**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра «Информатика и программное обеспечение»**

**ЛАБОРАТОРНАЯ РАБОТА № 2.5**

**ПО ДИСЦИПЛИНЕ**

**«Архитектура вычислительных систем»**

Выполнил студент гр. О-21-ИВТ-1-по-Б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шпаков В.В,

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_ к.ф.-м.н., доц. Дмитроченко О.Н.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

**Брянск 2024**

# Цель работы

Целью данной программы является исследование эффективности и различий в производительности при использовании SSE (Streaming SIMD Extensions) инструкций для умножения векторов различных типов данных. Рассматриваются два типа данных: double и char.

## Тип double

Для типа double программа оценивает производительность умножения векторов с применением OpenMP и интринсиков SSE. Сравниваются времена выполнения операций без использования SSE и с использованием SSE, а также выводятся результаты умножения.

## Тип char

Для типа char программа сравнивает производительность и результаты умножения при использовании SSE и без использования SSE для массивов типа char. Различия в результатах объясняются особенностями обработки переполнений при использовании SSE. Выводится время выполнения и результаты умножения для обоих случаев.

# Теоретические сведения

## **SSE (Streaming SIMD Extensions)**

SSE представляет собой набор инструкций SIMD (Single Instruction, Multiple Data), предназначенных для одновременной обработки нескольких данных. Интринсики SSE в языке программирования C++ обеспечивают прямое использование этих инструкций в коде.

## Тип double

Инструкции SSE для типа double предоставляют параллельные операции с двойной точностью, что может повысить эффективность выполнения определенных вычислений.

## Тип char

При использовании SSE для типа char важно учитывать особенности обработки насыщенного умножения и переполнений, которые могут отличаться от стандартного умножения типа char.

## Вывод

Программа выполняет сравнение времени выполнения и результатов умножения с использованием и без использования SSE для двух различных типов данных. Это позволяет продемонстрировать потенциальное улучшение производительности при использовании SIMD-инструкций.

# Задания

## Задание 1 (double)

Код написан на C++ и использует OpenMP и интринсики SSE для умножения векторов, что может повысить производительность за счет параллельного выполнения и векторизации операций.

*Листинг 1*

|  |
| --- |
| #include <omp.h>  #include <iostream>  #include <emmintrin.h>  using namespace std;  int main()  {  setlocale(LC\_ALL, "rus");  const int size = 4;  double a[size] = { 10.0, 20.0, 30.0, 40.0 };  double b[size] = { 1.5, 2.5, 3.5, 4.5 };  double c[size];  // Выполнение операций без SSE для типа double  double start\_time = omp\_get\_wtime();  for (int i = 0; i < 1000000; i++) {  for (int j = 0; j < size; j++) {  c[j] = a[j] \* b[j];  }  }  double elapsed\_time = omp\_get\_wtime() - start\_time;  cout << "Для типа double без SSE: " << elapsed\_time << " сек." << endl;  // Вывод времени выполнения для double без SSE  for (int i = 0; i < size; i++) {  cout << "c[" << i << "] = " << c[i] << "\n";  }  cout << endl;  // Выполнение операций с SSE для типа double  start\_time = omp\_get\_wtime();  \_\_m128d xmm\_a, xmm\_b, xmm\_c;  for (int i = 0; i < 1000000; i++) {  for (int j = 0; j < size; j += 2) {  xmm\_a = \_mm\_loadu\_pd(&a[j]);  xmm\_b = \_mm\_loadu\_pd(&b[j]);  xmm\_c = \_mm\_mul\_pd(xmm\_a, xmm\_b);  \_mm\_storeu\_pd(&c[j], xmm\_c);  }  }  elapsed\_time = omp\_get\_wtime() - start\_time;  // Вывод времени выполнения для double с SSE  cout << "Для типа double с SSE: " << elapsed\_time << " сек." << endl;  for (int i = 0; i < size; i++) {  cout << "c[" << i << "] = " << c[i] << "\n";  }  return 0;  } |

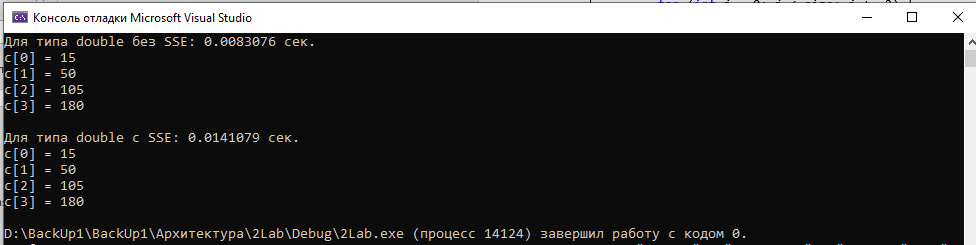


Рис. 1. Результат выполнения

## Задание 2 (char)

Программа сравнивает производительность и результаты умножения при использовании SSE и без использования SSE для массивов типа char. Различия в результатах объясняются различием в обработке переполнений при использовании SSE.

