НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки

**КУРСОВА РОБОТА**

з дисципліни «Паралельні та розподілені обчислення»

на тему: «Розробка програмного забезпечення для паралельних

комп’ютерних систем»

Студента 3 курсу групи ІО-01

напряму підготовки 050102

«Комп’ютерна інженерія»

Середенко А. С.

Керівник доцент Корочкін О.В.

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ - 2013 рік

Національний технічний університет України

«Київський політехнічний інститут»

Факультет (інститут) інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

***З А В Д А Н Н Я***

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Середенка Андрія Станіславовича

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 11 травня 2013 р.

3. Вихідні дані до роботи

- огляд і порівняння чотирьох’ядерних процесорів компаній AMD і Intel

- математична задача A=(B\*C)\*Z + R\*(MO \* MT)

- структури ПКС ОП та ПКС ЛП

- мови програмування: C#, Ada

- засоби організації взаємодії процесів: семафори, м’ютекси, замки, критичні секції, події, атомарні змінні мови C#, механізм рандеву мови Ada

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно

розробити)

- огляд і порівняння чотирьох’ядерних процесорів компаній AMD і Intel

- розробка і тестування програми ПРГ1 для ПКС ОП

- розробка і тестування програми ПРГ2 для ПКС ЛП

5. Перелік графічного матеріалу

- структурна схема ПКС ОП

- структурна схема ПКС ЛП

- схеми алгоритмів процесів і головної програми для ПРГ1

- схеми алгоритмів процесів і головної програми для ПРГ2.

6. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_

*КАЛЕНДАРНИЙ ПЛАН*

|  |  |  |
| --- | --- | --- |
| №  з/п | Назва етапів виконання КР | Строк виконання етапів КР |
| 1 | Виконання огляду для розділу 1 | 20.03.2013 |
| 2 | Розробка паралельного алгоритму  рішення задачі | 1.04.2013 |
| 3 | Розробка алгоритмів процесів | 6.04.2014 |
| 4 | Розробка схем взаємодії процесів | 13.04.2013 |
| 5 | Розробка програм | 20.04.2013 |
| 6 | Тестування програм | 30.04.2013 |
| 7 | Оформлення КР | 10.05.2013 |
| 8 | Захист КР | 18.05.2013 |

**Студент** \_\_\_\_\_\_\_\_\_ Середенко А.С.

( підпис )

**Керівник роботи** \_\_\_\_\_\_\_\_\_ Корочкін О.В.

( підпис )

ТЕХНІЧНЕ ЗАВДАННЯ

C#

Рис. 1. Структура паралельної обчислювальної системи зі спільною пам’яттю.  
Ада. Рандеву

Рис. 2. Структура паралельної обчислювальної системи з локальною пам’яттю.

Матричне рівняння  
A=(B \* C) \* Z + R \* (MO \* MT)

ЗМІСТ

[РОЗДІЛ 1. ОГЛЯД ЧЬОТИРЬОХ’ЯДЕРНИХ ПРОЦЕСОРІВ 8](#_Toc356167018)

[1.1 Процесори корпорації Intel 8](#_Toc356167019)

[1.1.1 Процесори Intel Core i7 8](#_Toc356167020)

[1.1.2 Процесори Intel Core i5 11](#_Toc356167021)

[1.1.3 Сімейство процесорів Intel Core 2 12](#_Toc356167022)

[1.2 Процесори корпорації AMD 15](#_Toc356167023)

[1.2.1 Технологии корпорації AMD 18](#_Toc356167024)

[1.2.2 64-розрядні процесори AMD 19](#_Toc356167025)

[1.2.3 Процесори AMD Phenom II і Phenom 20](#_Toc356167026)

[1.2.4 AMD Quad FX 22](#_Toc356167027)

[1.3 Тестування процесорів 24](#_Toc356167028)

[1.4 Висновки до розділу 1 25](#_Toc356167029)

[РОЗДІЛ 2. Розробка програми ПРГ1 для ПКС СП 27](#_Toc356167030)

[2.1 Аналіз задачі на внутрішній паралелізм з використанням концепції необмеженого паралелізму. 27](#_Toc356167031)

[2.2 Розробка паралельного математичного алгоритму 28](#_Toc356167032)

[2.3 Розробка алгоритмів процесів 29](#_Toc356167033)

[2.4 Розробка схеми взаємодії процесів 32](#_Toc356167034)

[2.5 Розробка програми ПРГ1 33](#_Toc356167035)

[2.6 Тестування програми ПРГ1 33](#_Toc356167036)

[2.6 Висновки до розділу 2 38](#_Toc356167037)

[РОЗДІЛ 3. Розробка програми ПРГ1 для ПКС з ЛП 39](#_Toc356167038)

[3.1 Розробка алгоритмів процесів 39](#_Toc356167039)

[3.2 Розробка схеми взаємодії задач 42](#_Toc356167040)

[3.3. Розробка програми 43](#_Toc356167041)

[3.4 Тестування програми ПРГ2 44](#_Toc356167042)

[3.5 Висновки до Розділу 3 49](#_Toc356167043)

[ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ 50](#_Toc356167044)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 51](#_Toc356167045)

[ДОДАТКИ 54](#_Toc356167046)

[Додаток А. Схеми алгоритмів процесів 55](#_Toc356167047)

[Додаток Б. Схеми алгоритмів основних програм 56](#_Toc356167048)

[Додаток В. Лістинг програм 57](#_Toc356167049)

[Лістинг програми для системи зі спільною пам’яттю 58](#_Toc356167050)

[Лістинг програми для системи з локальною пам’яттю 69](#_Toc356167051)

**ВСТУП**

Багатоядерний процесор складається з двох і більше «обчислювальних ядер» на одному кристалі. Він має один корпус і встановляється в один роз'єм на системній платі комп'ютера, але операційна система сприймає кожне його обчислювальне ядро як окремий процесор з повним набором обчислювальних ресурсів [1]. На сьогоднішній день найбільш популярні процесори виробляють фірми Intel і AMD. Всі ці фірми займаються виробництвом мультиядерних процесорів для настільних комп'ютерів.

