ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики, механики и компьютерных наук им. И.И. Воровича

Кафедра алгебры и дискретной математики

**ОТЧЕТ**

на тему:

### **Блочные вычисления. Модели времени выполнения программ. Блочные размещения массивов, дополняющие блочные вычисления**

**Выполнила:**

студент 4 курса 2 группы

Руднев Данил Олегович

Ростов-на-Дону

2018

**Содержание**

[Постановка задачи 3](#_Toc487789905)

[Алгоритм решения 3](#_Toc487789908)

[Результаты работы программы 4](#_Toc487789912)

[Характеристики компьютера 5](#_Toc487789916)

[Выводы 5](#_Toc487789931)

[Приложение 1 7](#_Toc487789938)

**Постановка задачи**

Задание 42.

Написать программу блочного умножения двух матриц C = A\*B.

Матрица A верхне-треугольная. Хранится в виде одномерного массива по блочным строкам.

Матрица B симметричная, хранится как нижне-треугольная. Хранится в виде одномерного массива по блочным столбцам.

Распараллелить блочную программу умножения двух матриц C = A\*B с использованием технологии OpenMP двумя способами

* Перемножение каждых двух блоков выполнить параллельно
* В разных вычислительных ядрах одновременно перемножать разные пары блоков.

Определить оптимальные размеры блоков в обоих случаях. Провести численные эксперименты и построить таблицу сравнений времени выполнения различных программных реализаций решения задачи. Определить лучшие реализации.

Проверить корректность (правильность) программ.

**Алгоритм решения**

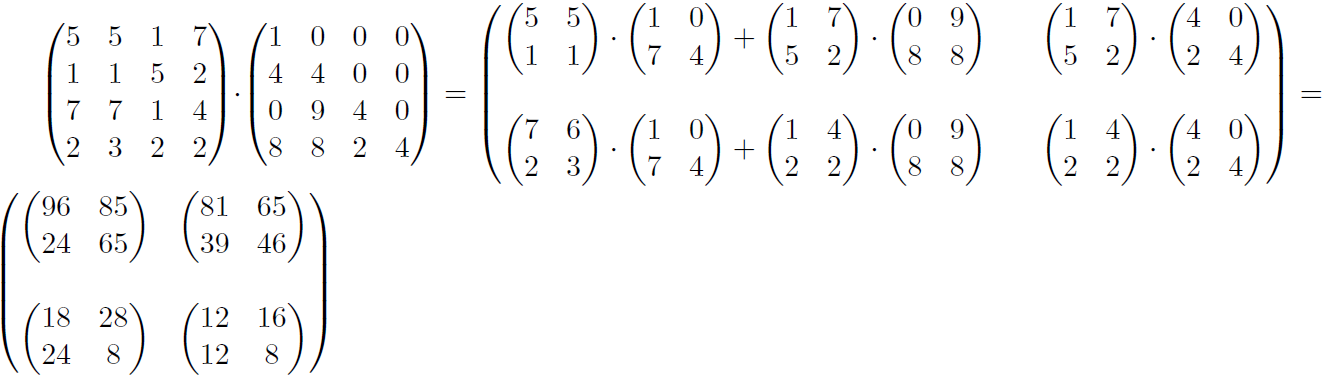


Рис. 1 Демонстрация блочного умножения матрицы 4x4 с разбиением на блоки размером 2x2

Алгоритм умножения будет выглядеть следующим образом:

1. Организуем стандартный алгоритм умножения для блоков:

for (i = 0; i<S; ++i)

for (j = 0; j<S; ++j)

for (k = j; k<S; ++k)

S=N/k,где N – количество строк/столбцов исходной матрицы, sz\_b – количество строк/столбцов в блоке.

1. Найдем индекс начала необходимого блока, записанного в векторе.

