НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ

СІКОРСЬКОГО»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра обчислювальної техніки\_\_\_\_\_\_\_\_\_\_\_\_

**ДОМАШНЯ КОНТРОЛЬНА РОБОТА**

з дисципліни «Паралельні та розподілені обчислення»

(«Паралельне програмування»)

на тему: «Розробка програмного забезпечення для паралельних комп’ютерних систем»

Студентки 3 курсу групи ІП-74

спеціальності 121

«Інженерія програмного забезпечення»

Герасименко Ю.К.

Керівник доцент Корочкін О.В.

Київ - 2020 рік

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

## ***З А В Д А Н Н Я***

### **НА ДКР СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Герасименко Юлії Костянтинівни* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема роботи «Розробка програмного забезпечення для паралельних

комп’ютерних систем»

керівник роботи Корочкін Олександр Володимирович к.т.н.**,** доцент

2. Строк подання студентом роботи 18 травня 2020 р.

3. Вхідні дані до роботи

- математична задача

- структури ПКС ОП та ПКС ЛП

- мова (бібліотека) для ПРГ1

- мова (бібліотека) для ПРГ2

4. Зміст розрахунково-пояснювальної записки

- розробка і тестування програми ПРГ1 для ПКС ОП

- розробка і тестування програми ПРГ2 для ПКС ЛП

5. Перелік графічного матеріалу

- структурна схема ПКС СП

- структурна схема ПКС ЛП

- схеми алгоритмів процесів і головної програми для ПРГ1

- схеми алгоритмів процесів і головної програми для ПРГ2.

7. Дата видачі завдання \_\_\_\_\_\_\_\_\_1 04 2020\_\_\_\_\_\_\_\_\_\_\_.

ЗМІСТ

ВСТУП…………………………………………………………………………… 6

РОЗДІЛ 1. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП……………………8

1.1 Розробка паралельного математичного алгоритму……………………8

1.2 Розробка алгоритмів процесів………………………………………….9

1.3 Розробка схеми взаємодії процесів…………………………………….11

1.4 Розробка програми ПРГ1……………………………………………….13

1.5 Тестування програми ПРГ1 …………………………………………….13

1.6 Висновки до розділу 1 …………………………………………………..16

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП ……………………17

2.1 Розробка паралельного математичного алгоритму………………….17

2.2 Розробка алгоритмів процесів…………………………………………18

2.3 Розробка схеми взаємодії процесів…………………………………..23

2.4 Розробка програми ПРГ2……………………………………………25

2.5 Тестування програми ПРГ2……………………………………………25

2.6 Висновки до розділу 2…………………………………………………28

ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ …………….………………………28

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………… 35

ДОДАТКИ……………………………………………………............................. 35

**ВСТУП**

Багато чого на світі можна та навіть нужно розпаралелити. Ви спитаєте:”Чому?”

Я відповім:”По-перше, не одній людині не під силу зробити велику частину роботи не поділівши на окремі функціональні частини, тим-паче, що машина, авжеж, краща за людину у цьому смислі, але один процесор не всесильний та сверхшвидкий, навіть суперкомп’ютер ”. Цікаво, що навіть перевірка роботи викладачем потребує саме паралелізації роботи на кожного студента та перевірка кожного потоку окремо(в якості потоку виступають студенти).

У розділі 1 я познайомлюся із засобами для паралельної розробки програми в мовах C# та в розділі 2 в мові Java. Дізнаюся більше про синхронізацію потоків у C# засобами Win32:

Т2.Join() - очівання у інших потоках на завершення виконання потоку Т2();

Semaphore;

atomic operations: клас Interlocked для числової змінної Сmin

Класи EventWaitHandle, що є подією для синхронізації потоків(наприклад для введення та очікування на ввід(вивід)змінних

В Win32 не має бар’єрів для синхронізації. Трохи інша форма від бар’єрів часто називають join. Він дозволяє потоки очікувати на завершення інших потоків. Це досягається за допомогою очікування на обробники потоку за допомогою WaitForMultipleObjects() API в Win32.

Win32 не має моніторів

У Java:

бібліотечка MPI дозволяє відправляти(MPI.COMM\_WORLD.Send) та приймати(MPI.COMM\_WORLD.Recv)

використовувати барʼєр MPI.COMM\_WORLD.Barrier();

MPI.Finalize();-закінчити виконання потоків

MPI.Init(); -почати виконання потоків

Також я зрозуміла, що можу полегшити написання за допомогою, встроєних засобів таких як:

MPI.Bcast - маючи в одному процесі дані розіслати в усі інші

MPI.Reduce-зібрати дані з усіх процесів та знайти (мін,макс, суму)

MPI.Gather-зібрати в одномі процесі всі дані з процесів

MPI.Scatter- маючи в одному процесі дані розіслати в усі інші, попередньо розбивши на кількість процесів

Але, на жаль, я спочатку намалювала схему взаємодії процесів, а вже потім не захотіла перероблювати, наприклад з використанням MPI.Bcast, хоча це найлегший спосіб розпаралелення, бо ти повинен писати лише 1 строчку коду замість 2\*(N-1) = recv+send. Також хочу зауважити, що MPI.Reduce дозволяє навіть обчислити мін, макс, суму з усіх процесів, що може використовуватись для вирішення задачі ПРГ2. MPI.COMM\_WORLD.Reduce(v, 0, answer, 0, 1, MPI.DOUBLE, MPI.MAX, 0);

Я познайомлюсь з MPJ, так би мовити, MPI у світі Java. Відмінністю від реалізації MPI у інших мовах буде те, що замість нижнього підкреслення ставитися будуть точки та перед заданням таких функцій як MPI.Recv потрібно буде ставити СOMM\_WORLD.

Проведено тестування отриманих програмних продуктів і зроблено висновки по їх ефективності.

Лістинги та алгоритми розроблених програм наведено у додатках.

**РОЗДІЛ 1. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС CП**

**1.1. Розробка паралельного математичного алгоритму.**

Завдання: A = min(C) \* Z+D (MX\* MR). Нехай N — розмірність матриці, P — кількість потоків (8 за умовою задачі). Тоді H = N / P.

Z\_H — H елементів вектора Z, MR\_H — H рядків матриці MR, MXR\_H — H рядків добутку матриць MR\_H i MX. Cmin має найменше значення на проміжку розміром H. CminZ\_H дорівнює добутку мін С та Z\_H. В minC знаходиться мінімум вектора С. Кожну операцію у формулі ми будемо виконувати паралельно:

Паралельний алгоритм множення вектора MR\_H на матрицю MX дає у

результаті вектор MXR\_H:

MXR\_H = MX \* MR\_H;

Паралельний алгоритм множення вектора D на вектор MXR\_H дає у

результаті вектор MXRD\_H:

MXRD\_H = D \* MXR\_H;

Паралельний алгоритм oбчислення мінімального елемента minC:

minC = MIN({MIN(C[0..H] ... C[7\*H..8\*H]) }).

Паралельний алгоритм знаходження векторa CminZ\_H множення векторa Z на число minC :

CminZ\_H = minC \* Z\_H;

Паралельний алгоритм знаходження вектора А

A[i] = CminZ\_H+MXRD\_H ; i = (0..P-1);

Повний математичний алгоритм:

1. Cmin[i] = min(C\_H), i = (0..P-1);

2. minC = min(minC, Cmin[i]), i = (0..P-1);

3. MXR\_H = MX \* MR\_H;

4.Z\_MXRD\_H = D \* MXR\_H;

5. CminZ\_H = minC \* Z\_H;i = (0..P-1);

6. A[i] = CminZ\_H+MXRD\_H ; i = (0..P-1);

**1.2. Розробка алгоритмів процесів.**

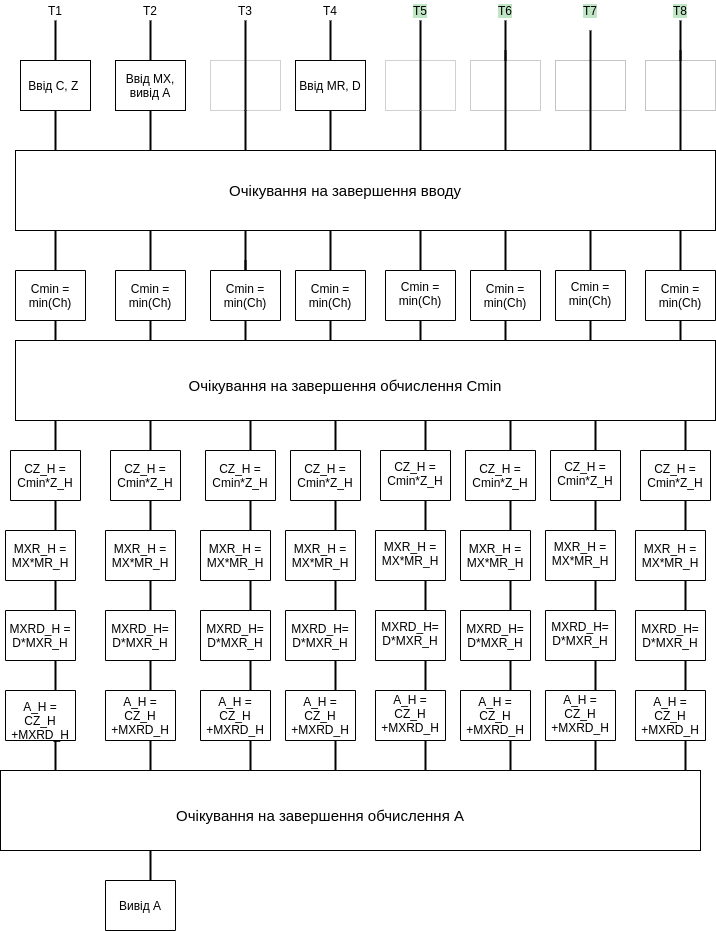
Розроблюване програмне забезпечення повинно працювати на системі

із кількістю процесорів P = 8.

Задачі Т(1) – Т(8)

|  |  |  |
| --- | --- | --- |
|  | Якщо Func1 Ввести C, Z. |  |
|  | Якщо Func2 Ввести MX. |  |
|  | Якщо Func4 Ввести D,MR. |  |
|  | System.Threading.EventWaitHandle | Подія синхронізації потоків(System.Threading.EventWaitHandle) |
|  | Обчислення мінімального елемента |  |
|  | Cmin[i] = min(C\_H), i = (0..P-1); |  |
|  | minC = MIN({MIN(C[0..H] ... C[7\*H..8\*H]) }). | atomic operations |
|  | Чекати закінчення присвоєння у всіх потоках. | Semaphores |
|  | Обчислення MXR\_H = MX \* MR\_H; |  |
|  | Чекати закінчення обчислення у всіх потоках. | Semaphore |
|  | Обчислення Z\_MXRD\_H = D \* MXR\_H; |  |
|  | Чекати закінчення обчислення у всіх потоках. | Semaphore |
|  | Обчислення CminZ\_H = minC \* Z\_H; |  |
|  | Чекати закінчення обчислення у всіх потоках. | Semaphore |
|  | Обчислення A[i] = CminZ\_H+MXRD\_H ; |  |
|  | Очікування на завершення потоку T2 | Join |
|  | Вивести результат A. |  |

**1.3. Розробка схеми взаємодії процесів.**

****

Т — потік, Т(1) — перший потік, Т(2) — другий потік, Т(3) — третій потік...Т(8) — останній потік

**1.4. Розробка програми ПРГ1.**

Програма була написана на мові C# з використанням бібліотеки System.Threading, Microsoft.Win32.