Intel Corporation — найбільша у світі напівпровідникова компанія та виробник x86-серії мікропроцесорів, процесорів для IBM-сумісних персональних комп'ютерів [2]. ЇЇ багаторічним суперником є Advanced Micro Devices, Inc. (AMD), що випускають свої процесори під торговими марками Athlon, Phenom (настільні комп'ютери і робочі станції), Duron, Sempron (бюджетні варіанти), Opteron (для серверів), Turion (для мобільних комп'ютерів, ноутбуків). [3]

# РОЗДІЛ 1. ОГЛЯД ЧЬОТИРЬОХ’ЯДЕРНИХ ПРОЦЕСОРІВ

## 1.1 Процесори корпорації Intel

Корпорація Intel виробляє величезну номенклатуру найрізноманітніших напівпровідникових приладів, у тому числі процесорів для самих різних застосувань. На кінець 2009 р. корпорація Intel підрозділяє процесори на наступні категорії: [4]

1. Процесори для настільних ПК
   * + - Процесори сімейства Intel Core
       - Процесори сімейства Intel Pentium
       - Процесори сімейства Intel Celeron
2. Процесори для серверів і робочих станцій

* Процесори Intel для серверів
* Процесори Intel для робочих станцій

1. Процесори для інтернет-пристроїв

* Процесор Intel Atom для нетбуків і неттопів
* Процесори Intel Atom для мобільних інтернет-пристроїв

1. Процесори для ноутбуків

* Процесори сімейства Intel Core
* Процесори сімейства Intel Celeron

Для мобільних ПК призначені процесори з різними технологіями енергозбереження і бездротових рішень. Для систем, які націлені на використання в якості серверів і робочих станцій, пропонуються процесори, які, як правило, не використовуються в настільних комп'ютерах Intel Itanium і Intel Xeon

### **1.1.1 Процесори Intel Core i7**

У 2008 році корпорація Intel запропонувала нову архітектуру процесора лінійки х86-64 під умовною назвою Nehalem (саме ядро ​​нових процесорів носить умовну назву Bloomfield). Це майже революція для процесорів Intel, тому що її розробники довго критикували AMD якраз за основні принципи побудови процесорів AMD64, але які тепер стали впроваджуватися в процесорах Intel Core i7 (раніше ці технології використовувалися на процесорах лінійки Xeon).

В першу чергу, це вбудований контроллер пам'яті, який дозволяє прискорити роботу з модулями пам'яті, що дозволяє ліквідувати затримки, внесені чіпсетом. Зокрема, в цьому випадку на частку північного моста залишається тільки функція роботи з відеопідсистемою і зв'язок з південним мостом, тобто різко зменшується складність чіпсета і істотно знижується енергоспоживання. Також в даній архітектурі з’явилася можливість працювати з трьохканальной пам'яттю, коли використовуються три модулі пам'яті

Друга революційна подія для процесорів Intel - це відмова від традиційної паралельної процесорної шини даних, яка суттєво обмежувала швидкодію процесорів, особливо при багатоядерній архітектурі. Як і в процесорах AMD, була впроваджена нова шина QPI (Quick Path Interconnects) з топологією точка-точка для обсягу з периферійними пристроями і процесорними ядрами (процесорами).

В чотирьох ядерні процесори Intel сімейства Core i7 були введені нові технології: [4]

* Технологія Intel Turbo Boost максимально підвищує продуктивність ресурсномістких програм, динамічно збільшуючи продуктивність відповідно до навантаження - продуктивність вище там, де вимагаєся. Тобто в залежності від навантаження на конкретне ядро, його тактова частота може підвищуватися або знижуватися на 1-2 ступені (ступінь - це коефіцієнт множення базової тактової частоти). В останніх моделях процесорів коефіцієнт зміни частоти складає вже 1-5 ступенів, що говорить про серйозну виконану роботу по вдосконаленню даної технології [5].
* Технологія Intel Hyper-Threading дозволяє багато поточним програмам виконувати більше завдань паралельно. Всі процесори Intel Core І7 підтримують дану технологію, що дозволяє використовувати до 8 логічних процесорів для чотирьох ядерних моделей.
* Технологія Intel Smart Cache забезпечує високу продуктивність і ефективність кеш-пам'яті, які оптимізовані для найсучасніших багатопоточних ігор.

У верхньому сегменті десктопних CPU лідером є Core i7-975.Чотири ядра, можливість обробки двох обчислювальних потоків одночасно, висока тактова частота, прискорювач у вигляді технології Turbo Boost і краща архітектура роблять його найшвидшим на ринку. Модель i7-920 навряд чи буде настільки популярною, оскільки позбавлена ​​вільного множника і має меншу робочу частоту[6]

Таблиця 1.1. Порівняльна характеристика специфікацій Core i7-975 і Core i7-920 [7,8]

|  |  |  |
| --- | --- | --- |
| Специфікація | Intel Core i7-975 EE | Intel Core i7-920 |
| К-сть ядер | 4 | 4 |
| К-сть потоків | 8 | 8 |
| Тактова частота | 3.33 GHz | 2.66 GHz |
| Максимальна тактова частота з технологією Turbo Boost | 3.6 GHz | 2.93GHz |
| Технология Intel Smart Cache | 8Mb | 8Mb |
| Intel QPI Speed | 6.4 GT/s | 4.8 GT/s |
| Множник | 25х | 20х |
| Літографія | 45 nm | 45nm |

### **1.1.2 Процесори Intel Core i5**

В процесорах Intel Core i5 використовується мікроархітектура Intel Core - динамічномасштабована мікроархітектура Intel, і вони мають такі ж енергоефективні ядра, як і процесори Intel Core І7, але при цьому призначені для комп'ютерів масової категорії.

Основна особливість нових процесорів: інтегрований контролер пам’яти, що підтримує 2 канали високошвидкісної пам'яті DDR3 з частотою 1333 МГц, тобто користувач серйозно економить на спрощення системної плати і підсистеми пам'яті. Використання вже випробуваної технології Intel Turbo Boost максимально підвищує продуктивність ресурсномістких програм, динамічно збільшуючи продуктивність відповідно до навантаження, підвищуючи або знижуючи коефіцієнт множення на 1-4 ступені. Так як на кристалі процесора інтегровано контролер шини PCI Express х16, то для роботи з цим процесором призначений чіпсет Intel Р55.

Процесори Intel Core І5 не підтримують технології Hyper-Threading і Vir-tualization technology for directed I/O (VT). [4, 9]

Таблиця 1.2 Порівняльна характеристика процесорів сімейства Core i5 [10]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Модель | Тактова частота | Turbo | Ядра | Кеш, L3 | Рівень енергозатрат | Технологія |
| Core i5-2300 | 2.8 ГГц | 1/2/2/3 | 4 | 6Мб | Звичайне | «Sandy Bridge» (32 нм) |
| Core i5-2500K | 3.3 ГГц | 1/2/3/4 | 4 | 6Мб | Низьке | «Sandy Bridge» (32 нм) |
| Core i5-2500T | 2.3 ГГц | 1/5/9/10 | 4 | 6Мб | Наднизьке | «Sandy Bridge» (32 нм) |

### **1.1.3 Сімейство процесорів Intel Core 2**

У сімейство Intel Core 2 входять двох-чотирьохядерні Intel Core 2 Extreme і чотирьохядерні Intel Core 2 Quad. Лінійка процесорів Intel Core 2 вважається восьмим поколінням процесорів Intel для архітектури х86. Але, фактично, це розвиток ідей мікроархітектури Intel Р6, яка істотно була покращена при розробці процесорів для ноутбуків, а також використання найбільш вдалих знахідок, отриманих при розробці Intel Pentium 4 з архітектурою NetBurst. Якщо згадати, то після появи процесорів сьомого покоління Intel Pentium 4 багато користувачів ще тривалий час продовжували використовувати процесори попереднього покоління - процесори Intel Pentium III. І це не дивно, тому що тільки при підвищенні тактової частоти приблизно вище 2 ГГц реальна продуктивність процесорів Pentium 4 стала вище процесорів попереднього покоління.

