**Курсов проект**

**по**

**Разпределени Софтуерни Архитектури**

Тема 10: Умножение на матрици

Изготвили: Проверил:

Ева Паунова, 61940 .................................

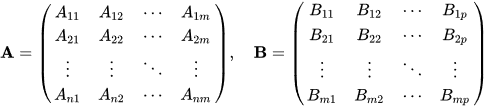
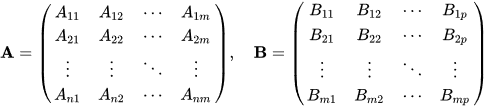
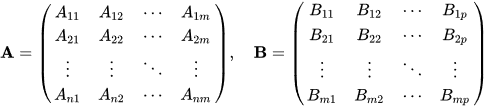
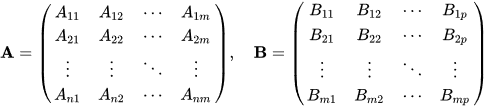
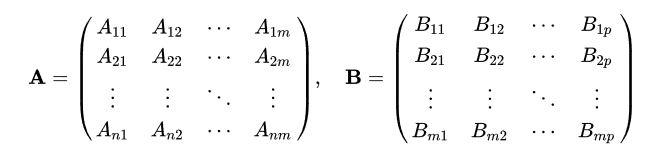
Лиляна Видева, 61887 /ас. Христо Христов/

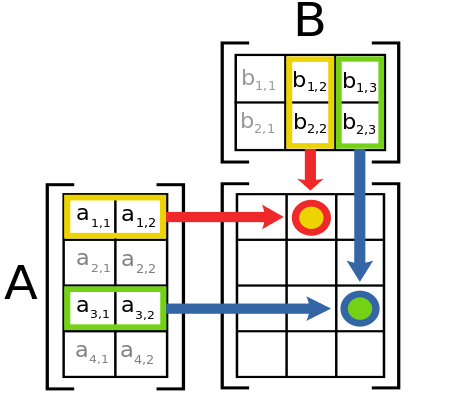
# Условие на задачата

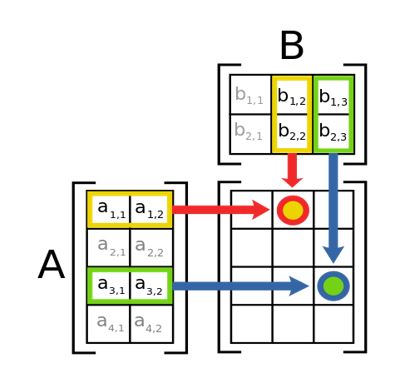
Разглеждаме матриците ***A*** с размерност (***m***, ***n***) и ***B*** с размерност (***n***, ***k***). Матрицата ***C*** = ***A***.***B***, равна на произведението на ***A*** и ***B*** ще има размерност (***m***, ***k***). Да се напише програма която пресмята матрицата ***C***. Работата на програмата по умножението на матриците да се раздели по подходящ начин на две или повече нишки (задачи).

* Изискванията към програмата
  + Задаване на размерността на матриците от подходящо избрани командни параметри – например „-***m*** 1024 -***n*** 512 -***k*** 2048“;
  + Произволно генериране на елементите на матриците с помощта на ***Math.random()*** или ***java.util.concurrent.ThreadLocalRandom***;
  + Команден параметър, указващ входен текстов файл, съдържащ матриците, които ще се умножават – например „-***i*** m3x-data.in“;
  + Команден параметър, указващ изходен файл, съдържащ резултата от пресмятането – например „-***о*** m3x-data.out“;
  + Команден параметър, указващ максималния брой нишки (задачи), на които разделяме работата по пресмятането на елементите на ***C*** – например “–***t*** 1” или “-***tasks*** 3”;
  + Извеждане на подходящи съобщения на различните етапи от работата на програмата, както и времето, отделено за изчисление;
  + Задаване на „***quiet***“ режим на работа на програмата, при който не се стартира графичния и интерфейс и се извежда само времето отделено за изчисление на резултатната матрица, чрез подходящо избран команден параметър – например “-***q***”.