Складається з одного модуля: «Program.cs».

Модуль “Program.cs” складається з класу Program, який в

свою чергу скаладається з:

методa Main — вхід в програму, містить в собі розмірність

метода Func1..8 — метод для обчислень алгоритму, який запускається у

різних задачах, які виконують обчислення

Лістинг програми ПРГ1 наведено у додатку Б.

**1.5. Тестування програми ПРГ1.**

Таблиця 1.1 Час виконання програми для ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 8 | 6 | 4 | 3 | 2 | 1 |
| 898 | 6036 | 2599 | 3154 | 3577 | 6752 | 5754 |
| 1800 | 13516 | 10276 | 24267 | 39384 | 38535 | 40011 |
| 2400 | 47165 | 105081 | 38624 | 62230 | 82928 | 103333 |

На основі даних із таблиці виконано розрахунок значень коефіцієнтів прискорення (КПI = T1 / TI) та ефективності (КЕІ = T1 / (TI \* I)\*100%), де І — кількість процесорів:

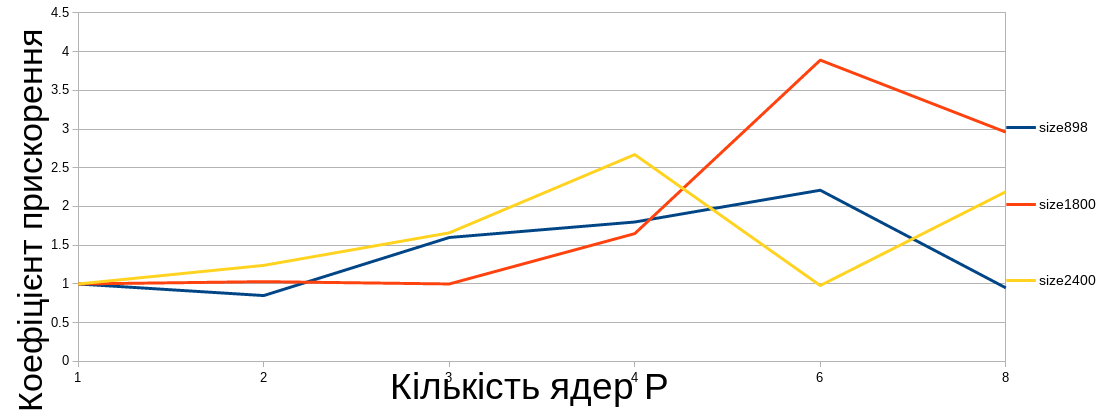
Таблиця 1.2 Значення Кп для ПРГ1

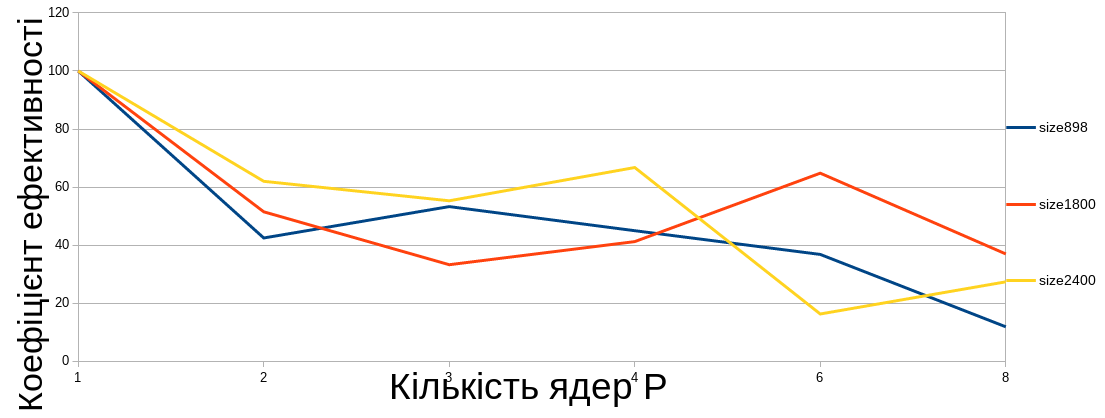
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 8 | 6 | 4 | 3 | 2 | 1 |
| 898 | 0.95 | 2.21 | 1.8 | 1.6 | 0.85 | 1 |
| 1800 | 2.96 | 3.89 | 1.65 | 1 | 1.03 | 1 |
| 2400 | 2.19 | 0.98 | 2.67 | 1.66 | 1.24 | 1 |

Таблиця 1.3 Значення Ке для ПРГ1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 8 | 6 | 4 | 3 | 2 | 1 |
| 900 | 11.9 | 36.83 | 45 | 53.3 | 42.5 | 100 |
| 1800 | 37 | 64.8 | 41.25 | 33.3 | 51.5 | 100 |
| 2400 | 27.4 | 16.33 | 66.75 | 55.3 | 62 | 100 |

Було побудовано графіки коефіцієнтів прискорення та ефективності:





**1.6. Висновки до розділу 1**

Виконано розробку програми ПРГ1 для ПКС1 зі спільною пам’яттю з

використанням мови С#. Тестування програми показало наступне:

значення коефіцієнту прискороння лежать в межах від 0,85 до 2, 24, для 2

потоків, 1-1,66 — для 3 потоків, 1,65 - 2,67 для 4 потоків;

0,98-3,89 для 6 потоків; 0,95-2,96 для 8 потоків;

значення коефіцієнту ефективності лежать в межах від 45,5 % до 62

% для 2 потоків, 33,3 % -55,3 % — для 3 потоків, 41,25 % - 66,75%

для 4 потоків, 16,33-64,8 для 6 потоків;11,9-37 для 8 потоків;

Деякі прискорення схожі на лінійні, найбільших різниці точок на графіку викликало використання замість 5 та 7 процесорів, процесорів 6 та 8, що на 2 більше ніж у кількість попередньо використанного процесору на графіку. Отже, можна п можна зпророкувати, що якби ми тестували на процессорах, різниця сусідніх відрізняється лише на 1, то графік набув більш лінійної форми. Коефіцієнт ефективності повільно падає зі збільшенням кількості потоків через накладні розходи, що витрачаються для їх створення. Також дещо уповільнюють програму множинні синхронізації.

**РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ЛП.**

**2.1. Розробка паралельного математичного алгоритму.**

Завдання: A = min(C) \* Z+D (MX\* MR). Нехай N — розмірність матриці, P — кількість потоків (8 за умовою задачі). Тоді H = N / P.

Z\_H — H елементів вектора Z, MR\_H — H рядків матриці MR, MXR\_H — H рядків добутку матриць MR\_H i MX. Cmin має найменше значення на проміжку розміром H. CminZ\_H дорівнює добутку мін С та Z\_H. В minC знаходиться мінімум вектора С. Кожну операцію у формулі ми будемо виконувати паралельно:

Паралельний алгоритм множення вектора MR\_H на матрицю MX дає у

результаті вектор MXR\_H:

MXR\_H = MX \* MR\_H;

Паралельний алгоритм множення вектора D на вектор MXR\_H дає у

результаті вектор MXRD\_H:

MXRD\_H = D \* MXR\_H;

Паралельний алгоритм oбчислення мінімального елемента minC:

minC = MIN({MIN(C[0..H] ... C[7\*H..8\*H]) }).

Паралельний алгоритм знаходження векторa CminZ\_H множення векторa Z на число minC :

CminZ\_H = minC \* Z\_H;

Паралельний алгоритм знаходження вектора А

A[i] = CminZ\_H+MXRD\_H ; i = (0..P-1);

Повний математичний алгоритм:

1. Cmin[i] = min(C\_H), i = (0..P-1);

2. minC = min(minC, Cmin[i]), i = (0..P-1);

3. MXR\_H = MX \* MR\_H;

4.Z\_MXRD\_H = D \* MXR\_H;

5. CminZ\_H = minC \* Z\_H;i = (0..P-1);

6. A[i] = CminZ\_H+MXRD\_H ; i = (0..P-1);

**2.2. Розробка алгоритмів процесів.**

**№ Алгоритм задачи Т1**

Ввести Z, МR

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Прийняти С2H від задачі Т7

Передати С2H в задачу Т5

Знайти мінімум у С2H та присвоїти в Cmin[2H]

Передать Cmin[2H] задачі Т5

Передать MR[2H], Z[2H] задачі Т3

Передать MR[3H], Z[3H] задачі Т6

Передать MR[2H], Z[2H] задачі Т4

Приняти D, МX, Cmin від задачі Т2

Обчислити A2Н = Сmin \* ZH + D \* (MX \* MRH)

Передать А2Н задачі Т7

**№ Алгоритм задачи Т2**

Приняти С4H від задачі Т6

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Знайти мінімум у С4H та присвоїти в Cmin[5H]

Передать Cmin[5H] задачі Т6

Приняти Cmin від задачі Т6

Приняти ZH, МRH від задачі Т1

Обчислити A4Н = Сmin \* ZH + D \* (MX \* MRH)

Передать А4H задачі Т6

**№ Алгоритм задачи Т3**

Приняти С3H, від задачі Т2

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Знайти мінімум у С3H та присвоїти в Cmin[4H]

Передать Cmin[4H] задачі Т2

Приняти D, MX, Cmin від задачі Т2

Приняти ZH, МRH від задачі Т4

Обчислити A3Н = Сmin \* ZH + D \* (MX \* MRH)

Передать А3H задачі Т2

**№ Алгоритм задачи Т4**

Приняти С2H від задачі Т3

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Знайти мінімум у С2H  та присвоїти в Cmin[H]

Передать Cmin[H] задачі Т1

Приняти D, МХ, Cminвід задачі Т3

Приняти Z2H, МR2H від задачі Т1

Обчислити AH = Сmin \* ZH + D \* (MX \* MRH)

Приняти А2H від задачі Т5

**№ Алгоритм задачи Т5**

Приняти СH від задачі Т1

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Знайти мінімум у СH та присвоїти в Cmin[3H]

Передать Cmin[3H] задачі Т3

Приняти МR2H, Z2H від задачі Т1

Приняти D, MX, Cmin від задачі Т3

Обчислити AН = Сmin \* ZH + D \* (MX \* MRH)

Передать АН задаче Т1

**№ Алгоритм задачи Т6**

Ввести С

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Відправити С[3H] задачі Т7

Відправити С[4H] задачі Т2

Приняти Cmin[2H] від задачі Т7

Приняти Cmin[5H] від задачі Т2

Знайти min = Сmin

Приняти МRH, ZH від задачі Т5

Приняти D, MX від задачі Т2

Обчислити AH = Сmin \* ZH + D \* (MX \* MRH)

Закінчити обчислення у всіх потоках(застосувати Бар’єр)

Вивести А

**№ Алгоритм задачи Т7**

Ввести D, MX

Приняти C3H від задачі Т6

Передати С2H в задачу Т1

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Знайти мінімум у С3H та присвоїти в Cminх2Н