Але, звичайно, ядро ​​Conroe процесора Intel Core 2 не є прямим спадкоємцем процесорів шостого покоління, оскільки було надто багато модернінізацій. Основне рішення, яке прийшло від шостого покоління - це короткий конвеєр, що дозволяє при менших тактових частотах отримувати продуктивність вище, ніж у Pentium 4 з його наддовгим конвеєром (гіперконвейер - це термін корпорації Intel, який вона запропонувала пілсля виходу процесора Pentium 4 [4]).

В даний час сімейство Intel Core 2 представлено величезною різноманітністю моделей, відомих під найменуванням Intel Core 2 Quad (чотирьохядерні), плюс самі просунуті моделі процесора на поточний момент отримують маркування Intel Core 2 Extreme ( двох-або чотирьохядерні і з розблокованим множником).

Перші процесори Intel Core 2 офіційно представлені влітку 2006 р., вони були випущені за технологією 65 нм і мали ядро ​​з умовною назвою Conroe. Надалі при модернізації процесори отримали нові ядра з умовними назвами Merom (для мобільних персональних комп’ютерів) і Kentsfield (чотирьохядерний Conroe). У 2007 р. після освоєння технології 45 нм стали випускатися процесори з ядром під умовною назвою Penryn. Для серверів випускаються процесори сімейства Intel Core 2 з модернізованими ядрами, які носять свої умовні назви.

В даний час чотириядерні процесори лінійки Intel Core 2 займають левову частку ринку процесорів. Їх мікроархітектура у відповідності до планів розробників удосконалюється для отримання ще більш високих результатів продуктивності.

Як сказано на сайті корпорації Intel, у процесорах лінійки Intel Core 2 для досягнення рекордних результатів продуктивності і енергозбереження використовуються наступні технології. [11]

* Intel Wide Dynamic Execution - забезпечує виконання більшого числа команд за тактовий цикл, що дозволяє скоротити час виконання і поліпшити енергозбереження. Дана технологія (Macro-Fusion , Macro-OPs Fusion) дозволяє об'єднувати мікрокоманди процесора і виконувати їх за один такт, в ряді випадків можуть об'єднуватися команди x86. Іноді можливо паралельне виконання до 5 мікрооперацій за такт.
* Intel Intelligent Power Capability-розроблена для забезпечення енерго-економічності і продуктивності, а також для максимального збільшення часу автономної роботи ноутбука. Родзинка цієї технології-це включення потрібних блоків і шин, коли це необхідно. Таким чином, якщо потрібен, наприклад, блок ММХ, то система без затримок підключає до нього живлення і необхідні шини. Раніше, в старих моделях процесорів для економії вимикалися непотрібні блоки, але для нової активації потрібно час, що призводило до сповільненої реакції комп'ютера, яскравий приклад - це ноутбуки з процесором Pentium М.
* Intel Smart Memory Access - підвищує продуктивність системи, оптимізуючи використання доступної пропускної здатності пам'яті. У перекладі - інтелектуальний доступ до пам'яті, тобто забезпечення спекулятивної вибірки даних раніше, ніж вони будуть потрібні.
* Intel Advanced Smart Cache-забезпечує високу продуктивність і ефективність кеш-пам'яті. Це оптимізація для використання з багатоядерними процесорами. Дана технологія, в основному, спрямована на ліквідацію дублювання даних в кеш L2, а також динамічного перерозподілу ресурсів кеша L2. Плюс технологічні поліпшення щодо прискорення вибірки даних з кеша.
* Intel Advanced Digital Media Boost - прискорює обробку відео, мовлення та зображень, перетворення фотографій, шифрування, роботу фінансових, інженерних і наукових додатків. Зокрема, дозволяє виконувати 128-бітові інструкції SSE за один такт.

Таблиця 1.3 Порівняльна характеристика процесорів сімейства Core 2 [12]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Модель | Тактова частота | Кеш, L2 | FSB | Множник | Технологія |
| Core 2 Quad Q6400 | 2.13 GHz | 2 × 4 MB | 1066 MT/s | 8х | "Kentsfield" (65 nm) |
| Core 2 Quad Q6700 | 2.67 GHz | 2 × 4 MB | 1066 MT/s | 10х | "Kentsfield" (65 nm) |
| Core 2 Quad Q8200 | 2.33 GHz | 2 × 2 MB | 1333 MT/s | 7х | "Yorkfield-6M" (45 nm) |
| Core 2 Quad Q8300 | 2.5 GHz | 2 × 2 MB | 1333 MT/s | 7.5х | "Yorkfield-6M" (45 nm) |
| Core 2 Quad Q9705 | 3.17 GHz | 2 × 3 MB | 1333 MT/s | 9.5х | "Yorkfield-6M" (45 nm) |
| Core 2 Extreme QX6700 | 2.67 GHz | 2 × 4 MB | 1066 MT/s | 10х | "Kentsfield XE" (65 nm) |

## 1.2 Процесори корпорації AMD

Корпорація AMD заснована в 1969 р., конкурент корпорації Intel. Фактично на лінії фронту між AMD і Intel, у всякому разі сьогодні, і відбуваються найбільш важливі в комп'ютерному світі події. І результатом запеклої конкуренції є зниження цін на морально застарілі вироби, а також впровадження в життя фантастичних технологій. Але в той же час з появою кожного нового "типу" процесора доводиться знову витрачати великі гроші на покупку комплекту "процесор-материнська плата-пам'ять", так як модернізувати старий комп'ютер заміною одного компонента, як це було раніше, тепер вже не вдається. Треба відзначити, що при появі перших моделей процесорів Intel 486, Pentium, Pentium II і Pentium III корпорація AMD завжди трохи відставала від Intel, щоправда, потім досить швидко знаходила свою наздоганяла, уводячи в свої аналогічні процесори нові команди й цікаві особливості архітектури.