*Листинг 2*

|  |
| --- |
| #include <immintrin.h>  #include <omp.h>  #include <iostream>  #include <locale.h>  using namespace std;  int main()  {  setlocale(LC\_ALL, "ru");  // Для типа char  char a[16] = { 10, 20, 30, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };  char b[16] = { 1, 2, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };  char c[16];  // Выполнение операций без SSE для типа char  double t = omp\_get\_wtime();  for (int i = 0; i < 1000000; i++) {  // Умножение элементов массивов a и b и сохранение результата в массив c  for (int j = 0; j < 16; j++) {  c[j] = a[j] \* b[j];  }  }  t = omp\_get\_wtime() - t;  // Вывод времени выполнения для char без SSE  cout << "Для типа char без SSE: " << t << " сек." << endl;  for (int i = 0; i < 16; i++) {  cout << "c[" << i << "] = " << static\_cast<int>(c[i]) << "\n";  }  cout << endl;  // Выполнение операций с SSE для типа char  t = omp\_get\_wtime();  for (int i = 0; i < 1000000; i++) {  // Загрузка данных в регистры SSE и умножение  \_\_m128i xmm\_a = \_mm\_load\_si128((\_\_m128i\*)a);  \_\_m128i xmm\_b = \_mm\_load\_si128((\_\_m128i\*)b);  \_\_m128i xmm\_c = \_mm\_mullo\_epi16(xmm\_a, xmm\_b);  // Сохранение результата в массив c  \_mm\_store\_si128((\_\_m128i\*)c, xmm\_c);  }  t = omp\_get\_wtime() - t;  // Вывод времени выполнения для char с SSE  cout << "Для типа char с SSE: " << t << " сек." << endl;  for (int i = 0; i < 16; i++) {  cout << "c[" << i << "] = " << static\_cast<int>(c[i]) << "\n";  }  return 0;  } |

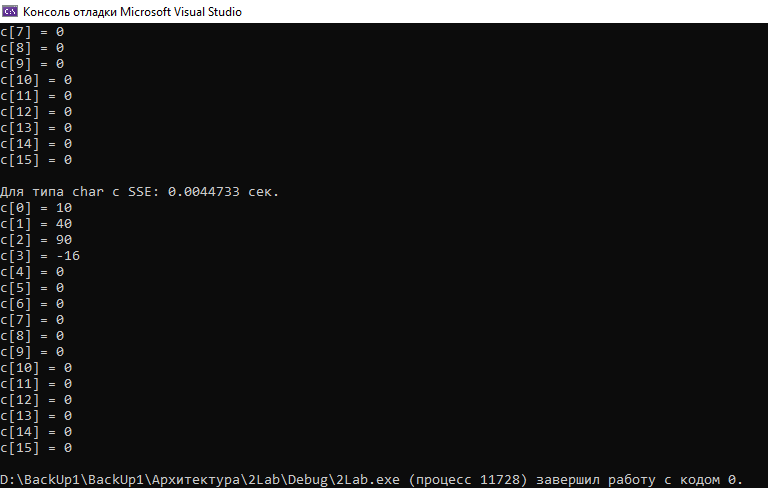
**

Рис. 2. Результат выполнения

Изменение значения c[3] между результатами с использованием SSE и без SSE обусловлено различиями в обработке переполнений при умножении. При использовании инструкции SSE \_mm\_malloc\_epi16 происходит насыщенное умножение, что означает, что результат умножения, превышающий максимальное значение для типа данных, будет усечен до максимального значения. В данном случае, при умножении a[3] на b[3] происходит переполнение, и значение усекается до максимально допустимого значения для типа char, равного 127.

В отсутствие использования SSE происходит обычное умножение для типа char, и переполнение приводит к циклическому переполнению буфера. В результате значения c[3] становится отрицательным (-96).

Таким образом, различие в значениях c[3] обусловлено разным подходом к обработке переполнений при умножении, что характерно при использовании насыщенного умножения SSE