Передать Cmin[2H] задачі Т6

Приняти D, MX, Cmin, МR3H, Z3H від задачі Т1

Обчислити A3H = Сmin \* ZH + D \* (MX \* MRH)

Передать АH задаче Т6

**№ Алгоритм задачи Т8**

Приняти CH від задачі Т4

Чекати закінчення вводу у всіх потоках. (застосувати Бар’єр)

Знайти мінімум у СH та присвоїти в Cmin[1H]

Передать Cmin[1H] задачі Т7

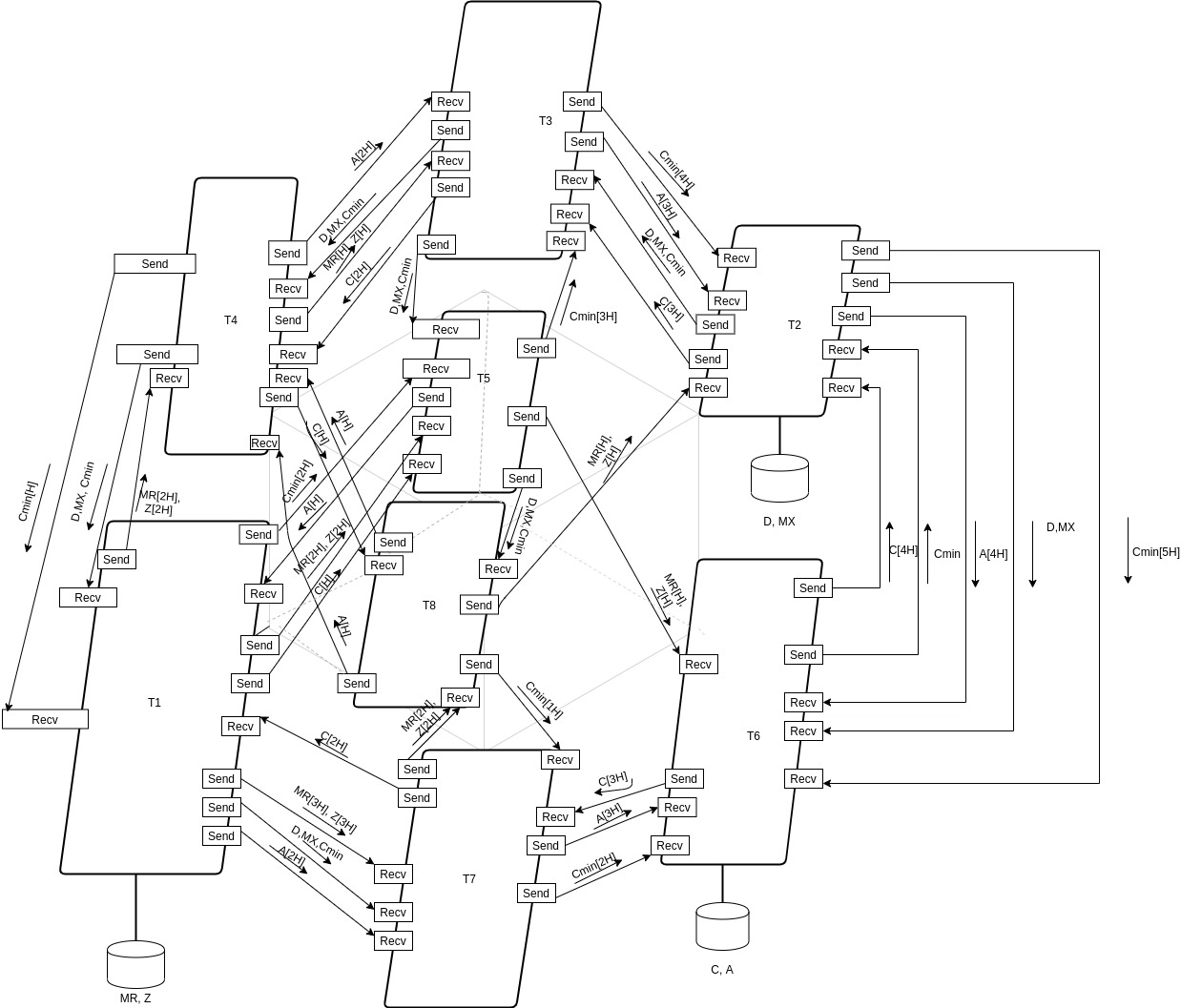
Приняти D, MX, Cmin від задачі Т5

Приняти МR2H, Z2H від задачі Т7

Обчислити AH = Сmin \* ZH + D \* (MX \* MRH)

Передать АH задаче Т4

**2.3. Розробка схеми взаємодії процесів**

****

Т — потік, Т(1) — перший потік, Т(2) — другий потік, Т(3) — третій потік...Т(8) — останній потік.

MR[2H], Z[2H] — 2/8 матриці MR та вектора Z

MR[3H], Z[3H] — 3/8 матриці MR та вектора Z

MR[2H], Z[2H] — 2/8 матриці MR та вектора Z

MR[H], Z[H] — 1/8 матриці MR та вектора Z

Cmin[X\*H] - X/8 вектора Cmin, where X = [1..5]

C[X\*H] - X/8 вектора C, where X = [1..4]

A[X\*H] - X/8 вектора A, where X = [1..4]

D, MX - вектор D та матриці MX

**2.4. Розробка програми ПРГ2.**

Програма була написана на мові Java з використанням бібліотеки

MPI. Складається з одного модуля: «Class.java».

Модуль “Class.java” містить клас Class та main. В них виконуються всі обчислення, ввод та інше.

Лістинг програми ПРГ2 наведено у додатку Г.

**2.5. Тестування програми ПРГ2.**

Таблиця 2.1 Час виконання програми для ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 8 | 6 | 4 | 3 | 2 | 1 |
| 898 | 782 | 1060 | 1412 | 2244 | 4320 | 3050 |
| 1800 | 1399 | 2634 | 3698 | 5261 | 8047 | 6227 |
| 2400 | 5068 | 4884 | 7160 | 9375 | 15956 | 13695 |

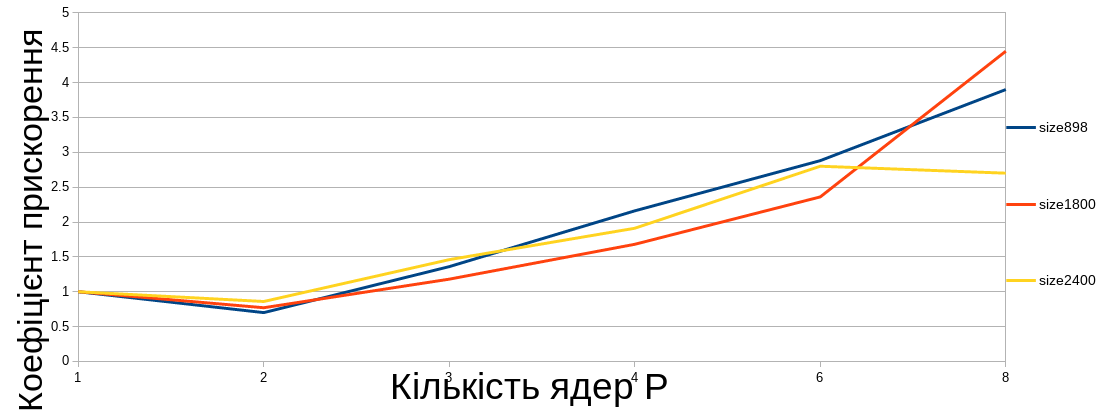
На основі даних із таблиці виконано розрахунок значень коефіцієнтів прискорення (КПI = T1 / TI) та ефективності (КЕІ = T1 / (TI \* I)\*100%), де І — кількість процесорів:

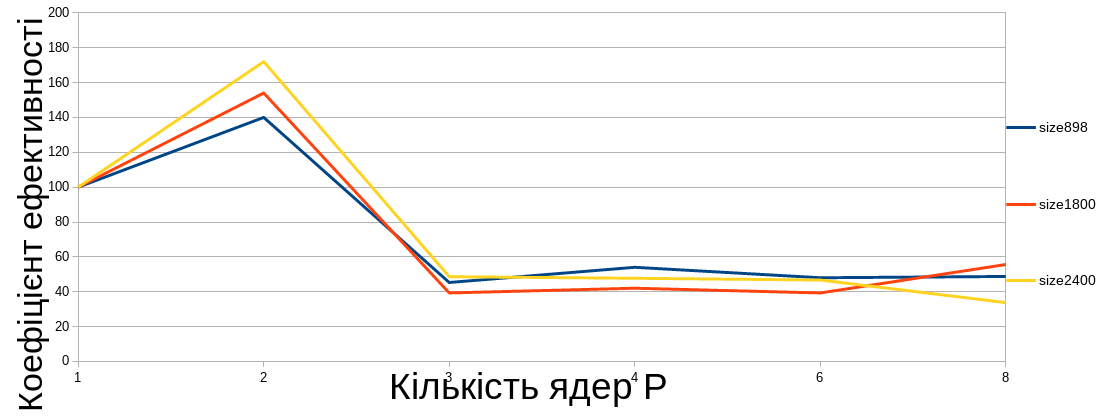
Таблиця 2.2 Значення Кп для ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 8 | 6 | 4 | 3 | 2 | 1 |
| 898 | 3.9 | 2.88 | 2.16 | 1.36 | 0.7 | 1 |
| 1800 | 4.45 | 2.36 | 1.68 | 1.18 | 0.77 | 1 |
| 2400 | 2.7 | 2.8 | 1.91 | 1.46 | 0.86 | 1 |

Таблиця 2.3 Значення Ке для ПРГ2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 8 | 6 | 4 | 3 | 2 | 1 |
| 900 | 48.75 | 48 | 54 | 45.3 | 140 | 100 |
| 1800 | 55.6 | 39.3 | 42 | 39.3 | 154 | 100 |
| 2400 | 33.75 | 46.7 | 47.7 | 48.6 | 172 | 100 |





**2.6 Висновок до розділу 2**

Виконано розробку програми ПРГ2 для ПКС2 з локальною пам’яттю з

використанням мови Java і MPI. Тестування програми

показало наступне: значення коефіцієнту прискорення лежать в межах від 0,7 до 0,86 для 2 потоків,1,18-1,46 — для 3 потоків, 1,68-2,16 для 4 потоків; 2,36-2,88 для 6 потоків; 2,7-4,45 для 8 потоків;

значення коефіцієнту ефективності лежать в межах від 140%-172% для 2 потоків, 39,3%-48,6% для 3 потоків, 42%-54 %, для 4 потоків, 39,3% - 48%для 6 потоків; 33,75%-55,6% для 8 потоків;