Найбільш важливий крок корпорацією AMD був зроблений, коли вийшли її процесори сьомого покоління, які мали початкову назву К7, але для більш успішного просування на ринку вони отримали торговельне найменування Athlon. При їх розробці було прийнято рішення відмовитися від випуску корпусів під сокети процесорів Intel, а розробити свій оригінальний варіант, що дозволяло не чекати, коли головний конкурент випустить нову модель процесорів. Спочатку процесори AMD випускалися під роз'єм типу Slot А, а потім його змінив Soket 462. В результаті, для всіх процесорів AMD, з цього моменту, потрібно свій тип чипсета і відповідно системної плати.

Спочатку поява 64-розрядних моделей процесорів під маркою Opteron і Athlon 64 викликало обережне ставлення з боку користувачів і розробників, так як були проблеми з сімейством 64 - розрядних процесорів Intel Itanium, які, незважаючи на великі надії користувачів настільних комп'ютерів, виявилися корисними лише для серверних систем. Але, на щастя, особливо після випуску двоядерних процесорів Athlon 64 Х2 і Athlon 64 FX, виявилося, що вони мають куди більш приємні споживчі характеристики, ніж "гарячі грубки" Intel Pentium 4 з ядром Prescott, зокрема, це стосується і комп'ютерних ігор. Результат позначився і на ринковій частці корпорації AMD, яка стала стрімко збільшуватися.

Фактично, 64-розрядне розширення архітектури від AMD стало стандартом де-факто, і вже розробники Intel вимушено стали орієнтуватися на нього. Аналогічна 64-розрядна технологія в процесорах Intel отримала назву ЕМ64Т.

Після виходу в кінці 2006 р. нових процесорів Intel Core 2, ситуація на комп'ютерному ринку знову стала не на користь корпорації AMD. Ще більш тривожний дзвінок пролунав для AMD, коли корпорація Intel випустила процесори з технологією 45 нм. З новими конкурентами старі версії процесорів AMD64 серйозно конкурувати вже не могли. І ринок знову хитнувся в бік корпорації Intel. Розробникам корпорації AMD довелося знову шукати адекватну відповідь на ініціативи Intel.

У листопаді 2007 р. корпорація AMD представила свої перші процесори Phenom, які відкрили "зоряну" лінійку процесорів (Stars, у багатьох джерелах називається архітектура К10), так як тепер планується процесорам привласнювати назви з зоряного каталогу. На жаль, проблеми з виправленням серйозної помилки в блоці кеш-пам'яті затримали початок використання цього процесора, і темп продажів був втрачений. До того ж, після виходу двох-і чотирьохядерних процесорів Intel Core 2 з оновленою мікроархітектури, які виявилися більш "спритними", процесори Phenom стали розглядатися тільки як дешева альтернатива дорогим Intel Core 2.

Поява в кінці 2008 р. процесорів Intel Core І7 взагалі не залишило шансів на перемогу по продуктивності процесорам Phenom, так як у конкурентів з'явився вбудований контроллер пам'яті, причому відразу DDR3, і шина, аналогічна HyperTransport, для обміну даними. Мабуть, це змусило корпорацію AMD поквапитися із запуском у виробництво нових процесорів Phenom І на початку 2009 р.

У 2009 р. корпорація AMD серйозно порадувала користувачів випуском спочатку процесорів трьох-і чотирьохядерних Phenom II. Наступний крок, який виявився до душі багатьом, хто вміє рахувати гроші, - це випуск на ринок бюджетних варіантів двоядерних Phenom І, які являють собою отбраковку чотирьохядерних моделей (серйозна економія грошей для домашніх комп'ютерів). Крім того, випущені двоядерні процесори Athlon II, які являють собою той же Phenom II, але без кеша L3 і спочатку з двома ядрами (і розпочато випуск моделей з трьома і чотирма ядрами). По продуктивності нові процесори стоять на одній лінії з аналогічною продукцією Intel, але значно дешевше, так що повторюється ситуація кількарічної давнини, коли процесори AMD були переважніше дорогих конкурентів.

На кінець 2009 р. корпорація AMD випускає процесори для найрізноманітніших застосувань. Найбільший обсяг продукції (процесори загального застосування) припадає на наступні категорії: AMD Phenom І, AMD Phenom ХЗ і Х4, AMD Athlon І, AMD Athlon Х2, AMD Athlon, AMD Sempron, а для мобільного застосування AMD Turion X2 і AMD Sempron різних модифікацій. Для серверів випускається велика номенклатура процесорів AMD Opteron (в тому числі і шестиядерні).

### **1.2.1 Технологии корпорації AMD**

Користувачі звикли, що самі передові технології пропонує корпорації Intel, а вже інші фірми слухняно йдуть зазначеним курсом. На практиці так і виходило, поки корпорація AMD не стала впроваджувати власні технології для процесорів, які не повторюють сліпо архітектуру процесорів Pentium і Core. Далі дані короткі описи ряду ключових технологій, які використовуються в процесорах корпорації AMD.

* Технологія HyperTransport - одна з ключових технологій сучасних процесорів корпорації AMD. Основна ідея цієї технології - скорочення вузьких місць у підсистемі вводу-виводу, що завжди було в комп'ютерах IBM PC з самого народження. Корпорація AMD запропонувала замість паралельної шини даних використовувати послідовну шину типу "точка-точка", правда, тільки для зв'язку процесора і північного мосту. Використання технології HyperTransport дозволяє збільшити загальну продуктивність системи, підвищити швидкість обробки даних і зменшити час відгуку. Плюс інтегрований контролер пам'яті DDR прискорює доступ до пам'яті завдяки прямому з'єднанню пам'яті і процесора. Відповідно, маркування модулів пам'яті по FSB, як це було зроблено для процесорів Intel, не зовсім справедлива для процесорів AMD [13].
* Технологія 3DNow! Professional - це інструмент для роботи з тривимірними мультимедійними об'єктами, зі звуковими даними, відеоматеріалами та цифровими фотографіями [14].
* Технологія AMD64 не тільки дозволяє використовувати 64-розрядні обчислення, але й гарантує сумісність з існуючими програмними засобами. Як кажуть розробники, це забезпечує безперешкодний перехід до майбутніх 64-розрядним програмам. Тобто на одній і тій же платформі, що використовує технологію AMD64, можуть одночасно і без будь-яких обмежень виконуватися як 32-розрядні, так і 64-розрядні додатки [15].
* Для захисту від вірусних атак і помилок програмного забезпечення в нових процесорах корпорації AMD введений апаратний механізм антивірусного захисту EVP (Enhanced Virus Protection). Принцип роботи цього механізму заснований на контролі помилки переповнення буфера. Але слід пам'ятати, механізм захисту від вірусів працює тільки тоді, коли цю функцію підтримує операційна система. Зокрема, апаратну захист від переповнювання буфера підтримують операційні системи, починаючи з Windows ХР SP2 (зауважимо, користувачеві самому необхідно включити захист додатків і відповідних їм файлів)[16].
* Технологія AMD PowerNow! збільшує час автономної роботи для ноутбуків, дозволяє знизити енергоспоживання процесора і зменшити тепловиділення в стандартних робочих умовах.[17]

### **1.2.2 64-розрядні процесори AMD**

У процесорах AMD поколінь К8 і К10 використовується архітектура AMD64, яка раніше була відома як "Hammer" або "х86-64". Родзинка цього покоління процесорів - 64-розрядна обробка даних (всі процесори x86, починаючи з i386, є 32-розрядними). Розробниками процесорів гарантується, що все старе програмне забезпечення, призначене для 32-розрядних процесорів, може працювати і на нових процесорах. Причому одночасно можуть виконуватися як 32-розрядні програми, так і 64-розрядні.