# Алгоритъм за умножение на матрици

Ако ***A*** е ***n*** × ***m*** матрица и ***B*** е ***m*** × ***p*** матрица, A = ( A 11 A 12 ⋯ A 1 m A 21 A 22 ⋯ A 2 m ⋮ ⋮ ⋱ ⋮ A n 1 A n 2 ⋯ A n m ) , B = ( B 11 B 12 ⋯ B 1 p B 21 B 22 ⋯ B 2 p ⋮ ⋮ ⋱ ⋮ B m 1 B m 2 ⋯ B m p ) {\displaystyle \mathbf {A} ={\begin{pmatrix}A\_{11}&A\_{12}&\cdots &A\_{1m}\\A\_{21}&A\_{22}&\cdots &A\_{2m}\\\vdots &\vdots &\ddots &\vdots \\A\_{n1}&A\_{n2}&\cdots &A\_{nm}\\\end{pmatrix}},\quad \mathbf {B} ={\begin{pmatrix}B\_{11}&B\_{12}&\cdots &B\_{1p}\\B\_{21}&B\_{22}&\cdots &B\_{2p}\\\vdots &\vdots &\ddots &\vdots \\B\_{m1}&B\_{m2}&\cdots &B\_{mp}\\\end{pmatrix}}} матричното произведение ***AB*** представлява матрица ***C n*** × ***p***A B = ( ( A B ) 11 ( A B ) 12 ⋯ ( A B ) 1 p ( A B ) 21 ( A B ) 22 ⋯ ( A B ) 2 p ⋮ ⋮ ⋱ ⋮ ( A B ) n 1 ( A B ) n 2 ⋯ ( A B ) n p ) {\displaystyle \mathbf {A} \mathbf {B} ={\begin{pmatrix}\left(\mathbf {AB} \right)\_{11}&\left(\mathbf {AB} \right)\_{12}&\cdots &\left(\mathbf {AB} \right)\_{1p}\\\left(\mathbf {AB} \right)\_{21}&\left(\mathbf {AB} \right)\_{22}&\cdots &\left(\mathbf {AB} \right)\_{2p}\\\vdots &\vdots &\ddots &\vdots \\\left(\mathbf {AB} \right)\_{n1}&\left(\mathbf {AB} \right)\_{n2}&\cdots &\left(\mathbf {AB} \right)\_{np}\\\end{pmatrix}}} , където всяка ***i****,* ***j*** клетка се получава при умножаването на клетките ***aik*** (хоризонтален ред ***i*** на ***A***) по клетките ***bkj*** (вертикална колона ***j*** на ***B***), за ***k*** = 1, 2, ..., ***m*** и сумирането на резултатите в ***k***:

( A B ) i j = ∑ k = 1 m A i k B k j . {\displaystyle (\mathbf {A} \mathbf {B} )\_{ij}=\sum \_{k=1}^{m}A\_{ik}B\_{kj}\,.}

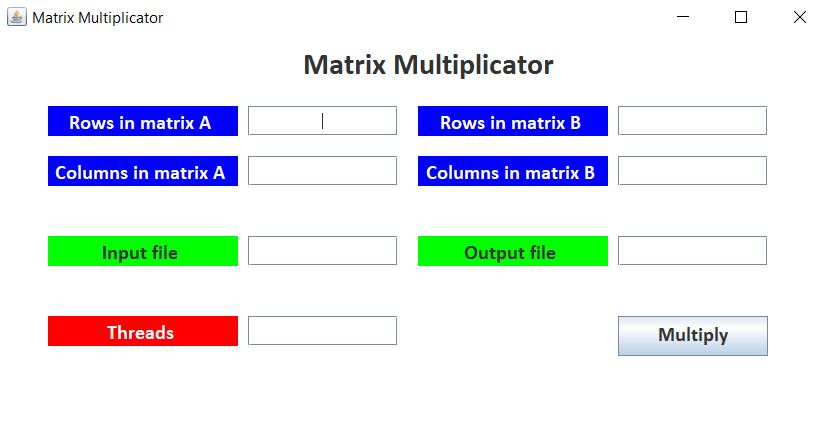
Нагледно умножението на матрици се представя по следния начин:

# Описание на реализацията

Програмата е разработена на езика ***Java*** и използва разделяне на изчисленията за по-бързо пресмятане на стойностите в новата матрица. Разработена е в 9 класа – ***ArgumentParser, Matrix, MatrixBuilder, MatrixesMultiplicator, MatrixMultiplicatorGui, MultiplicatorStarter, MultithreadedMatrixesMultiplier, MultithreadedMultiplyTask*** и ***Test***, като всеки от тях отговаря съответно за:

* ***ArgumentParser*** – клас, отговорен за обработката на подадените от потребителя аргументи в командния ред
* ***Matrix*** – клас, съхраняващ матрица, както и нейните размерности
* ***MatrixBuilder*** – клас, отговорен за изграждането на матриците от файл или от произволно генерирани числа
* ***MatrixesMultiplicator*** – клас, отговорен за разпределянето на извършваното умножение в зависимост от броя на нишки, както и за записването на резултатната матрица във файл
* ***MatrixMultiplicatorGui*** – клас, отговорен за изграждането на графичния интерфейс с помощта на класове от пакета ***javax.swing*** и обработката на потребителския вход
* ***MultiplicatorStarter*** – клас, отговорен за стартирането на програмата чрез графичен интерфейс или през конзолата, в зависимост от това дали е подаден параметър „***-desktop***“
* ***MultithreadedMatrixesMultiplier*** – клас, отговорен за контрола над нишките при извършването на умножението на матрици с помощта на класовете ***Executor*** и ***ExecutorService***
* ***MultithreadedMultiplyTask*** – клас, отговорен за инициализиране на нишка, която ще извършва изчисления
* ***Test*** – клас, отговорен за проверка дали графичният потребителски интерфейс се визуализира и потребителят може да работи с програмата

Визуализация на графичния потребителски интерфейс

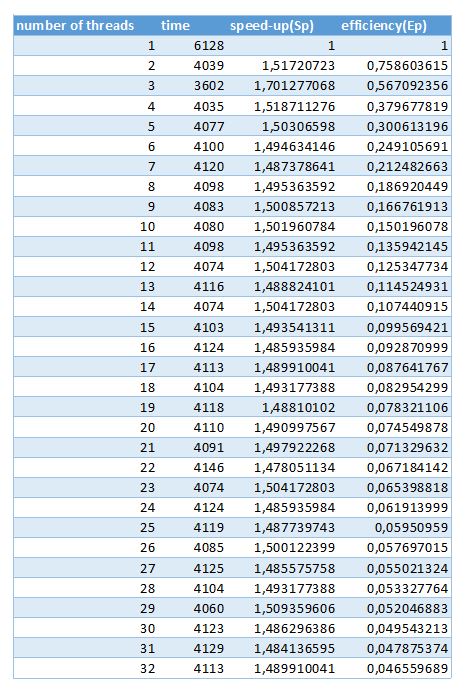


# Тестови замервания

Направени са тестови замервания за умножение на две матрици с размерности 1000х1000, като изчислението е пускано на различен брой нишки.

Първата таблица е от тестването върху лаптоп HP:

Model name: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90GHz



Резултатите във втората таблица са получени на машината t5600.rmi.yaht.net :