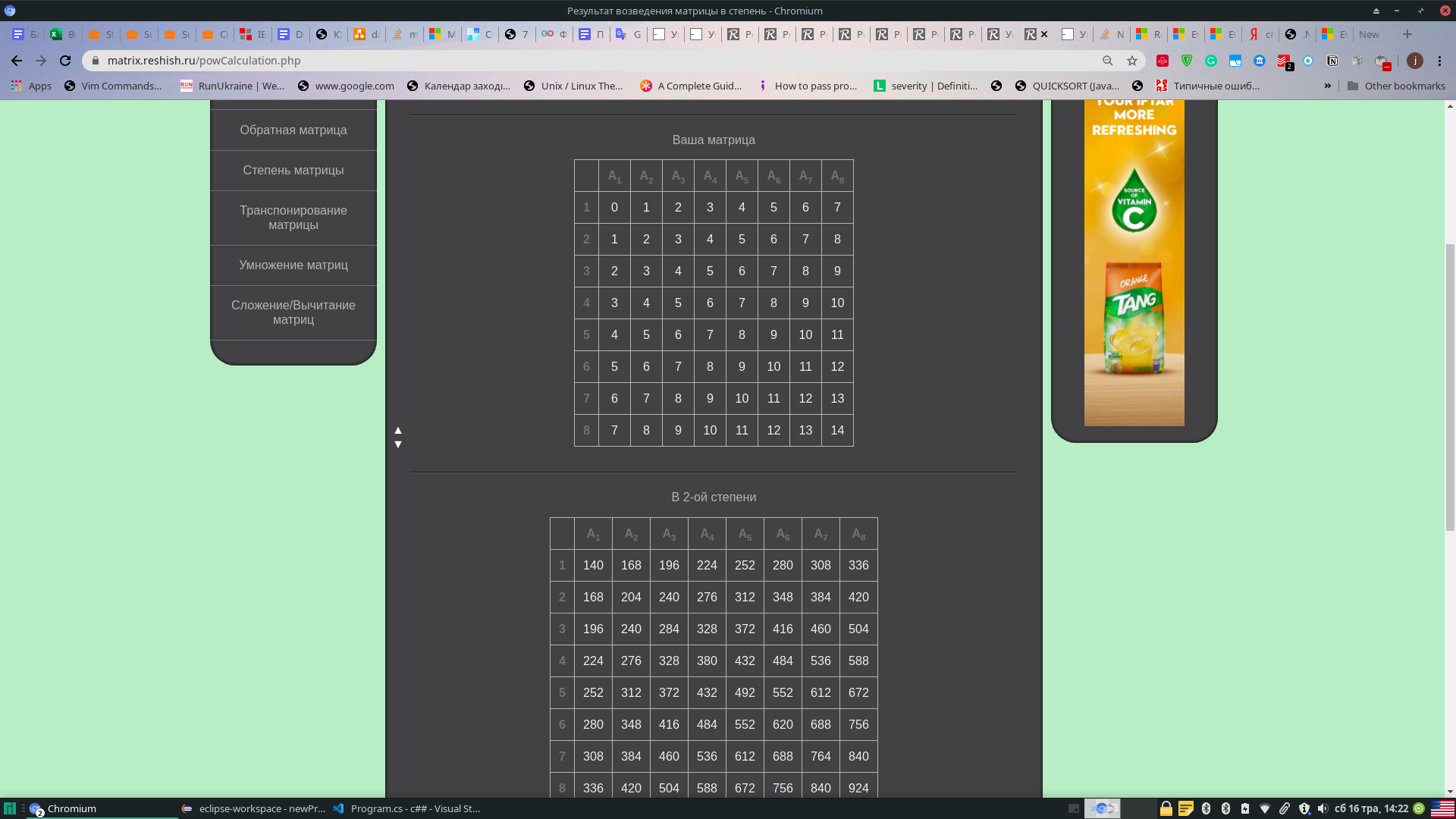
Отже, час виконання програми зменшується зі збільшенням кількості

потоків. Коефіцієнти ефективності говорять, що ми не можемо

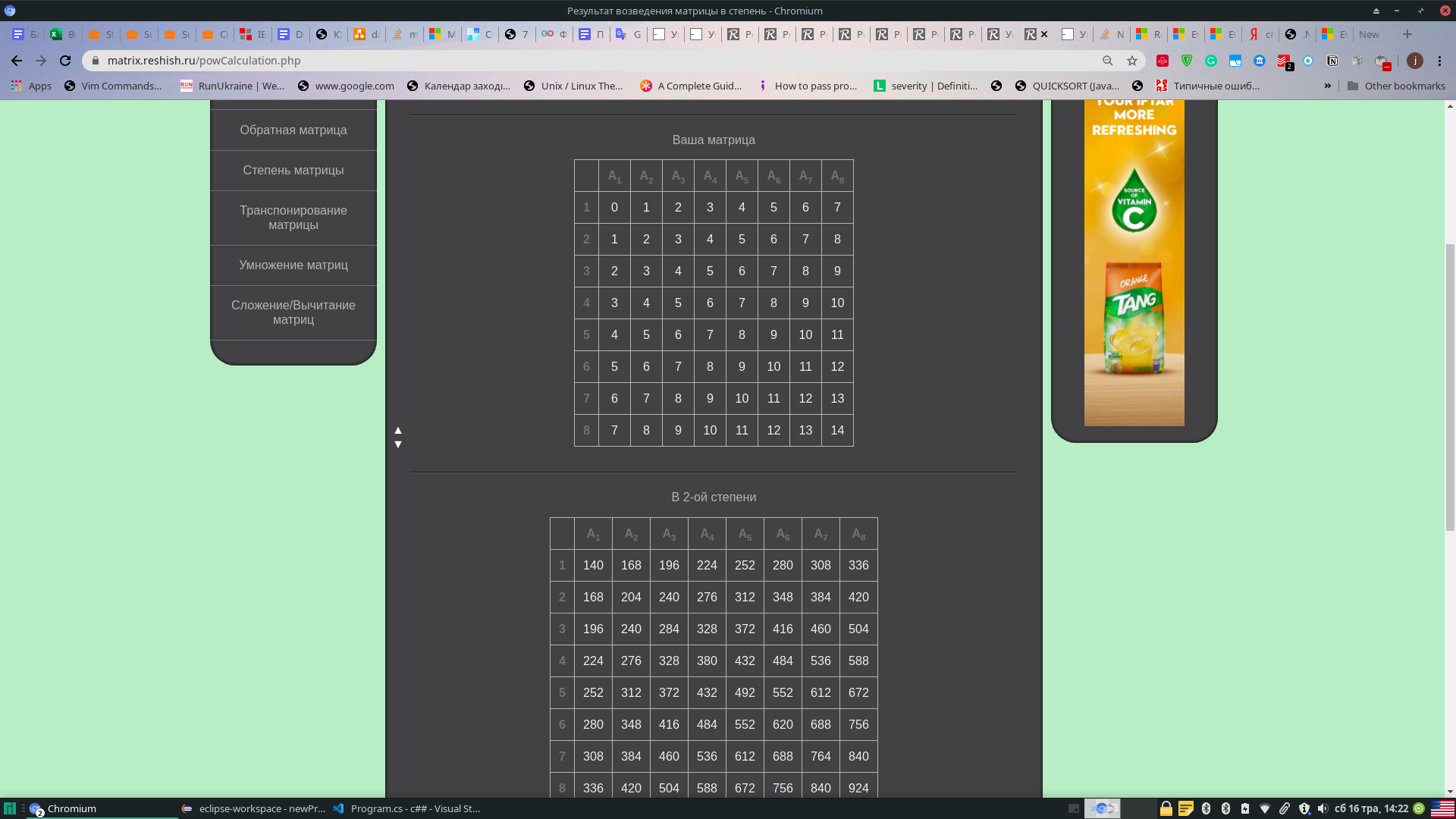
безкінечно ефективно збільшувати кількість потоків.

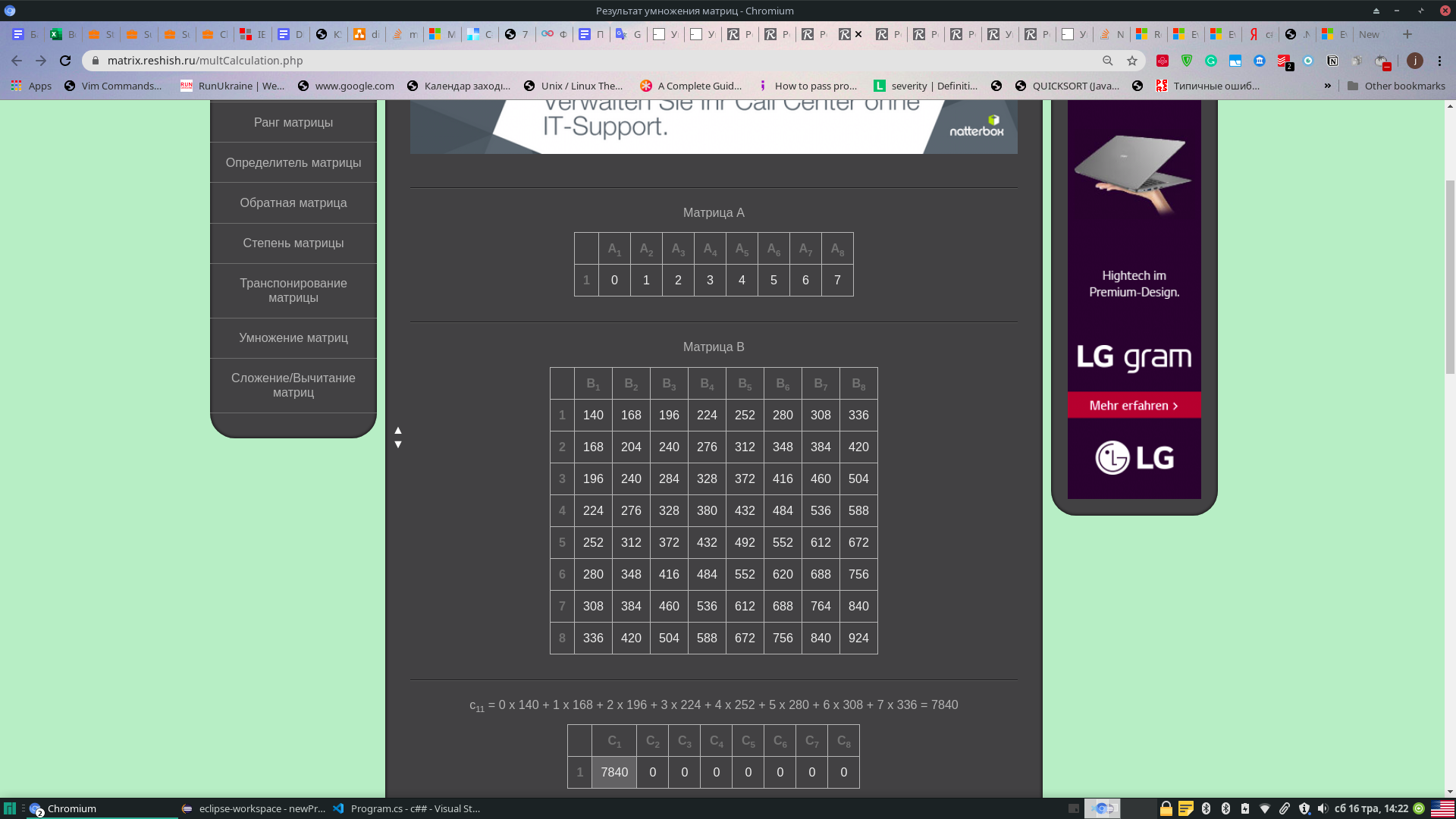
**ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ**

а) Доведення правильної виконання роботи програми зображено на мал1, мал2, мал3, мал4, мал5, мал6 - N=8; мал7, мал8,мал9 - N=16. На мал6 та мал9 зображено одинаковий результат ПРГ1 та ПРГ2.