При роботі на комп'ютері з 64-розрядним процесором не слід думати, що оскільки подвоєна розрядність процесора, на старому програмному забезпеченні буде досягнута дворазова продуктивність. Для отримання виграшу при використанні 64-розрядних процесорів необхідна нова операційна система і нове прикладне програмне забезпечення. В іншому випадку продуктивність навіть знижується, особливо це стосується 64-розрядного процесора Intel Itanium, якому доводиться при роботі зі старими програмами компілювати 32-розрядний код в свої мікрокоманди, що він робить не дуже швидко. Для Athlon 64 продуктивність може знижуватися, як зазначено на сайті корпорації AMD, для завдань, пов'язаних з шифруванням даних, а в інших випадках досягається цілком пристойне збільшення можливостей.

У процесорах AMD 64 використовується технологія HyperTransport, яка дозволяє збільшити загальну продуктивність системи за рахунок скорочення вузьких місць у підсистемі вводу-виводу, що дозволяє підвищити швидкість обробки даних і зменшити час відгуку. Крім того, технологія HyperTransport дозволяє просто створювати багатопроцесорні системи, а також об'єднувати на одному кристалі майже будь-яку кількість процесорних ядер, що використовується в процесорах AMD Opteron.

Новинкою став інтегрований контролер пам'яті DDR, а в подальшому DDR2 і DDR3, що прискорює доступ до пам'яті, так як для цього тепер не потрібні проміжні елементи (чіпсет, який вносить затримку як мінімум в такт). Такий спосіб роботи з пам'яттю дозволяє забезпечити більш швидке завантаження програм, поліпшену підтримку багатозадачності і ефективність виконання додатків.

### **1.2.3 Процесори AMD Phenom II і Phenom**

В кінці листопада 2007 р. корпорація AMD представила дві моделі чотирьох ядерних процесорів нового сімейства AMD Phenom Quad-Core, які позиціонуються як сімейство К10 процесорів х86. Ці процесори за планами повинні були замінити процесори, що раніше випускалися для настільних і мобільних комп'ютерів , а також скласти реальну конкуренцію новим процесорам Intel. До того ж процесори Phenom повинні стати основою для нової платформи для настільних персональних комп'ютер під кодовою назвою AMD Spider.

У перших партіях процесорів AMD Phenom, а також AMD Opteron з ядром Barcelona, ​​виявилася помилка в блоці TLB (Translation Lookaside Buffer) кеш-пам'яті третього рівня. Це призвело до затримок продажів, а всі подальші процесори AMD Phenom, щоб уникнути плутанини, отримали номери, що закінчуються на цифри "9x50".

Для ядра перших моделей AMD Phenom використовується умовне найменування Agena, і вони виконані за технологією 65 нм. Моделі з помилкою TLB мають степпіг В2, а з виправленою помилкою ВЗ..

У 2009 р. почався продаж процесорів AMD Phenom II, виконаних по 45 нм-технології та умовною назвою ядра Deneb. Всі моделі процесорів AMD Phenom II позиціонуються як трьох-і чотирьохядерні, але в роздрібному продажі можна зустріти версії з 2 ядрами, яких не так багато, тому що являють собою браковані чотирьохядерні моделі.

У процесорах AMD Phenon і Phenon II використано багато цікавих новинок, зокрема, запроваджено нові SSE-інструкції, що підвищують продуктивність в мультимедійних застосуваннях, модернізований блок пророкування розгалужень і т. д. Також в процесорах AMD Phenom, крім традиційних для лінійки AMD64 кеша L1 і кеша L2, для підвищення продуктивності введений загальний кеш третього рівня (L3), який дає суттєвий виграш в продуктивності для важких програм. [4]

Таблиця 2.1 Порівняльна характеристика процесорів сімейства AMD Phenom [18]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Модель | Тактова частота | Кеш, L2 | Кеш, L3 | Множник | Технологія |
| Phenom X4 9100e | 1.8 Гц | 4x 512 Кб | 2 Мб | 9х | «Agena», 65 нм |
| Phenom X4 9650 | 2,3 Гц | 4x 512 Кб | 2 Мб | 11,5х | «Agena», 65 нм |
| Phenom X4 9750B | 2,4 Гц | 4x 512 Кб | 2 Мб | 12х | «Agena», 65 нм |
| Phenom II X4 805 | 2,5 Гц | 4x 512 Кб | 4 Мб | 12,5х | «Deneb», 45 нм |
| Phenom II X4 945 | 3,0 Гц | 4x 512 Кб | 6 Мб | 15х | «Deneb», 45 нм |
| Phenom II X4 B95 | 3,0 Гц | 4x 512 Кб | 6 Мб | 15х | «Deneb», 45 нм |

### **1.2.4 AMD Quad FX**

Платформа AMD Quad FX (також відома як AMD 4x4) - перша ігрова платформа Advanced Micro Devices, розроблена спільно з NVIDIA як відповідь на Intel Core 2 Quad і складається з двох центральних процесорів Athlon 64 FX у виконанні Socket F, чіпсета nForce 680a і відеокарт GeForce 8. У 2006 році вийшли чотириядерні процесори Intel, випущені по 65 нм технологічному процесу. AMD використовувала менш "просунутий" 90 нм техпроцес, тому не змогла представити аналога. Замість цього каліфорнійська компанія представила ігрову систему, що складається з двох двоядерних процесорів Athlon. Єдиним представником цієї лінійки є Asus L1N64, яка випускалася в двох варіантах - перша (з префіксом WS) використовувала пару Athlon FX і трохи пізніше двохядерні Opteron Santa Rosa, друга (з префіксом WS / B) - підтримувала нові на той момент чотирьохядерні Opteron Barcelona. По суті це одна і та ж плата, яка відрізнялася тільки прошивкою BIOS

