Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Компьютерные системы и сети»

Тема: «Разработка ассемблерной вставки»

Вариант №6

Выполнил:

Студент 2 курса

Группы ПО-7

Комиссаров А.Е.

Проверил:

Савицкий Ю.В.

2021

**Цель работы:** изучить базовые понятия языка ассемблера, научиться работать с ассемблерными вставками языка программирования C++.

**Задание:**

Написать ассемблерную вставку, реализующую обработку строки

согласно варианту. Оформить ее в виде отдельной функции.

Реализовать данную обработку строки также в виде функции на С++.

Сравнить быстродействие обоих вариантов. В отчете отразить выводы.

Вариант №6 : **Нечетные символы заменить на +.**

**Код программы:**

#include<stdio.h>

#include<iostream>

#include<iomanip>

#include<chrono>

**void RepSymASM(char\* inputstr) {** *//замена символов средствами ассемблерной вставки*

**\_\_asm {**

**mov edx, inputstr** *//помещаем начало строки в EDX*

**replacingF:** *//первая (First) стадия*

**mov [edx], '+'** *//заменяем символ строки на* **+** *по адресу EDX*

**inc edx** *//переходим на следующий символ*

**cmp [edx], 0** *//сравниваем символ с символом-терминатором*

**je breakReplacing** *//если символ-терминатор, заканчиваем замену*

**jmp replacingS** *//если не символ-терминатор, переход во вторую стадию*

**replacingS:** *//вторая (Second) стадия*

**inc edx** *//не трогая текущий, переходим на следующий символ*

**cmp [edx], 0** *//сравниваем символ с символом-терминатором*

**je breakReplacing** *//если символ-терминатор, заканчиваем замену*

**jmp replacingF** *//если не символ-терминатор, переход в первую стадию*

**breakReplacing:** *//конец замены*

**}**

**}**

**void RepSymCPP(char\* inputstr) {** *//замена символов средствами C++*

**for (int x = 0; x < strlen(inputstr); x++) {**

*//strlen(inputstr) – длина строки*

**inputstr[x] = '+';** *//заменить символ на +*

**x++;** *//пропустить следующий символ*

**}**

**}**

**void main() {**

**char My\_StringASM[] = "Hello world";** *//строка для ассемблера*

**char My\_StringCPP[] = "Hello world";** *//строка для С++*

*//запуск отсчёта (ассемблер)*

**auto startASM = std::chrono::high\_resolution\_clock::now();**

**RepSymASM(My\_StringASM);** *//Вычисления (ассемблер)*

*//остановка отсчёта и вычисление разницы*

**auto stopASM = std::chrono::high\_resolution\_clock::now();**

**auto durationASM = std::chrono::duration\_cast<std::chrono::nanoseconds>(stopASM - startASM);**

*//запуск отсчёта (с++)*

**auto startCPP = std::chrono::high\_resolution\_clock::now();**

**RepSymCPP(My\_StringCPP);** *//Вычисления (С++)*

*//остановка отсчёта и вычисление разницы*

**auto stopCPP = std::chrono::high\_resolution\_clock::now();**

**auto durationCPP = std::chrono::duration\_cast<std::chrono::nanoseconds>(stopCPP - startCPP);**

*//вывод результатов (ассемблер)*

**std::cout << "(ASM) New String: ";**

**for (int x = 0; x < sizeof(My\_StringASM) / sizeof(char); x++) {**

**std::cout << My\_StringASM[x];**

**} std::cout << "\n";**

**std::cout << "(ASM) Elapsed time: "**

**<< durationASM.count()**

**<< " nanoseconds"**

**<< std::endl;**

*//вывод результатов (с++)*

**std::cout << "\n(CPP) New String: ";**

**for (int x = 0; x < sizeof(My\_StringCPP) / sizeof(char); x++) {**

**std::cout << My\_StringCPP[x];**

**} std::cout << "\n";**

**std::cout << "(CPP) Elapsed time: "**

**<< durationCPP.count()**

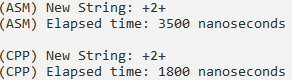
**<< " nanoseconds"**

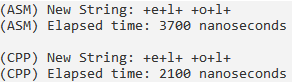
**<< std::endl;**

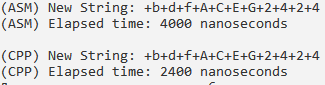
**system("pause");** *//пауза*

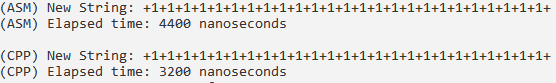
**}**

**Результат работы программы:**

**1.** Строка «123»

**2.** Строка «Hello world »

**3.** Строка «abcdefgABCDEFG12341234»

**4.** Строка «111111111111111111111111111111111111111111111111111»

**Вывод:** я изучил базовые понятия языка ассемблера, научился работать с ассемблерными вставками языка программирования C++.

Язык ассемблера обладает возможностью максимально полного использования всех особенностей аппаратной платформы, что позволяет, теоретически, писать самый быстрый и компактный код из всех возможных для данного процессора. Искусный программист, как правило, способен значительно оптимизировать программу по сравнению с транслятором с языка высокого уровня. Однако, в случае этой лабораторной работы, программа выполняется приблизительно в два, если не в три раза быстрее при использовании стандартных средств С++.