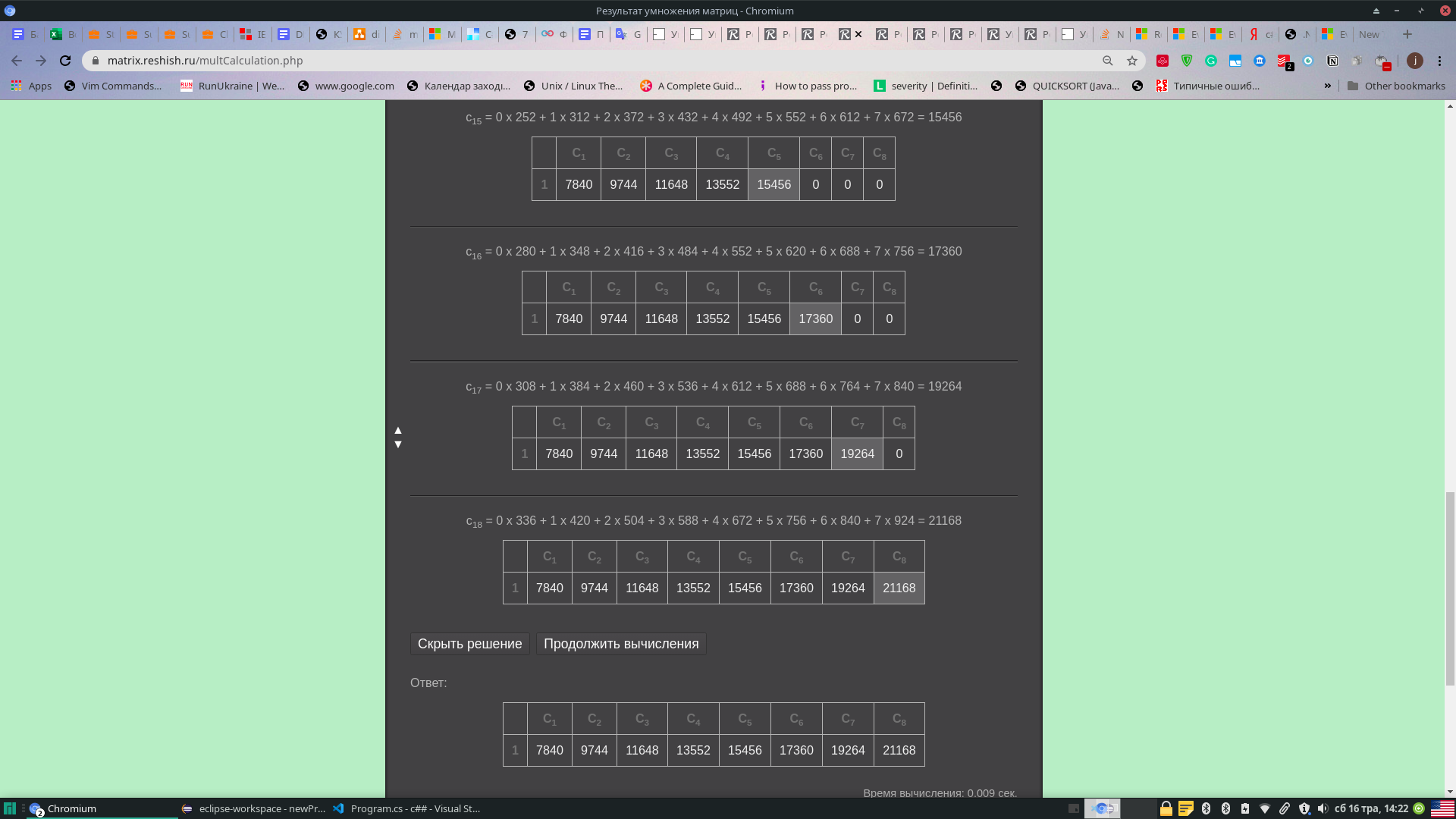


мал1. Виконання МX\*MR, мають однаковий розмір, тому можна возвести в квадрат.