Таблиця 2.2 Модельний ряд процесорів AMD FX і порівняння характеристик[19]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер моделі | Частота | Кеш, L3 | Кеш, L2 | Технологія CMOS |
| FX 8350 | 4.0/4.2 ГГц | 8МБ | 8МБ | 32нм SOI |
| FX 8320 | 3.5/4.0 ГГц | 8МБ | 8МБ | 32нм SOI |
| FX 8150 | 3.6/4.2 ГГц | 8МБ | 8МБ | 32нм SOI |
| FX 8120 | 3.1/4.0 ГГц | 8МБ | 8МБ | 32нм SOI |
| FX 8100 | 3.1/3.7 ГГц | 8МБ | 8МБ | 32нм SOI |
| FX 6300 | 3.5/4.1 ГГц | 6МБ | 8МБ | 32нм SOI |
| FX 6200 | 3.8/4.1 ГГц | 6МБ | 8МБ | 32нм SOI |
| FX 6100 | 3.3/3.9 ГГц | 6МБ | 8МБ | 32нм SOI |
| FX 4300 | 3.8/4.0 ГГц | 4МБ | 4МБ | 32нм SOI |
| FX 4170 | 4.2/4.3 ГГц | 4МБ | 8МБ | 32нм SOI |
| FX 4130 | 3.8/3.9 ГГц | 4МБ | 4МБ | 32нм SOI |
| FX 4100 | 3.6/3.8 ГГц | 4МБ | 8МБ | 32нм SOI |

Таблиця 2.3 Технологія AMD64 [19]

|  |  |
| --- | --- |
| Технологія AMD64 | Характеристики |
| Одночасні 32- і 64-бітні обчислення | Да |
| Кеш-пам'ять 1-го рівня (інструкції + дані) для кожного ядра | 128 КБ (64 КБ + 64 КБ) |
| Кеш-пам'ять 2-го рівня (1 МБ для кожного ядра) | 8 МБ або 6 МБ або4 МБ |
| Кеш-пам'ять 3 рівня | 8 МБ (загальна кеш-пам'ять 3 рівня) |

## 1.3 Тестування процесорів

Найкращим результатом порівняння характеристик процесора – є їх робота в однакових умовах. Один із способів дізнатися, який процесор кращий – подивитися які результати дає в іграх. Якщо порівнювати Intel Core i5, Intel Core i3, AMD Phenom II i AMD FX, то опираючись на статтю «Тестирование процессоров AMD FX-6200 BE, FX-4170 BE, FX-4130 BE и Phenom II X4 960T BE в играх»[20] можна прийти до висновків, що прикрим в процесорах AMD, заснованих на архітектурі Zambezi, є той факт, що, незважаючи на підвищені тактові частоти, вони не змогли випередити своїх попередників – Intel Core i5

У номінальному режимі роботи процесор Phenom II X4 960Т BE був найповільнішим серед лінійки Phenom II, але при цьому випереджав FX-6100 BE і FX-4100 BE. Після розгону він розділив третє місце серед ЦП AMD з FX-6200 BE і Phenom II X4 965 BE.

У штатному режимі процесор FX-6200 BE посів третє місце серед родичів, а після розгону не розгубив свої позиції. Однак він був конкурентоспроможним порівняно зі старшими побратимами і чотирьохядерними CPU Intel.

Процесори FX-4170 BE і FX-4130 BE завдяки високим номінальним частотам зайняли сильні позиції, однак після розгону все встало на свої місця: вони розташувалися між FX-6100 BE і FX-4100 BE.

Після розгону весь модельний ряд AMD випередив Core i3-2120.

За рівнем енергоспоживання і тим більше по співвідношенню «енергоспоживання / продуктивність» процесори Intel залишилися поза конкуренцією.

Якщо дивитися на еволюцію чотирьох ядерних процесорів Intel Bloomfield - Lynnfield - Sandy Bridge - Ivy Bridge в іграх [21], то значний стрибок продуктивності відбувся тільки при переході з Core 2 Quad Q9500 на Core i5-760, Core i7-860 і Core i7-930. При переході з останніх ЦП на Core i7-2600К і Core i5-2500К продуктивність також виросла на помітну величину. Однак тут не варто забувати про різницю в тактових частотах процесорів, яка склала 600 - 700 МГц. Саме з цим моментом пов'язаний зростання результатів, а аж ніяк не зі змінами мікроархітектури CPU. Безперечним лідером цієї еволюції виявився вже знятий з виробництва Core i5-760, слідом за ним розташувалися більш «свіжі» моделі Core i5-2500К і Core i5-3570K.

Крім зростання продуктивності при зміні поколінь виразно спостерігалася збільшена енергоефективність процесорів. Найекономічнішими стали моделі 22 нм - Core i7-3770K і Core i5-3570K, що показали як низьке енергоспоживання, так і високу продуктивність.

## 1.4 Висновки до розділу 1

З вищесказаного видно, що в останні роки виробники процесорів не прагнуть до досягнення максимальної тактової частоти - замість цього вони нарощують міць CPU, збільшуючи кількість ядер.

У 2009 році в продуктових лінійках двох конкурентів відбулися істотні зміни. На зміну застарілому сімейства Intel Core 2 Duo прийшли нові чотирьох ядерні процесори Intel серій Core i3, i5 і i7. Вони обзавелися мікроархітектурою Sandy Bridge і виробляються за 32-нанометровим техпроцесом. Тим часом AMD суттєво доопрацювала свій чотирьохядерний Phenom X4, збільшивши об'єм кеш-пам'яті і освоївши 45-нанометровий технологічний процес. Одним з найважливіших нововведень є модульний принцип розташування ядер у системі х86 - по два на кожному модулі. Завдяки цій особливості компанії нескладно вибудувати модельний ряд, пропонуючи рішення з різними кількістю обчислювальних блоків і тактовими частотами. У світлі своїх останніх творінь компанія AMD налаштована на серйозне протистояння з процесорами Intel, а остання, в свою чергу, не планує поступатися місцем.

# РОЗДІЛ 2. Розробка програми ПРГ1 для ПКС СП

## 2.1 Аналіз задачі на внутрішній паралелізм з використанням концепції необмеженого паралелізму.

Основні принципи концепції необмеженого паралелізму:

1. Кількість процесорів – необмежена ();
2. Час виконання операції однаковий і дорівнює 1.
3. Не враховується час передачі даних.

Виходячи із даної концепції побудуємо ярусно-паралельну форму алгоритму обчислення заданого математичного виразу.

Замінивши на еквіваленті вирази матричні операції, перепишемо задане завдання A= (B \* C) \* Z + R \* (MO \* MT) наступним чином:

ki = (b1,1\* c1,1,b1,2\* c1,2,…,bN, N-1\* cN, N-1, bN,N\* cN,N)

ai = (k1,1\* z1,1,k,2\* z1,2,…,kN, N-1\* zN, N-, kN,N\* zN,N) + (r1\*(o1,1\*t1,i+ … + o1,N\*tN,i)+… rN\* (oN,1\*t1,i+ … + oN,N\*tN,i))

де i=1..N

З виразу видно, що елементи вектора ai шукаються в два етапи.

