {

// if (cur >= 'a')

// cur += 10 - 'a';

// else if (cur >= 'A')

// cur += 10 - 'A';

// else

// cur -= '0';

// i\_out += 4;

// for (int j = 1; j <= 4; ++j) {

// out[i\_out-j] = '0' + (cur & 1);

// cur >>= 1;

// }

// } else {

// out[i\_out++] = cur;

// }

// }

// out[i\_out] = 0;

}

char buf1[max\_in\_len+1];

char buf2[max\_out\_len+1];

int main() {

fgets(buf1, max\_in\_len, stdin);

convert(buf1, buf2);

printf("%s", buf2);

}