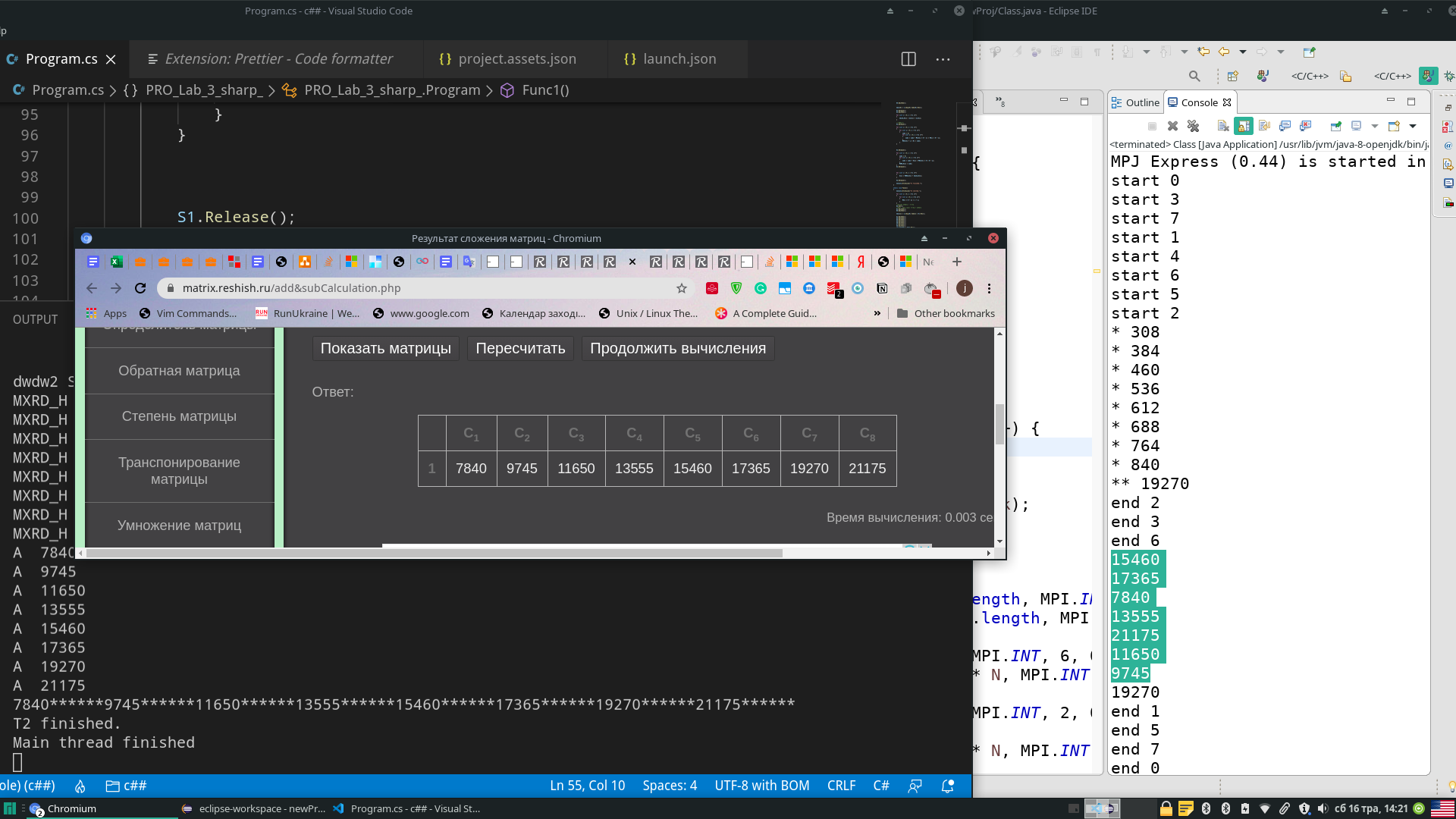
мал.3. MXR\_H = МR\*MX, N=8 

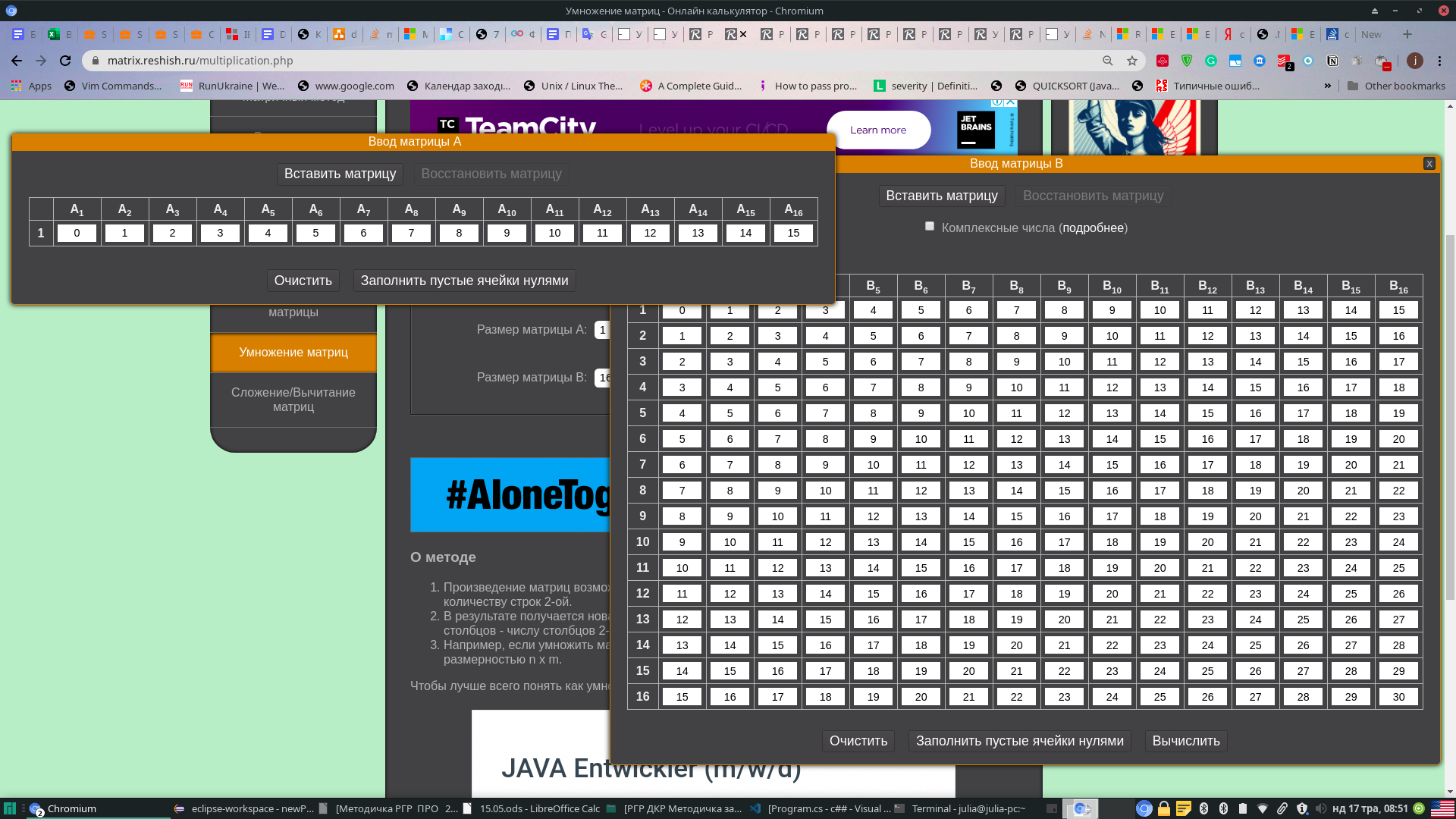


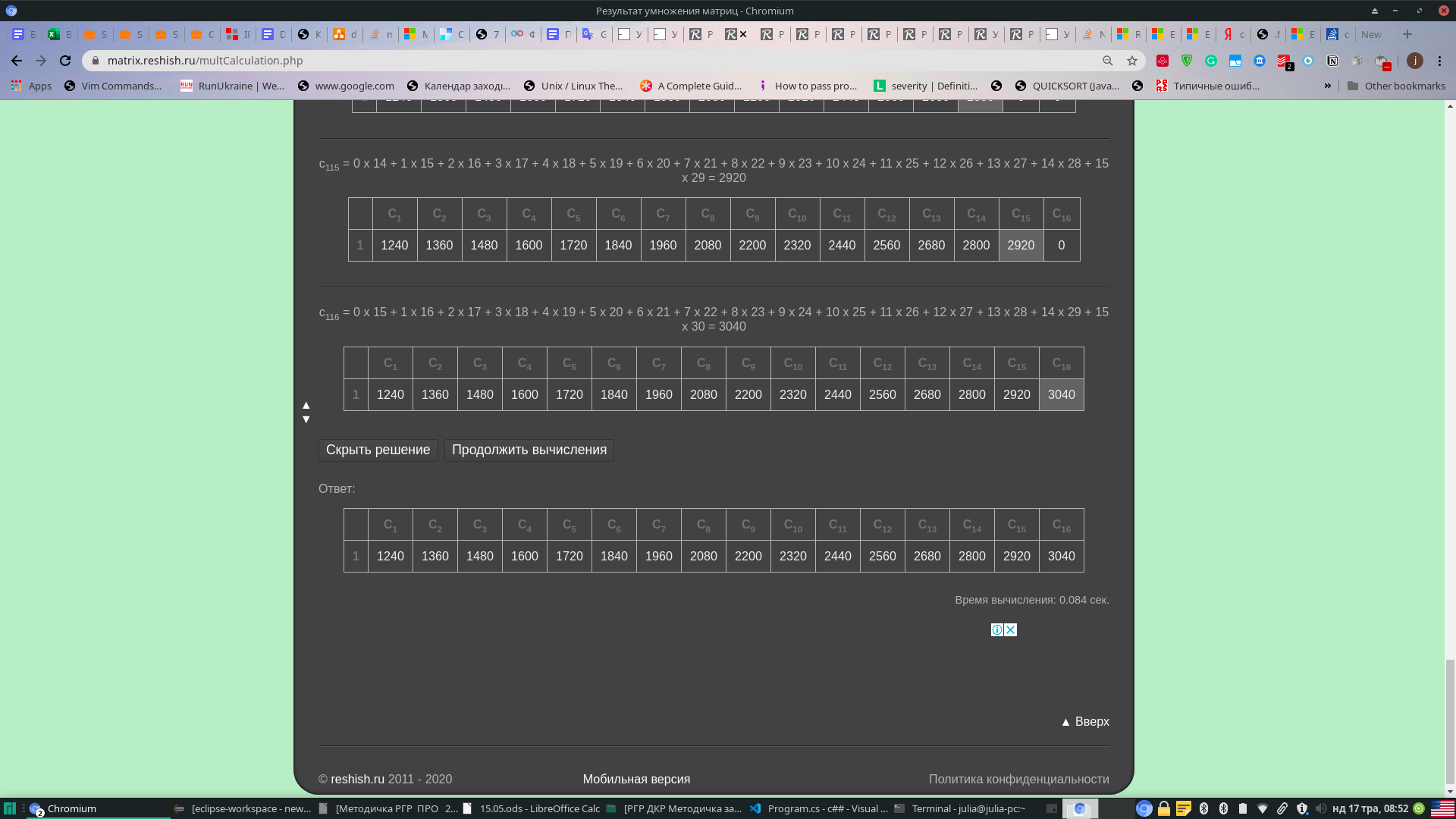
мал4. MXRD\_H = D\* MXR\_H; H=1; N=8;(ч.1)



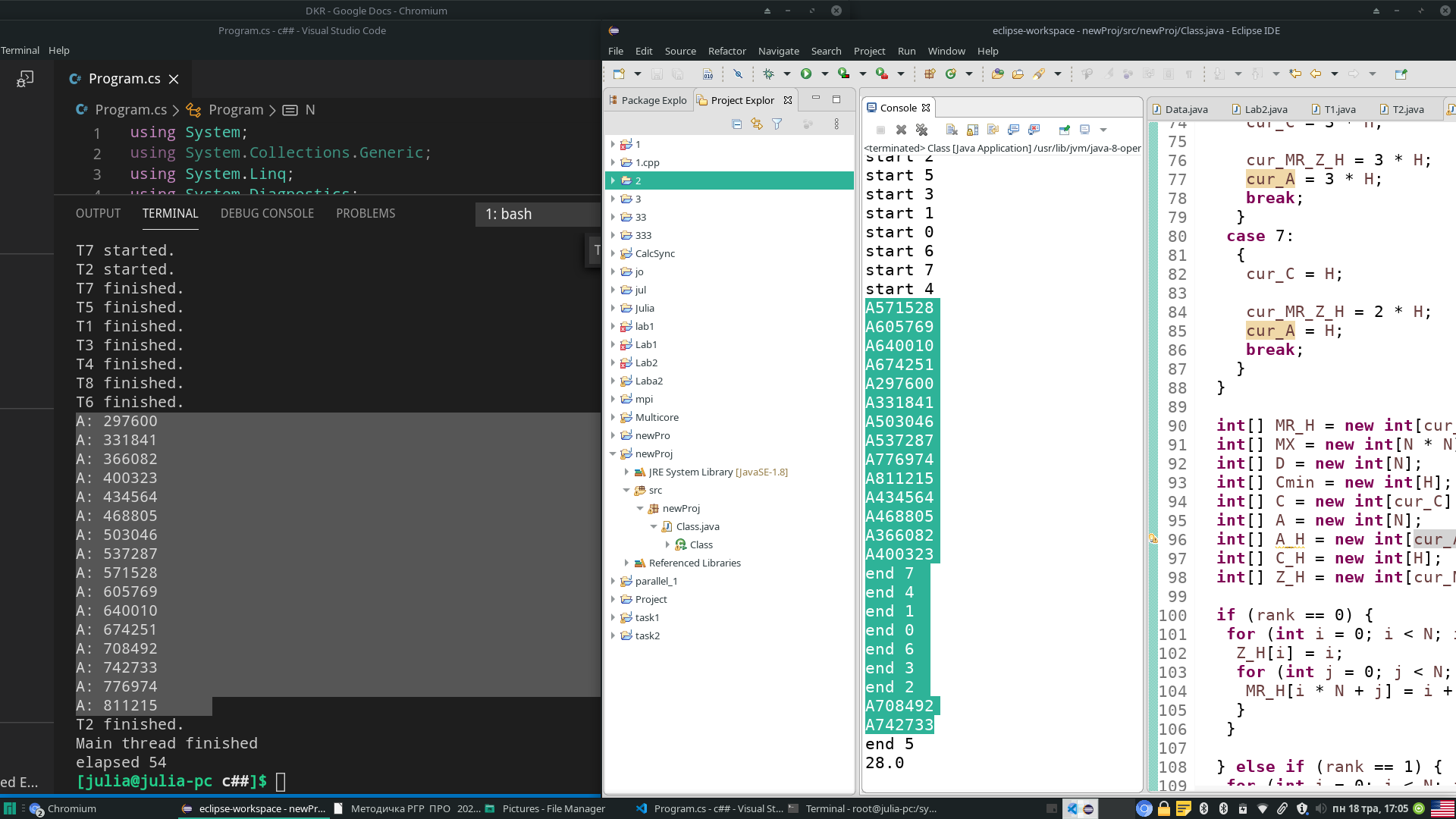
мал5. MXRD\_H = D\* MXR\_H; H=1; N=8;(ч.2)



мал6. Доведено коректність програми для Н=1, N=8.

мал7. Множення МR\_H\*MX при N=16

мал8. Результат MXR\_H, from (0..N-1), where N=16



мал9.Доведено коректність програми для Н=2, N=16.

б)У ході розрахунково-графічної роботи було розглянуто приклади ПКС зі спільною пам’яттю з використанням мови C# та ПКС з локальною пам’яттю за допомогою Java та бібліотеки MPI. Результатом даної роботи було розроблено 2 програми (ПРГ1 і ПРГ2), що забезпечують значне скорочення часу обчислення математичних задач на багатопоточних системах. Коефіцієнти прискорення мають максимальну дельту = 0,15 та коефіцієнт ефективності має максимальну дельту = 33 для матриць різної розмірності. Очевидно, що коефіцієнт ефективності у аналогічних тестах більший ніж у C#, тому, що у ПКС з локальною пам’яттю синхронізація відбувається за рахунок обміну повідомленнями, а, отже, ПРГ2 є ефективнішим та швидшим, про що говорять тестові дані, бо немає очікування та блокування потоків з боку семафорів та подій очікування(EventWaitHandler) в С# Win32. Отже, розглянуті системи реалізують паралельні обчислення, що є дуже ефективними, та хоча реалізація дуже різняться від мови програмування та засобів, але вони синтаксис та головні парадигми перетинаються.

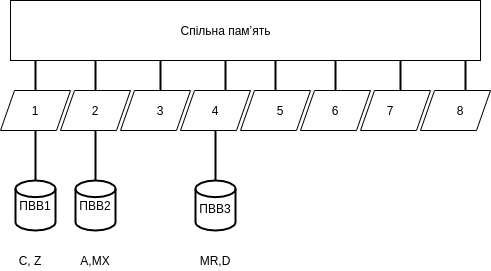
**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Жуков І. А., Корочкін О. В. Паралельні та розподілені обчислення. Навч.