Ярусно-паралельна форма ai зображена на рис. 2.1.

ЯПФ знаходження зображена на рис. 2.1.

Знайдемо параметри цієї ЯПФ:

1. Висота ЯПФ:

.

1. Ширина ЯПФ:

.

1. Загальна кількість вузлів (операцій):

.

Оскільки для знаходження результату необхідно виконати розглянутий вище алгоритм N раз, то як кінцевий результат отримуємо:

1. Необхідна кількість процесорів .
2. Час обчислення завдання на процесорах .
3. Час обчислення завдання на 1 процесорі

.

1. Коефіцієнт прискорення .
2. Коефіцієнт ефективності .

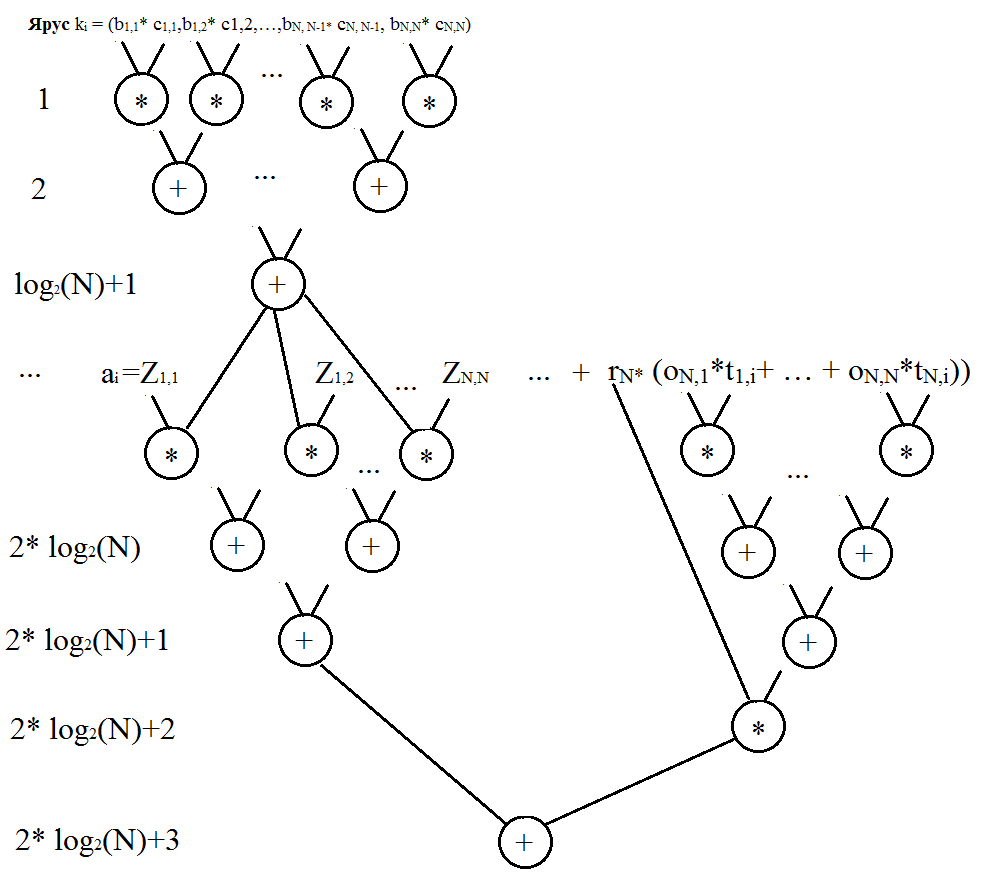


Рис. 2.1. Ярусно-паралельна форма алгоритму обчислення .

## 2.2 Розробка паралельного математичного алгоритму

Розрахунок даного матричного рівняння можливо провести в такі етапи:

ai = BH \* CH

ai =a + ai

Спільні ресурси: а

AH = а \*ZH + R (MO \* MTH)

де H = N / P, N – розмірність матриць, P – кількість процесорів.

Спільні ресурси: R, MO .

## 2.3 Розробка алгоритмів процесів

Таблиця 2.1. Алгоритми потоків для системи зі спільною пам’яттю

|  |  |  |
| --- | --- | --- |
| *№* | *Алгоритм процесу №1* | *ТС/КУ* |
| 1. | Введення B,Z |  |
| 2. | Сигнал всім задачам про введення | S2-1, S3-1, S4-1, S5-1, S6-1 |
| 3. | Очікування вводу в T2 і T6 | W2-1,W6-1 |
| 4. | Обрахування *a1* = BH\*CH |  |
| 5. | Обрахування *a* = *a* + *a1* |  |
| 6. | Сигнал про закінчення обрахунку *а* | S2-2, S3-2, S4-2, S5-2, S6-2 |
| 7. | Ждати закінчення обрахунку *a* в інших процесах | W2-2, W3-2, W4-2, W5-2, W6-2 |
| 8. | Копіювання a1 := a, R1 := R, MO1 := MO | КУ |
| 9. | Обрахування AH = а1 \*ZH + R1 (MO1\* MTH) |  |
| 10. | Очікування обрахування AH | W2-3, W3-3, W4-3, W5-3, W6-3 |
| 11. | Вивід А |  |

|  |  |  |
| --- | --- | --- |
| *№* | *Алгоритм процесу №2* | *ТС/КУ* |
| 1. | Введення C, R, MO |  |
| 2. | Сигнал всім задачам про введення | S1-1, S3-1, S4-1, S5-1, S6-1 |
| 3. | Очікування вводу в T1 і T6 | W1-1, W6-1 |
| 4. | Обрахування *a2* = BH\*CH |  |
| 5. | Обрахування *a* = *a* + *a2* |  |
| 6. | Сигнал про закінчення обрахунку *а* | S1-2, S3-2, S4-2, S5-2, S6-2 |
| 7. | Ждати закінчення обрахунку *a* в інших процесах | W1-2, W3-2, W4-2, W5-2, W6-2 |
| 8. | Копіювання a2 := a, R2 := R, MO2 := MO | КУ |
| 9. | Обрахування AH = а2 \*ZH + R2 (MO2\* MTH) |  |
| 10. | Сигнал про закінчення обрахування AH | S1-3 |

