МИНЕСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧЕРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«Брестский государственный технический университет»
Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №2
По дисциплине «Аппаратно-программное обеспечение ЭВМ и сетей»
За 6 семестр
Тема: «Ассемблерные вставки»

Выполнила:
студентка 3 курса
группы АС-56
Карпенко М.В.

Проверил:
Булей Е.В.

Брест 2022

Вариант 4

Написать ассемблерную вставку, реализующую следующую обработку строки: согласно варианту. Оформить ее в виде отдельной функции. Реализовать данную обработку строки также в виде функции на C++. Сравнить быстродействие обоих вариантов. В отчете отразить выводы. Для разработки использовать MS Visual Studio.

4. Даны 2 строки. Совместить четные символы одной строки с нечентными другой.

```
#include <iostream>
//using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");
    const int size = 100;
    char stroka[size];
    std::cout << "Input string:"<<std::endl;
    std::cin.getline(stroka, size);
    int sizer = strlen(stroka);
    int top;
    int sdvig = 3;
    char* stroka2 = new char[size];
    __asm
    {
        mov ebx, 0;                             //Счётчик для подсчёта английских букв в регистре
ebx
        mov ecx, sizer;                         //Записываем в регист счётчик размер нашей строки
        lea esi, [stroka];                      //Закидываем адрес в регистр-входной источник

    looper:                                     //Начало тела цикла
        lodsb;                                  //Считываем первую букву
        cmp al, 46;                             //Сравниваем с точкой, если точка, то завершаем
подсчёт
        jz endstep;                             //Прыжок в конец опервции, с завершением подсчёта
        cmp al, 32;                             //Проверка на пробел, если пробел, в конец
перемещаемся для следующей итерации
        jz ender;                               //Прыжок в конец цикла, для последующего
итерирования
        cmp al, 90;                             //Проверка на английские буквы
        jg check_lowercase;                     //Чек верхнего регистра
        jle check_uppercase;                    //Чек нижнего регистра

    check_uppercase:                            //Проверка >=65(A) Плюсуем инкремент счётчик ebx
        cmp al, 65;
        jge increment;
        jmp ender;                              //Иначе прыгаем в конец цикла

    check_lowercase:                            //Проверка >=(a) Проверка двойная, т.к. есть ещё и
верхняя граница 122(z)
        cmp al, 97;                             //Проверка на нижнюю границу
        jge check_lowercase2;
        jmp ender;

    check_lowercase2:                           //Проверка на верхнюю границу
        cmp al, 122;
        jle increment;                          //Если сравнения прошли, инкрементим ebx-счётчик
        jmp ender;                              //Прыжок в конец цикла

    increment:                                  //Операция инкремента счётчика
        inc ebx;

    ender:
```

```
        loop looper;                            //Конец цикла подсчёта английских букв

    endstep:
        mov top, ebx;                           //Запихваем кол-во наших букв в top для вывода

        mov ecx, sizer;                         //Обновляем счётчик для следующего шага
        mov eax, top;                           //Записываем сколько букв
        cmp eax, 3;                             //Проверка сколько букв, больше 3 выполняем первое
действие иначе второе
        jge function_one;
        jl function_two;

    function_one:                               //Первая операция
        mov ebx, 0;                             //Обновляем счётчик символов, в цикле будем
считывать до точки и формировать выходную строку
        lea esi, [stroka];                      //Входную строку в регистр-источник
        mov edi, stroka2;                       //Выходной адрес помещаем в выходной-регистр
    looper1:                                    //Цикл для записи символов, при этом с необходимой
модификацией символов
        lodsb;                                  //Считываем символ
        cmp al, 46;                             //Проверка на точку, да - в конец проги отлетаем
        jz endstep1;                            //Конец цикла
        cmp al, 90;                             //Нам нужны только буквы верхнего регистра поэтому
<=90(Z) инначе берём следующую букву
        jg next_letter;                         //След. буква
        jle check_uppercase1;                   //Проверка на нижнюю границу

    check_uppercase1:                           //Проверка на нижнюю границу
        cmp al, 65;                             //Проверка >=65(A), да - смещаем на один символ по
алфавиту
        jge letter_plus_one;                    //Прыжок на смещение
        mov[edi], al;                           //Записываем в выходной-регист, символ если он не
является верхним регистром
        inc edi;                                //Адрес следующего символа делаем
        jmp next;                               //Следующии шаг

    letter_plus_one:                            //Смещение по алфавиту на 1
        add al, 1;                              //Плюсуем к нашему считанному символу 1
        cmp al, 90;                             //Если больше 90, то пишем первую букву
        jg letter_back;                         //Операция записи первой буквы
        mov[edi], al;                           //Запись символа измененного
        inc edi;                                //След. символ
        jmp next;                               //Цикл конец

    letter_back:                                //Запись вместо 91 символа, букву A
        mov al, 65;
        mov[edi], al;                           //Запись
        inc edi;                                //След. символ
        jmp next;                               //Цикл конец

    next_letter:                                //Запись символа и следующии устанавливаем
        mov[edi], al;
        inc edi;
        jmp next;

    next:                                       //Конец цикла с подсчётом количества необходимых
сиволов для записи в выходную строку
        inc ebx;                                //Инкрементим счётчик ebx
        loop looper1;                           //Цикл

    endstep1:                                   //Конец операции подсчета и модиикации
        xor al, al;                             //С помощью исключающего или получаем ноль-символ
для конца строки
        mov[edi], al;                           //Запись его
        inc edi;
        jmp ender_last;                         //Завершаем программу

    function_two:                               //Втрорая операция
        mov ecx, sizer;                         //Записываем исходной строки длину
```

```
        lea esi, stroka;                        //Адрес источник записываем
        mov ebx, 0;                             //Начинаем счётчик считанных символов
    looper3:
        lodsb;                                  //Грузим символ из источника
        cmp al, 46;                             //Проверяем на точку
        jz next_step;                           //Когда точка отлетаем в основной операцию

        inc ebx;                                //Инкрементим ebx для определения сколько надо
считать
        loop looper3;                           //Цикл

    next_step:                                  //Непосредственно операция сдвига
        mov eax, sdvig;                         //Записываем в eax сколько сдвиг в лево
        lea esi, [stroka + eax];                //В источник пихаем строку с +3 адресом, на 3 вправо
        mov edi, stroka2;                       //Выходная строка
        mov ecx, ebx;                           //Из ebx получаем сколько символов необходимо
считать
        sub ecx, sdvig;                         //Отнимаем от этого количества на сдвиг
        rep movsb;                              //Записываем строковой командой повторющейся
        lea esi, [stroka];                      //Обновляем источник, теперь с первого символа
        mov ecx, eax;                           //Записываем, сколько символов считаем, возьмём eax,
т.к. мы отнимали его
        rep movsb;                              //Записываем в конец оставшиеся символы
        xor al, al;                             //Нуль для конца строки
        stosb;                                  //Его запись

    ender_last:
        mov ecx, 0;
    }
    std::cout << "Проверка : более 3 букв английского алфавита" << std::endl;
    std::cout << "Верно    : Заменить каждую прописную латинскую букву следующей за ней по
алфавиту.\n" << "Не верно : Сдвиг букв на " << sdvig << " позиции " << std::endl;
    std::cout << "\nВходная строка:\n" << stroka << " - найдено " << top << " букв английского
алфавита" << std::endl;
    std::cout << "\nВыходная строка:\n" << stroka2 << "" << std::endl;
}
```