Посібник — 2-ге вид. Виправл. і доп. К.: «Корнійчук», 2014. — 284 с

2. <https://docs.microsoft.com/ru-ru/dotnet/standard/threading/overview-of-synchronization-primitives>

**ДОДАТОК A СТРУКТУРНА СХЕМА ПКС1**

****

**ДОДАТОК Б Лістинг програмного коду для ПРГ1**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Diagnostics;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Microsoft.Win32;

//A = Cmin\*Z\_H + D\*(MX\*MR\_H);

class Program {

const int N = 16;

const int P = 8;

const int H = N / P;

static int[] C = new int[N];

static int[] A = new int[N];

static int[] MR\_H = new int[N \* N];

static int[] Z\_H = new int[N];

static int[] MX = new int[N \* N];

static int[] D = new int[N];

static int[] Cmin = new int[P];

static int[] MXR\_H = new int[N \* N];

static int[] MXRD\_H = new int[N];

static int[] CminZ\_H = new int[N];

static int sum1 = 0;

static int sum2 = 0;

static int sum3 = 0;

static int sum4 = 0;

static int sum5 = 0;

static int sum6 = 0;

static int sum7 = 0;

static int sum8 = 0;

static long minC;

static EventWaitHandle E1 = new ManualResetEvent(false);

static EventWaitHandle E2 = new ManualResetEvent(false);

static EventWaitHandle E4 = new ManualResetEvent(false);

static Semaphore S1 = new Semaphore(0, 6);

static Semaphore S2 = new Semaphore(0, 14);

static Semaphore S3 = new Semaphore(0, 6);

static Semaphore S4 = new Semaphore(0, 6);

static Semaphore S5 = new Semaphore(0, 6);

static Semaphore S6 = new Semaphore(0, 6);

static Semaphore S7 = new Semaphore(0, 6);

static Semaphore S8 = new Semaphore(0, 6);

static void Func1() {

Console.WriteLine("T1 started.");

for (int i = 0; i < N; i++) {

C[i] = i + 1;

Z\_H[i] = i;

}

//Сигнал о вводе

E1.Set();

//Ждать завершения ввода в others

E2.WaitOne();

E4.WaitOne();

Cmin[0] = C.Skip(0).Take(H).Min();

S1.Release();

S2.WaitOne();

for (int i = 0; i < H; i++) {

CminZ\_H[i] = Convert.ToInt32(Interlocked.Read(ref minC)) \* Z\_H[i];

}

// MXR\_H =

S1.Release();

for (int i = 0; i < N; i++) {

for (int j = 0; j < H; j++) {

sum1 = 0;

for (int z = 0; z < N; z++) {

sum1 += MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum1;

}

}

S1.Release();

for (int i = 0; i < H; i++) {

sum1 = 0;

for (int j = 0; j < N; j++) {

sum1 = sum1 + D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum1;

}

S1.Release();

for (int i = 0; i < H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T1 finished.");

}

static void Func2() {

Console.WriteLine("T2 started.");

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

MX[i \* N + j] = i + j;

}

}

//Сигнал others о вводе

E2.Set();

//Ждать завершения ввода в others

E1.WaitOne();

E4.WaitOne();

Cmin[1] = C.Skip(H).Take(2 \* H).Min();

S1.WaitOne();

S3.WaitOne();

S4.WaitOne();

S5.WaitOne();

S6.WaitOne();

S7.WaitOne();

S8.WaitOne();

int minimal = Cmin.Min();

Interlocked.Add(ref minC, minimal);

S2.Release();

S2.Release();

S2.Release();

S2.Release();

S2.Release();

S2.Release();

S2.Release();

for (int i = H; i < 2 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* minimal;

}

S1.WaitOne();

S3.WaitOne();

S4.WaitOne();

S5.WaitOne();

S6.WaitOne();

S7.WaitOne();

S8.WaitOne();

// MXR\_H =

for (int i = 0; i < N; i++) {

for (int j = H; j < 2 \* H; j++) {

sum2 = 0;

for (int z = 0; z < N; z++) {

sum2 += MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum2;

}

}

S1.WaitOne();

S3.WaitOne();

S4.WaitOne();

S5.WaitOne();

S6.WaitOne();

S7.WaitOne();

S8.WaitOne();

for (int i = H; i < 2 \* H; i++) {

sum2 = 0;

for (int j = 0; j < N; j++) {

sum2 += D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum2;

}

S1.WaitOne();

S3.WaitOne();

S4.WaitOne();

S5.WaitOne();

S6.WaitOne();

S7.WaitOne();

S8.WaitOne();

for (int i = H; i < 2 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.WaitOne();

S2.WaitOne();

S2.WaitOne();

S2.WaitOne();

S2.WaitOne();

S2.WaitOne();

S2.WaitOne();

for (var i = 0; i < N; i++) {

Console.Write("A: " + A[i].ToString().PadLeft(4));

Console.WriteLine();

}

Console.WriteLine("T2 finished.");

}

static void Func3() {

for (int i = 2 \* H; i < 3 \* H; i++)

Cmin[2] = C.Skip(2 \* H).Take(3 \* H).Min();

S3.Release();

S2.WaitOne();

for (int i = 2 \* H; i < 3 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* Convert.ToInt32(Interlocked.Read(ref minC));

}

// MXR\_H =

S3.Release();

for (int i = 0; i < N; i++) {

for (int j = 2 \* H; j < 3 \* H; j++) {

sum3 = 0;

for (int z = 0; z < N; z++) {

sum3 += MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum3;

}

}

S3.Release();

for (int i = 2 \* H; i < 3 \* H; i++) {

sum3 = 0;

for (int j = 0; j < N; j++) {

sum3 += D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum3;

}

S3.Release();

for (int i = 2 \* H; i < 3 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T3 finished.");

}

static void Func4() {

Console.WriteLine("T4 started.");

//Ввод МR, D

for (int i = 0; i < N; i++) {

D[i] = i;

for (int j = 0; j < N; j++) {

MR\_H[i \* N + j] = i + j;

}

}

//Сигнал Т1,Т3 о вводе

E4.Set();

//Ждать завершения ввода в Т1,Т3

E1.WaitOne();

E2.WaitOne();

//Счёт1

for (int i = 3 \* H; i < 4 \* H; i++)

Cmin[3] = C.Skip(3 \* H).Take(4 \* H).Min();

S4.Release();

S2.WaitOne();

for (int i = 3 \* H; i < 4 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* Convert.ToInt32(Interlocked.Read(ref minC));

}

// MXR\_H =

S4.Release();

for (int i = 0; i < N; i++) {

for (int j = 3 \* H; j < 4 \* H; j++) {

sum4 = 0;

for (int z = 0; z < N; z++) {

sum4 += MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum4;

}

}

S4.Release();

for (int i = 3 \* H; i < 4 \* H; i++) {

sum4 = 0;

for (int j = 0; j < N; j++) {

sum4 = sum4 + D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum4;

}

S4.Release();

for (int i = 3 \* H; i < 4 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T4 finished.");

}

static void Func5() {

Console.WriteLine("T5 started.");

for (int i = 4 \* H; i < 5 \* H; i++)

Cmin[4] = C.Skip(4 \* H).Take(5 \* H).Min();

S5.Release();

S2.WaitOne();

for (int i = 4 \* H; i < 5 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* Convert.ToInt32(Interlocked.Read(ref minC));

}

// MXR\_H =

S5.Release();

for (int i = 0; i < N; i++) {

for (int j = 4 \* H; j < 5 \* H; j++) {

sum5 = 0;

for (int z = 0; z < N; z++) {

sum5 = sum5 + MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum5;

}

}

S5.Release();

for (int i = 4 \* H; i < 5 \* H; i++) {

sum5 = 0;

for (int j = 0; j < N; j++) {

sum5 = sum5 + D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum5;

}

S5.Release();

for (int i = 4 \* H; i < 5 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T5 finished.");

}

static void Func6() {

Console.WriteLine("T6 started.");

for (int i = 5 \* H; i < 6 \* H; i++)

Cmin[5] = C.Skip(5 \* H).Take(6 \* H).Min();

S6.Release();

S2.WaitOne();

for (int i = 5 \* H; i < 6 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* Convert.ToInt32(Interlocked.Read(ref minC));

}

// MXR\_H =

S6.Release();

for (int i = 0; i < N; i++) {

for (int j = 5 \* H; j < 6 \* H; j++) {

sum6 = 0;

for (int z = 0; z < N; z++) {

sum6 = sum6 + MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum6;

}

}

S6.Release();

for (int i = 5 \* H; i < 6 \* H; i++) {

sum6 = 0;

for (int j = 0; j < N; j++) {

sum6 += D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum6;

}

S6.Release();

for (int i = 5 \* H; i < 6 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T6 finished.");

}

static void Func7() {

Console.WriteLine("T7 started.");

for (int i = 6 \* H; i < 7 \* H; i++)

Cmin[6] = C.Skip(6 \* H).Take(7 \* H).Min();

S7.Release();

S2.WaitOne();

for (int i = 6 \* H; i < 7 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* Convert.ToInt32(Interlocked.Read(ref minC));

}

// MXR\_H =

S7.Release();

for (int i = 0; i < N; i++) {

for (int j = 6 \* H; j < 7 \* H; j++) {

sum7 = 0;

for (int z = 0; z < N; z++) {

sum7 = sum7 + MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum7;

}

}

S7.Release();

for (int i = 6 \* H; i < 7 \* H; i++) {

sum7 = 0;

for (int j = 0; j < N; j++) {

sum7 += D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum7;

}

S7.Release();

for (int i = 6 \* H; i < 7 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T7 finished.");

}

static void Func8() {

Console.WriteLine("T8 started.");

for (int i = 7 \* H; i < 8 \* H; i++)

Cmin[7] = C.Skip(7 \* H).Take(8 \* H).Min();

S8.Release();

S2.WaitOne();

for (int i = 7 \* H; i < 8 \* H; i++) {

CminZ\_H[i] = Z\_H[i] \* Convert.ToInt32(Interlocked.Read(ref minC));

}

S8.Release();

for (int i = 0; i < N; i++) {

for (int j = 7 \* H; j < 8 \* H; j++) {

sum8 = 0;

for (int z = 0; z < N; z++) {

sum8 = sum8 + MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i + j \* N] = sum8;

}

}

S8.Release();

for (int i = 7 \* H; i < 8 \* H; i++) {

sum8 = 0;

for (int j = 0; j < N; j++) {

sum8 = sum8 + D[j] \* MXR\_H[i \* N + j];

}

MXRD\_H[i] = sum8;

}

S8.Release();

for (int i = 7 \* H; i < 8 \* H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

S2.Release();

Console.WriteLine("T8 finished.");

}

static void Main(string[] args) {

var sw = new Stopwatch();

sw.Start();

Console.WriteLine("Main thread started.");

Thread T1 = new Thread(Func1);

Thread T2 = new Thread(Func2);

Thread T3 = new Thread(Func3);

Thread T4 = new Thread(Func4);

Thread T5 = new Thread(Func5);

Thread T6 = new Thread(Func6);

Thread T7 = new Thread(Func7);

Thread T8 = new Thread(Func8);

T1.Start();

T2.Start();

T3.Start();

T4.Start();

T5.Start();

T6.Start();

T7.Start();

T8.Start();

T2.Join();

Console.WriteLine("Main thread finished");

// Console.Read();

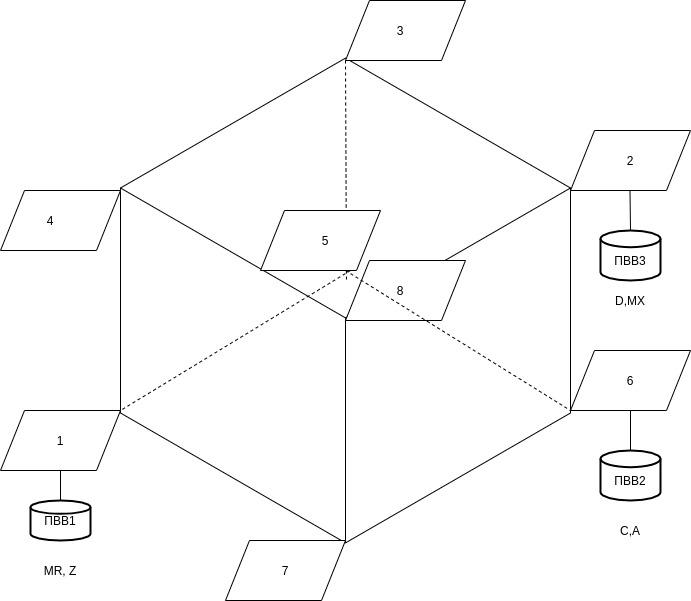
sw.Stop();

