

Министерство образования Республики Беларусь
Учреждение Образования
«Брестский Государственный Технический Университет»
Кафедра ИИТ

Лабораторная работа №2
По дисциплине АПОЭВМ за VI семестр
Тема: «Язык ассемблера. Обработка символьных данных»

Выполнил:

Студент 3-го курса

Группы АС-56

Ровенский Д.В.

Проверил:

Булей Е.В.

Брест 2022

Цель работы: научиться обрабатывать символьные данные на языке ассемблера

Ход работы

Вариант 1

Задание

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.

Содержание варианта:

Проверяемое условие.

- 1) Текст оканчивается прописной латинской буквой, которая больше в тексте не встречается. (с.к.)

Первое правило преобразования.

- 1) Заменить каждую ненулевую цифру соответствующей ей строчной буквой латинского алфавита (1 → a, 2 → b и т.д.).

Второе правило преобразования.

- 1) Перевернуть текст, не используя дополнительную память.

Текст программы:

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

const int bg_color = system("color f4");
```

```

int check_dot(char* c1) { //Проверка, есть ли в тексте признак окончания ('.')
    int ret = 0;
    __asm{
        mov     eax, c1
        mov ecx, 99
        mov bl, '.'
        start_cycle:
            cmp byte ptr[eax + ecx], bl
            je brk
        loop start_cycle
        jmp end

        brk :
            mov ret, 1
            jmp end
        end :
    }
    return ret;
}

int main_check(char* c1, char* c2) { //Проверка условия
    int ret2 = 0;
    __asm{
        mov     ebx, c1           //ebx = str
        mov ecx, 99
        mov al, '.'
        mov edx, c2           //edx = alphabet

        start_cycle :
            cmp byte ptr[ebx + ecx], al
            je brk
        loop start_cycle

        brk :
            mov ah, byte ptr[ebx + ecx - 1]           //ah - последний символ строки
            mov ecx, 26                               //т.к. в англ. алфавите 26 букв

            start_cycle2 :
                cmp byte ptr[edx + ecx], ah           //Ищет совпадения последнего символа
                                                        //строки с
буквами алфавита
            loop start_cycle2

            jmp end                                     //если совпадений нет,
то идет в конец

        brk2 :
            mov al, ah
            mov di, bx
            mov ecx, 99

            start_cycle3 :
                repne scasb           //Найти байт не равный al в блоке из ecx
                jcxz end             //байт по адресу (e)di, если не находит- переходит
в конец
            inc ret2                 //если находит - увеличивает ret2 на единицу
            jmp start_cycle3
        end :
    }
    return ret2;
}

```

```

int do1(char* c1) { //Первое преобразование
    int ret3 = 0;
    __asm{
        mov     ebx, c1                                //     ebx
        = str
        mov ecx, 99
        start_cycle:
            cmp byte ptr [ebx + ecx], '1'
            je case1
            cmp byte ptr [ebx + ecx], '2'
            je case2
            cmp byte ptr [ebx + ecx], '3'
            je case3
            cmp byte ptr [ebx + ecx], '4'
            je case4
            cmp byte ptr [ebx + ecx], '5'
            je case5
            cmp byte ptr [ebx + ecx], '6'
            je case6
            cmp byte ptr [ebx + ecx], '7'
            je case7
            cmp byte ptr [ebx + ecx], '8'
            je case8
            cmp byte ptr [ebx + ecx], '9'
            je case9
        loop start_cycle
        jmp end

        case1 :
            mov byte ptr [ebx + ecx], 'a'
            jmp start_cycle
        case2 :
            mov byte ptr [ebx + ecx], 'b'
            jmp start_cycle
        case3 :
            mov byte ptr [ebx + ecx], 'c'
            jmp start_cycle
        case4 :
            mov byte ptr [ebx + ecx], 'd'
            jmp start_cycle
        case5 :
            mov byte ptr [ebx + ecx], 'e'
            jmp start_cycle
        case6 :
            mov byte ptr [ebx + ecx], 'f'
            jmp start_cycle
        case7 :
            mov byte ptr [ebx + ecx], 'g'
            jmp start_cycle
        case8 :
            mov byte ptr [ebx + ecx], 'h'
            jmp start_cycle
        case9 :
            mov byte ptr [ebx + ecx], 'i'
            jmp start_cycle

        end :
    }
    return ret3;
}

```

```

int do2(char* c1) { //Второе преобразование
    int ret3 = 0;

    __asm{
        mov     eax, c1
        mov ecx, 99
        mov dl, '.'

        start_cycle:
            cmp byte ptr[eax + ecx], dl
            je brk2
        loop start_cycle

        brk2 :
        start_cycle2:
            dec     ecx
            cmp ecx, 0
            je brk
            mov     bh, byte ptr[eax]
            mov     bl, byte ptr[eax + ecx]
            mov     byte ptr[eax], bl
            mov     byte ptr[eax + ecx], bh

            inc     eax
            loop start_cycle2
        brk :
    }
    return ret3;
}

int main() {
    char str1[100];
    char alph[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    printf("Input string: ");
    std::cin.get(str1, 99);
    if (check_dot(str1) == 0) { //Проверка, есть ли в тексте признак окончания ('.')
        printf("Not text\n");
        return main();
    }
    else {
        printf("This is text\n");
    }
    if (main_check(str1, alph) == 1) { //Проверка условия
        printf("If is true\n");
        do1(str1); //Первое преобразование
    }
    else {
        printf("If is not true\n");
        do2(str1); //Второе преобразование
    }
    printf("Output string: %s\n", str1);
    return 0;
}

```

Результат выполнения программы:

```
Консоль отладки Microsoft Visual Studio
Input string: Hehe haha hihi 123456789.
This is text
If is not true
Output string: 987654321 ihih ahah eheH.

D:\uni\3.2\APO\lab 2\Project1\Debug\Project1.exe (процесс 11088) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
```

```
Консоль отладки Microsoft Visual Studio
Input string: hehe haha hihi 123456789Q.
This is text
If is true
Output string: hehe haha hihi abcdefghiQ.

D:\uni\3.2\APO\lab 2\Project1\Debug\Project1.exe (процесс 22092) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Вывод: Я научился обрабатывать символьные данные на языке ассемблера