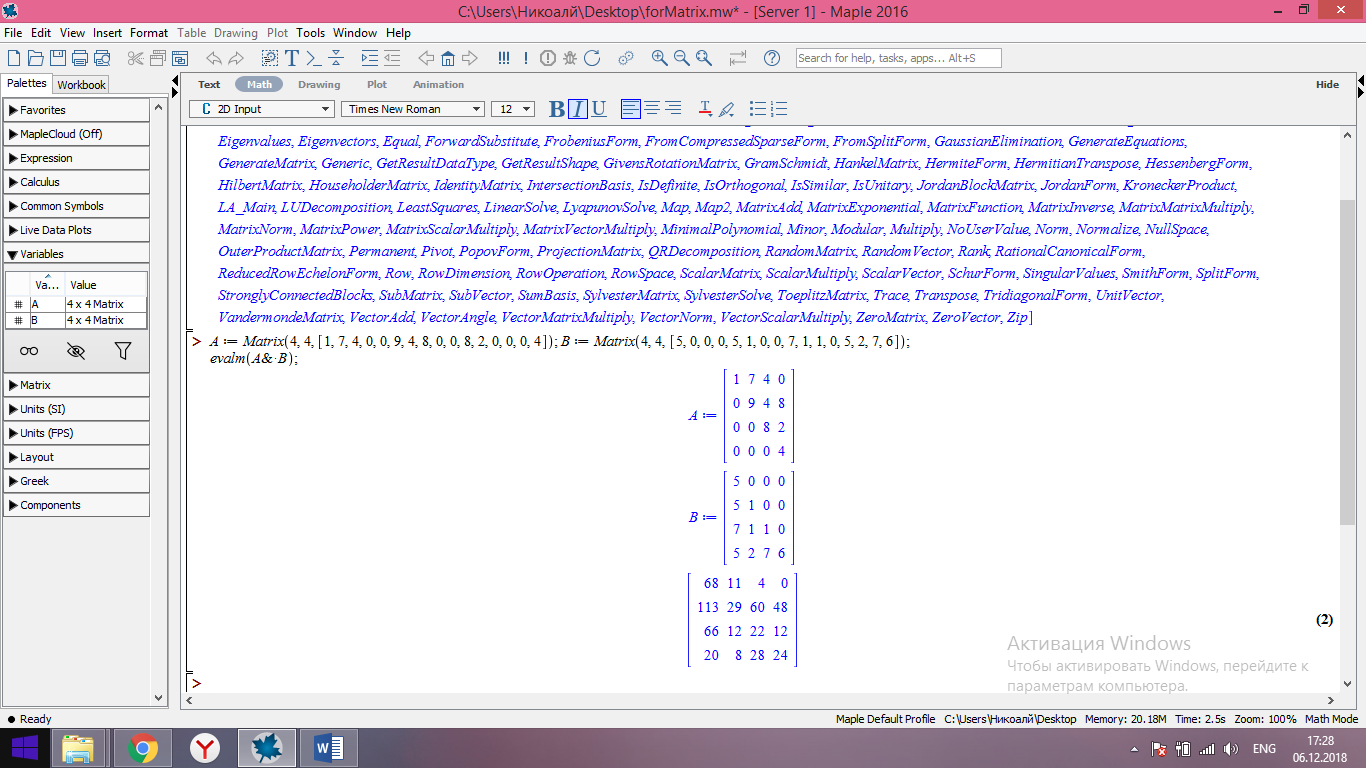
Так, для вектора, полученного из верхне-треугольной матрицы A путем построчной записи блоков, получено выражение: int \*a = A + (i \* (S + 1) - (i + 1)\*i / 2 + (k - i)) \* sz\_b\*sz\_b. Для матрицы B: int \*b = B + (j \* (S + 1) - (j + 1)\*j / 2 + (k - j)) \* sz\_b\*sz\_b.

1. Организуем стандартный алгоритм умножения для элементов внутри блока.

Внутри всех блоков расположение элементов построчное, поэтому для перехода на следующую строку требуется умножение индекса строки на количество элементов. Изменение индексов элементов будет происходить одинаково во всех блоках независимо от представления матрицы в векторе.

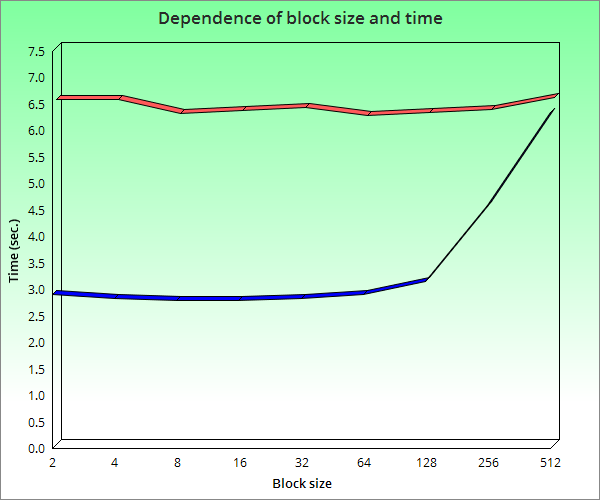
В условии не уточняется, каким образом мы будем хранить результирующую матрицу C, поэтому будем считать, что она должна быть записана в виде вектора, блоки которой расположены построчно. Тогда вычисление размера блока будет выглядеть следующим образом: int \*c = C + i \* N\*sz\_b + j \* sz\_b\*sz\_b.

**Проверка правильности работы программы**



**Результаты работы программы**

При блочном умножении матриц размером 512x512 получили результаты, которые можно представить в виде графика:



Red line is Blocked non parallel  
Blue line is parallel (16 Threads)

Non blocked multiplying takes 14.7697sec.

**Характеристики компьютера**

Процессор Intel Celeron i5-3230M, тактовая частота до 2,60 GHz.

Количество ядер: 2

Количество потоков: 2

Объем оперативной памяти: 8GB

**Выводы**

Неоптимальные варианты - разбиения матрицы на блоки размером 1x1 и 512x512. Лучший размер блока от 2 до 128. Такой результат обосновывается особенностью работы с кэш-памятью: при наилучшем разбиении, количество кэш-промахов уменьшается. Как следствие, получаем более быструю работу программы.

**Дополнительная статистика приведена в файле experience.txt**