Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 2 семестр

По дисциплине: «КСиС»

Выполнил:

Студент 2 курса

Группы ПО-6(1)

Борисюк Р.С.

Проверил:

Бойко Д.О.

2022

**Лабораторная работа #2**

**ЯЗЫК АССЕМБЛЕРА. ОБРАБОТКА СИМВОЛЬНЫХ ДАННЫХ**

**Постановка задачи**

Дан текст – непустая последовательность не длиннее ста символов. Признаком конца ввода является точка, в сам текст точка не входит.

Проверить, удовлетворяет ли текст заданному условию. Если условие выполнено, преобразовать текст по одному правилу, в противном случае – по другому правилу. Преобразованный текст напечатать.

Проверяемое условие и правила обработки текста определяются конкретным вариантом задания.

Если введенная последовательность символов не является текстом, преобразовывать ее не нужно, а следует напечатать соответствующее сообщение.

Ввод текста, проверка условия, обработка текста и печать результата должны выполняться последовательно, отдельными частями программы.

**Вариант 3**

**Проверяемое условие:**

В тексте больше латинских букв, чем цифр

**Первое правило преобразования:**

Заменить каждую латинскую букву симметричной ей в алфавите (A ↔ Z, b ↔ y, ...)

**Второе правило преобразования:**

Поменять местами каждый символ на нечетной позиции и следующий за ним символ текста, если таковой имеется

**Код программы:**

section .text

global \_start

\_start:

call counter ;вызов функции подсчёта символов

cmp dl, dh ;стравнивает dl - количество символов

;и dh - количество цифр

ja first ;если dl > dh, то перепрыгнуть в точку fitst

call second\_rule ;иначе пойдёт дальше и вызовет обработку по второму правилу

jmp output ;после обработки, перейти в точку output

first:

call first\_rule ;вызов обработки по первому правилу

output:

mov edx, len ;помещаем в edx длину строки

mov ecx, s1 ;помещаем в ecx саму строку

mov ebx, 1 ;дескриптор файла (stdout)

mov eax,4 ;номер системного вызова (sys\_write)

int 0x80 ;вызов ядра

mov eax,1 ;номер системного вызова (sys\_exit)

int 0x80 ;вызов ядра

counter:

mov ecx, len - 1 ;помещает в ecx длину строки

mov esi, s1 ;в esi помещаем строку, из которой будем извлека по символу

mov edx, 0 ;из edx будем использовать регистры dl и dh,

;для подсчёта количества символов и цифр соответсвенно.

;Поэтому обнуляем его в начале

count\_loop:

lodsb ;выгружаем из строки esi символ, в al

cmp al, '.' ;условие для выхода и цикла

je exit\_loop ;если al = '.', переместиться к выходу процедуры

push ecx ;поместим ecx в стек, так как он будет изменять в методах scasb

mov ecx, 10 ;помещаем в ecx длину строки Numbers.

mov edi, Numbers ;помещаем в edi строку из цифр,

;в которой будем искаться символ из al

repne scasb ;вызов поиска

je inc\_num ;если нашёл, то перепрыгнуть в inc\_num

or al, 20h ;используем логическое или,

;если al буква, то его регист уменишится

mov ecx, len\_letters ;помещаем в ecx длину строки Letters.

mov edi, Letters ;помещаем в edi строку из сиволов,

;в которой будем искаться символ из al

repne scasb ;вызов поиска

je inc\_let ;если нашёл, то перепрыгнуть в inc\_let

next\_iter:

pop ecx ;извлекаем в ecx, старое значение из стека

loop count\_loop ;вызываем цикл сново, пока ecx != 0

exit\_loop:

ret ;возврат процедуры

inc\_let:

inc dl ;увеличиваем количество символов на 1

jmp next\_iter ;перепрыгиваем в конец цикла counter\_loop

inc\_num:

inc dh ;увеличиваем количество цифр на 1

jmp next\_iter ;перепрыгиваем в конец цикла counter\_loop

first\_rule:

mov ecx, len - 1 ;поместить длину строки

mov esi, s1 ;поместить строку источник

first\_loop: ;загрузить символ из esi, в al

lodsb

cmp al, '.' ;условие для выхода и цикла

je first\_exit\_loop ;если al = '.', переместиться к выходу процедуры

mov ah, al ;сохраняем симвлол al, в ah

or al, 20H ;если al - символ, уменьшится регистор

push ecx ;сохраняем ecx, в стек

mov ecx, len\_letters ;помещаем в ecx длину строки Letters.

mov edi, Letters ;помещаем в edi строку из сиволов,

;в которой будем искаться символ из al

repne scasb ;вызов поиска

je replace ;если нашёл, то перепрыгнуть в replace

next\_first\_iter:

pop ecx ;извлекаем в ecx, старое значение из стека

loop first\_loop ;вызываем цикл сново, пока ecx != 0

first\_exit\_loop:

ret ;выход их процедуры

replace:

mov dl, 'z' ;сохраняем в dl, символ 'z'

or al, 20H ;уменьшаем регистор al

sub al, 'a' ;отнимаем из 'a', al. Так получаем номер буквы

sub dl, al ;отнимаем из dl, al.

;Получаем противоположный символ для al

and ah, 20H ;побитовое 'И' ah - изначальное

;значение символа al, с 20H == 32.

cmp ah, 20H ;если ah == 20H, это значит что

;символ изначально был младшего региста

mov [esi - 1], dl ;сохраняем значение dl, в строку

je next\_first\_iter ;если младший регист, то вернуться в конец цикла

sub dl, 20H ;если был старший, то отнять из dl 20H,

;чтобы получить старший регистор

mov [esi - 1], dl ;сохраняем значение dl, в строку

jmp next\_first\_iter ;вернуться в конец цикла

second\_rule:

mov esi, 0 ;будем использовать esi,

;для подсчёта нужного нам индекса в строке

second\_loop:

mov al, [s1 + esi] ;помещаем в al, нечётный элемент строки

mov ah, [s1 + esi + 1] ;в ah - чётный

mov [s1 + esi], ah ;меняем местами

mov [s1 + esi + 1], al ;символы в строке

add esi, 2 ;увеличиваем esi на 2

cmp esi, len - 1 ;если esi < len - 1

jb second\_loop ;вызвать цикл сново

ret

section .data

s1 db 'A1B1C1D1AA1.', 0xa

len equ $-s1

Letters db 'abcdefghijklmnopqrstuvwxyz'

len\_letters equ $-Letters

Numbers db '0123456789'

**Результат программы:**

**Исходная строка:** 121212121212121212

**Результат:** 

**Исходная строка:** ABCDEFG.

**Результат:** 

**Вывод:** написал обработку строк на ассемблере.