Console.WriteLine("elapsed " + sw.ElapsedMilliseconds);

}

}

**ДОДАТОК B СТРУКТУРНА СХЕМА ПКС2**



**ДОДАТОК Г Лістинг програмного коду для ПРГ2**

package newProj;

import mpi.\*;

import java.util.Arrays;

public class Class {

public static void main(String[] args) throws Exception {

MPI.Init(args);

int rank = MPI.COMM\_WORLD.Rank();

int P = 8;

int N = 16;

int H = N / P;

int[] MXR\_H = new int[N \* N];

int[] MXRD\_H = new int[N];

int[] CminZ\_H = new int[H];

int sum1;

//input vectors and matrixes

int cur\_MR\_Z\_H = 0, cur\_A = 0, cur\_C = 0;

long m = System.currentTimeMillis();

switch (rank) {

case 0:

{

cur\_C = 2 \* H;

cur\_MR\_Z\_H = N;

cur\_A = 2 \* H;

break;

}

case 1:

{

cur\_C = 4 \* H;

cur\_MR\_Z\_H = H;

cur\_A = 4 \* H;

break;

}

case 2:

{

cur\_C = 3 \* H;

cur\_MR\_Z\_H = H;

cur\_A = 3 \* H;

break;

}

case 3:

{

cur\_C = 2 \* H;

cur\_MR\_Z\_H = 2 \* H;

cur\_A = 2 \* H;

break;

}

case 4:

{

cur\_C = H;

cur\_MR\_Z\_H = 2 \* H;

cur\_A = H;

break;

}

case 5:

{

cur\_C = N;

cur\_MR\_Z\_H = H;

cur\_A = N;

break;

}

case 6:

{

cur\_C = 3 \* H;

cur\_MR\_Z\_H = 3 \* H;

cur\_A = 3 \* H;

break;

}

case 7:

{

cur\_C = H;

cur\_MR\_Z\_H = 2 \* H;

cur\_A = H;

break;

}

}

int[] MR\_H = new int[cur\_MR\_Z\_H \* N];

int[] MX = new int[N \* N];

int[] D = new int[N];

int[] Cmin = new int[H];

int[] C = new int[cur\_C];

int[] A = new int[N];

int[] A\_H = new int[cur\_A];

int[] C\_H = new int[H];

int[] Z\_H = new int[cur\_MR\_Z\_H];

if (rank == 0) {

for (int i = 0; i < N; i++) {

Z\_H[i] = i;

for (int j = 0; j < N; j++) {

MR\_H[i \* N + j] = i + j;

}

}

} else if (rank == 1) {

for (int i = 0; i < N; i++) {

D[i] = i;

for (int j = 0; j < N; j++) {

MX[i \* N + j] = i + j;

}

}

} else if (rank == 5) {

for (int i = 0; i < C.length; i++) {

C[i] = i + 1;

}

}

System.out.println("start " + rank);

MPI.COMM\_WORLD.Barrier();

if (rank == 0) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 6, 0);

} else if (rank == 1) {

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 5, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 5, 0);

} else if (rank == 2) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 1, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 1, 0);

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 4, 0);

} else if (rank == 3) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 0, 0);

} else if (rank == 4) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(D, 0, N, MPI.INT, 7, 0);

MPI.COMM\_WORLD.Send(MX, 0, N \* N, MPI.INT, 7, 0);

} else if (rank == 5) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 1, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 1, 0);

} else if (rank == 6) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 0, 0);

} else if (rank == 7) {

MPI.COMM\_WORLD.Recv(D, 0, D.length, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Recv(MX, 0, MX.length, MPI.INT, 4, 0);

}

MPI.COMM\_WORLD.Barrier();

if (rank == 0) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Send(C, H, H, MPI.INT, 4, 0);

} else if (rank == 1) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 5, 0);

MPI.COMM\_WORLD.Send(C, H, 3 \* H, MPI.INT, 2, 0);

} else if (rank == 2) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 1, 0);

MPI.COMM\_WORLD.Send(C, H, 2 \* H, MPI.INT, 3, 0);

} else if (rank == 3) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(C, H, H, MPI.INT, 7, 0);

} else if (rank == 4) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 0, 0);

} else if (rank == 5) {

MPI.COMM\_WORLD.Send(C, H, 3 \* H, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Send(C, 4 \* H, 4 \* H, MPI.INT, 1, 0);

} else if (rank == 6) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 5, 0);

MPI.COMM\_WORLD.Send(C, H, 2 \* H, MPI.INT, 0, 0);

} else if (rank == 7) {

MPI.COMM\_WORLD.Recv(C, 0, C.length, MPI.INT, 3, 0);

}

//MR Z

if (rank == 0) {

//Alternative way

//MPI.COMM\_WORLD.Isend(MR\_H, H \* N, 2 \* H \* N, MPI.INT, 3, 0)

MPI.COMM\_WORLD.Send(MR\_H, H \* N, 2 \* H \* N, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(Z\_H, H, 2 \* H, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(MR\_H, 3 \* H \* N, 2 \* H \* N, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Send(Z\_H, 3 \* H, 2 \* H, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Send(MR\_H, 5 \* H \* N, 3 \* H \* N, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Send(Z\_H, 5 \* H, 3 \* H, MPI.INT, 6, 0);

} else if (rank == 1) {

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 7, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 7, 0);

} else if (rank == 2) {

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 3, 0);

} else if (rank == 3) {

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Send(MR\_H, H \* N, H \* N, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(Z\_H, H, H, MPI.INT, 2, 0);

} else if (rank == 4) {

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Send(MR\_H, H \* N, H \* N, MPI.INT, 5, 0);

MPI.COMM\_WORLD.Send(Z\_H, H, H, MPI.INT, 5, 0);

} else if (rank == 5) {

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 4, 0);

} else if (rank == 6) {

// Alternative way

//MPI.COMM\_WORLD.Irecv(MR\_H, 0, MR\_H.length, MPI.INT, 0, 0)

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Send(MR\_H, H \* N, 2 \* H \* N, MPI.INT, 7, 0);

MPI.COMM\_WORLD.Send(Z\_H, H, 2 \* H, MPI.INT, 7, 0);

} else if (rank == 7) {

MPI.COMM\_WORLD.Recv(MR\_H, 0, MR\_H.length, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Recv(Z\_H, 0, Z\_H.length, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Send(MR\_H, H \* N, H \* N, MPI.INT, 1, 0);

MPI.COMM\_WORLD.Send(Z\_H, H, H, MPI.INT, 1, 0);

}

//Find the certain a slice array to examine for min values.

if (rank == 0) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 1) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 2) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 3) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 4) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 5) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 6) {

C\_H = Arrays.copyOfRange(C, 0, H);

} else if (rank == 7) {

C\_H = Arrays.copyOfRange(C, 0, H);

}

//Find Min from all arrays and send it to the processor 5

MPI.COMM\_WORLD.Reduce(C\_H, 0, Cmin, 0, C\_H.length, MPI.INT, MPI.MIN, 5);

//Send Cmin for all processors

if (rank == 0) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 6, 0);

} else if (rank == 1) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 5, 0);

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 2, 0);

} else if (rank == 2) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 1, 0);

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 3, 0);

} else if (rank == 3) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 0, 0);

} else if (rank == 4) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 5, 0);

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 7, 0);

} else if (rank == 5) {

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Send(Cmin, 0, Cmin.length, MPI.INT, 1, 0);

} else if (rank == 6) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 0, 0);

} else if (rank == 7) {

MPI.COMM\_WORLD.Recv(Cmin, 0, Cmin.length, MPI.INT, 4, 0);

}

//A\_H = Cmin\*Z\_H + DH \* (MX\*MR\_H);

for (int i = 0; i < N; i++) {

for (int j = 0; j < H; j++) {

sum1 = 0;

for (int z = 0; z < N; z++) {

sum1 = sum1 + MR\_H[j \* N + z] \* MX[i \* N + z];

}

MXR\_H[i \* N + j] = sum1;

}

}

for (int i = 0; i < H; i++) {

sum1 = 0;

for (int j = 0; j < N; j++) {

sum1 += D[j] \* MXR\_H[j \* N + i];

}

MXRD\_H[i] = sum1;

}

for (int i = 0; i < H; i++) {

CminZ\_H[i] = Z\_H[i] \* Cmin[0];

}

for (int i = 0; i < H; i++) {

A[i] = MXRD\_H[i] + CminZ\_H[i];

}

//Send AH parts and gather them for getting the answer

if (rank == 0) {

MPI.COMM\_WORLD.Recv(A, H, H, MPI.INT, 4, 0);

MPI.COMM\_WORLD.Send(A, 0, 2 \* H, MPI.INT, 6, 0);

} else if (rank == 1) {

MPI.COMM\_WORLD.Recv(A, H, 3 \* H, MPI.INT, 2, 0);

MPI.COMM\_WORLD.Send(A, 0, 4 \* H, MPI.INT, 5, 0);

} else if (rank == 2) {

MPI.COMM\_WORLD.Recv(A, H, 2 \* H, MPI.INT, 3, 0);

MPI.COMM\_WORLD.Send(A, 0, 3 \* H, MPI.INT, 1, 0);

} else if (rank == 3) {

MPI.COMM\_WORLD.Recv(A, H, H, MPI.INT, 7, 0);

MPI.COMM\_WORLD.Send(A, 0, 2 \* H, MPI.INT, 2, 0);

} else if (rank == 4) {

MPI.COMM\_WORLD.Send(A, 0, H, MPI.INT, 0, 0);

} else if (rank == 5) {

MPI.COMM\_WORLD.Recv(A, H, 3 \* H, MPI.INT, 6, 0);

MPI.COMM\_WORLD.Recv(A, 4 \* H, 4 \* H, MPI.INT, 1, 0);

} else if (rank == 6) {

MPI.COMM\_WORLD.Recv(A, H, 2 \* H, MPI.INT, 0, 0);

MPI.COMM\_WORLD.Send(A, 0, 3 \* H, MPI.INT, 5, 0);

} else if (rank == 7) {

MPI.COMM\_WORLD.Send(A, 0, H, MPI.INT, 3, 0);

}

MPI.COMM\_WORLD.Barrier();

if (rank == 5) {

for (int i = 0; i < A.length; i++) {

System.out.println("A" + A[i]);

}

}

System.out.println("end " + rank + " ");

MPI.Finalize();

double[] v = new double[1];

v[0] = (System.currentTimeMillis() - m);

double[] answer = new double[1];

MPI.COMM\_WORLD.Reduce(v, 0, answer, 0, 1, MPI.DOUBLE, MPI.MAX, 0);

if (rank == 0) {

System.out.println(answer[0]);

}

}

}