Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «КСиС»

Тема: «Язык ассемблера. Обработка символьных данных»

Вариант №6

Выполнил:

Студент 2 курса

Группы ПО-7

Комиссаров А.Е.

Проверил:

Савицкий Ю. В.

2022

**Цель**: Выполнить преобразование текстовой строки на языке ассемблера.

**Задание:**

Условие: Текст содержит равное количество прописных латинских и прописных русских букв.

Преобразование 1: заменить каждую строчную латинскую букву соответствующей прописной буквой, а прописную – строчной.

Преобразование 2: циклически сдвинуть текст на K (константа) позиций влево без использования дополнительной памяти, реализовав следующий алгоритм: перевернуть подмассив из первых K символов; перевернуть оставшийся подмассив; перевернуть весь текст.

**Код программы:**

#include <iostream>

#include <Windows.h>

using namespace std;

//CP-1251 Кодировка:

//65-90 - ПРОПИСНЫЕ ЛАТИНСКИЕ буквы

//192-223 - ПРОПИСНЫЕ РУССКИЕ буквы

//97-122 - строчные латинские буквы

//224-255 - строчные русские буквы

int condition(char\* str, int len) { //Это функция на проверку строки, для того чтобы знать, какое из двух условий использовать.

int checker = 0; \_asm{

mov eax, checker

mov esi, str //указатель на текст

mov ecx, len //счётчик цикла ecx будет равен длине строки

xor ebx, ebx //очищаем edx (счётчик прописных латинских букв)

xor edx, edx //очищаем ebx (счётчик прописных русских букв)

jmp start

//собственно проверка условия

start: mov al, [esi]

cmp al, 65 //является ли символ буквой или знаком препинания?

jb nxt //знак препинания - переход в nxt

jge latin //буква - переход в latin

nxt : inc esi //переходим на следующий символ

loop start //начинаем всё заново если ecx (счётчик цикла) не ноль

jmp check //если мы прошли всю строку (ecx = 0), заканчиваем проверку

latin : cmp al, 90 //является ли символ латинским прописным?

jg rus1 //нет - переход в rus1 (начинаем проверку на русский символ)

jbe inceng //да - переход в inceng (увеличим счётчик ebx)

rus1 : cmp al, 192 //является ли символ русским прописным (1 часть)?

jge rus2 //возможно - переход в rus2 (заканчиваем проверку на русский символ)

jb nxt //нет - переход в nxt

rus2 : cmp al, 223 //является ли символ русским прописным (2 часть)?

jbe incrus //да - переход в incrus (увеличим edx)

jg nxt //нет - переход в nxt

inceng: inc ebx //увеличиваем счетчик прописных латинских букв

jmp nxt

incrus: inc edx //увеличиваем счетчик прописных русских букв

jmp nxt

check : cmp ebx, edx //проверка на равенство прописных русских и латинских букв

je rule1 //если равны переход в rule1

jne rule2 //если не равны переход в rule2

rule1 : mov eax, 1 //Checker = 1, будем выполнять первое правило

mov checker, eax

jmp ext

rule2 : mov eax, 0 //Checker = 0, будем выполнять второе правило

mov checker, eax

jmp ext

ext : //выход

} return checker;

}

char\* invert(char\* str, int K, int strOffset) {

\_asm{

jmp start

revrse: mov bx, 2 //делим eax на 2 части

div bx //чтобы поменять половину строки с другой

mov ecx, eax //поместим в ecx значение eax

rev : mov al, [edi]

movsb //скопируем байт из edi в esi

dec esi //уменьшим esi

mov[esi], al

dec esi //уменьшим esi

dec ecx //уменьшим ecx

cmp ecx, 0

jne rev //если не равно 0 переход в rev

ret //выход из подпрограммы

start : mov ecx, str

mov eax, K //установим количество символов, которые нужно перевернуть

mov edx, strOffset //установим смещение

cmp edx, 0

jne m1 //если не равно переход в m1

m1 : inc ecx //сместим строку на 1 символ вправо

dec edx //уменьшим счетчик смещения на 1

cmp edx, 0

jne m1 //если не равно переход в m1

jmp m2 //переход в m2

m2 : mov edi, ecx //установим указатель на элемент строки, с которого начать смещение

mov esi, ecx //установим указатель на элемент строки, с которого начать смещение

add esi, eax

dec esi

call revrse //вызов смещения

jmp ext

ext :

} return str;

}

char\* firstRule(char\* str, int len) {

\_asm

{

jmp start

//Заменить каждую строчную ЛАТИНСКУЮ букву соответствующей прописной буквой, а прописную – строчной

start: mov esi, str //записываем указатель на текст в регистр esi

mov ecx, len //устанавливаем счётчик цикла равным длине строки

jmp checkRule1

checkRule1 :

mov al, [esi]

cmp al, 65 //входит ли буква в список латинских прописных (часть 1)?

jb nxt //нет - если меньше переход в nxt

jge h2 // да - если больше или равно переход в h2

nxt : inc esi //переход на след. символ

loop checkRule1

jmp ext

h2 : cmp al, 90 //буква является прописной латинской (часть 2)?

jg h3 //нет - если больше переход в h3

jbe h4 //да - заменяем на строчную

h4 : add al, 32 //замена прописной латинской на строчную букву

mov[esi], al

jmp nxt

h3 : cmp al, 97 //проверка: входит ли буква в список латинских строчных?

jge h5 //если больше или равно переход в h5

jb nxt //если меньше переход в h1

h5 : cmp al, 122

jbe h6 //если меньше или равно переход в h6

jg nxt //если больше переход в h1

h6 : sub al, 32 //замена строчной латинской на прописную букву

mov[esi], al

jmp nxt

ext :

} return str;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int maxLength = 100; int realLength; //Максимальная длина строки - 100 символов

char\* mystring = new char[maxLength]; //выделяем память для введённого

cout << "Enter a string: " << endl; //вводим строку

for (int i = 0; i < maxLength; i++) { //ввод, пока не найдём точку

cin >> mystring[i]; if (mystring[i] == '.') {mystring[i] = NULL;break;}

}

realLength = strlen(mystring); //определим длину строки

if (condition(mystring, realLength) == 1) {

cout << "First rule: \n"; //После проверки выполняется либо это правило

cout << firstRule(mystring, realLength) << endl;

}

else {

cout << "Second rule: \n"; //либо это правило (второе)

int K = rand() % (realLength - 1) + 1; //выберем случайное число для сдвига

int secondPart = realLength - K; //это будет длина второй сдвигаемой части

invert(mystring, K, 0); //переворачиваем первую часть

invert(mystring, secondPart, K); //потом вторую, не трогая первую

cout << "Moving symbols (K = " << K << "):\n"; //вывод результата

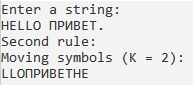
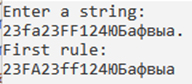
cout << invert(mystring, realLength, 0) << endl;//(нужно полностью перевернуть строку)

}

return 0;

}

**Результат выполнения:**

** **

Вывод: написали на ассемблере программу обработки текста