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

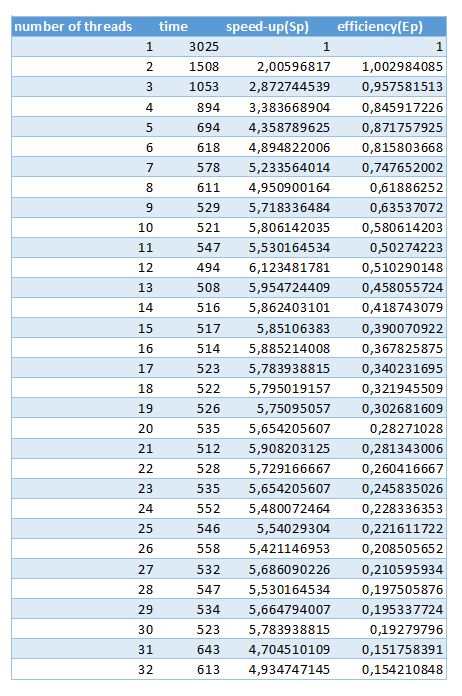
CPU(s): 32

Thread(s) per core: 2

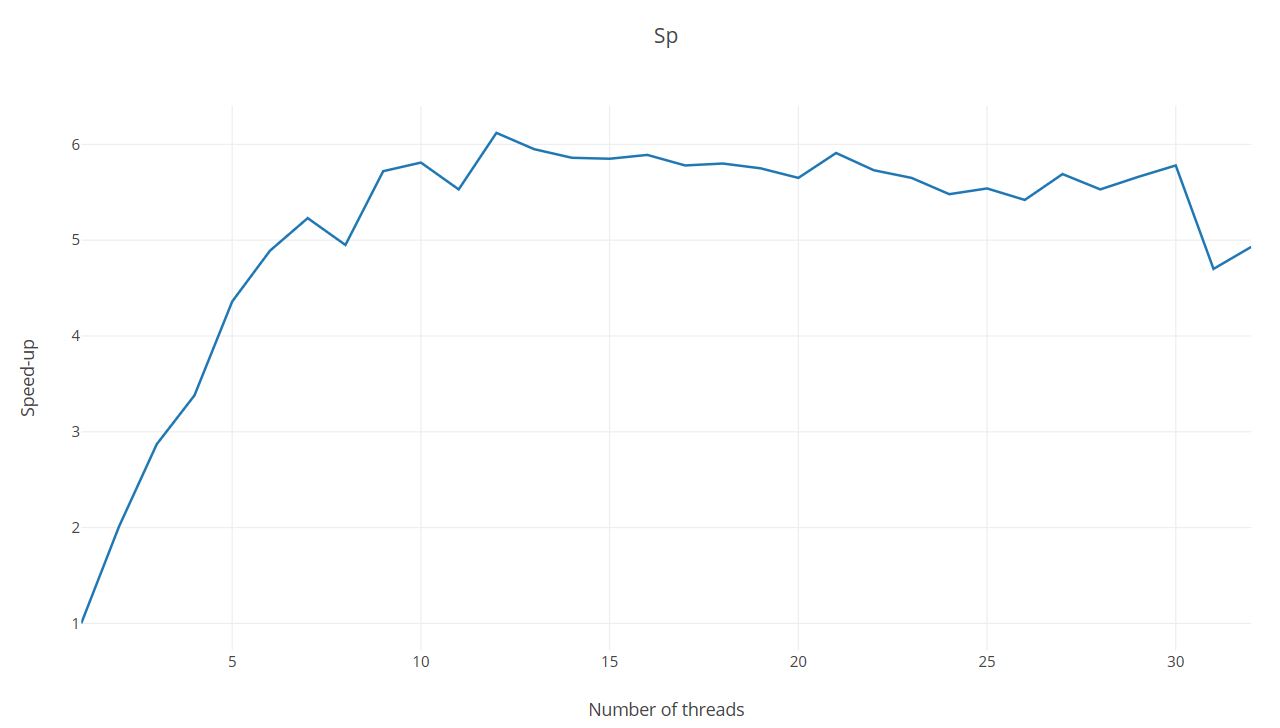
Model: 45

Model name: Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz

CPU MHz: 2199.865



* На диаграмата е показано постигнатото ускорение при умножение на две матрици спрямо броя на използваните нишки (***Sp***=***T1***/***Tp***):



* На диаграмата е показана постигнатата ефективност при умножение на две матрици спрямо броя на използваните нишки (***Ep***=***Sp***/***p***):