|  |  |  |
| --- | --- | --- |
| *№* | *Алгоритм процесу №6* | *ТС/КУ* |
| 1. | Введення MT |  |
| 2. | Сигнал всім задачам про введення | S1-1, S2-1, S3-1, S4-1, S5-1 |
| 3. | Очікування вводу в T1 і T2 | W1-1, W2-1 |
| 4. | Обрахування *a6* = BH\*CH |  |
| 5. | Обрахування *a* = *a* + *a6* |  |
| 6. | Сигнал про закінчення обрахунку *а* | S1-2, S2-2, S3-2, S4-2, S5-2 |
| 7. | Ждати закінчення обрахунку *a* в інших процесах | W1-2, W2-2, W3-2, W4-2, W5-2 |
| 8. | Копіювання a6 := a, R6 := R, MO6 := MO | КУ |
| 9. | Обрахування AH = а6 \*ZH + R6 (MO6\* MTH) |  |
| 10. | Сигнал про закінчення обрахування AH | S1-3 |

|  |  |  |
| --- | --- | --- |
| *№* | *Алгоритм процесів №3,4,5* | *ТС/КУ* |
| 1. | Ждати введення даних | W1-1, W2-1, W6-1 |
| 2. | Обрахування *a3* = BH\*CH |  |
| 3. | Обрахування *a* = *a* + *a3* |  |
| 4. | Сигнал про закінчення обрахунку *а* | S1-2, S2-2, S3-2, S4-2, S5-2, S6-2 |
| 5. | Ждати закінчення обрахунку *a* в інших процесах | W1-2, W2-2, W3-2, W4-2, W5-2, W6-2 |
| 6. | Копіювання ai := a, Ri := R, MOi := MO | КУ |
| 7. | Обрахування AH = а3 \*ZH + R3 (MO3\* MTH) |  |
| 8. | Сигнал про закінчення обрахування AH | S1-3 |

## 2.4 Розробка схеми взаємодії процесів

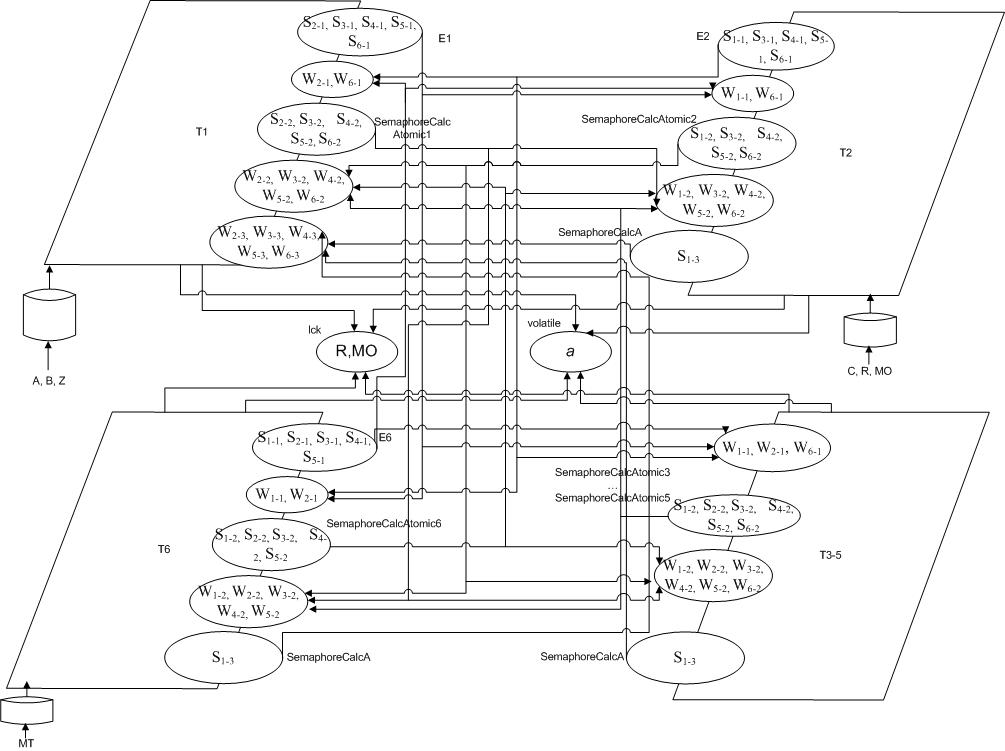


Рис. 2.2. Схема взаємодії задач для системи зі спільною пам’яттю

## 2.5 Розробка програми ПРГ1

Для вирішення проблеми синхронізації при вводі даних використовується засіб паралельного програмування мови C# - події. E1, E2, E6 – відповідають за введення даних в задачах T1, T2 і T6.

Для реалізації синхронізації при обрахунку *a* використовуються семафори.

Для копіювання спільного ресурсу використовуються механізм замків в мові C#, що реалізований за допомогою оператора lock. Тільки один потік може мати доступ до об’єкту, що синхронізований за допомогою даного оператора.

Синхронізація по виводу реалізована за допомогою семафорів.

## 2.6 Тестування програми ПРГ1

Тестування проводилося на системі з характеристиками:

Центральний процесор:

Оперативна пам’ять:

Операційна система:

Версії програмного забезпечення:

Для замірів часу використовувались функції зі стандартної бібліотеки мови -

DateTime

Таблиця 2.2. Час виконання програм обчислення на системі зі СП

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N \ P | 1 | 2 | 3 | 4 | 5 | 6 |
| 600 | 16,4268 | 8,5332 | 5,6784 | 4,1184 | 3,8376 | 3,26 |
| 1200 | 130,8374 | 65,876 | 45,2012 | 35,9364 | 31,4776 | 24,923 |
| 1800 | 437,5339 | 219,9231 | 149,455 | 111,5873 | 98,5405 | 82,2464 |

Коефіцієнт прискорення обраховується за формулою [22]:



Таблиця 2.3. Значення коефіцієнтів прискорення для системи з СП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N / P | 2 | 3 | 4 | 5 | 6 |
| 600 | 1,925046 | 2,892857 | 3,988636 | 4,280488 | 5,038896 |
| 1200 | 1,986116 | 2,894556 | 3,640804 | 4,156524 | 5,249665 |
| 1800 | 1,989486 | 2,927529 | 3,921001 | 4,440143 | 5,319794 |

Коефіцієнт ефективності обраховується за формулою [22]:



Таблиця 2.4. Значення коефіцієнтів ефективності для системи з СП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N / P | 2 | 3 | 4 | 5 | 6 |
| 600 | 96,25229 | 96,42857 | 99,71591 | 85,60976 | 83,9816 |
| 1200 | 99,30582 | 96,4852 | 91,02011 | 83,13048 | 87,49442 |
| 1800 | 99,47429 | 97,58431 | 98,02502 | 88,80286 | 88,66323 |

Рис. 2.3. Залежність коефіцієнту прискорення від кількості ЦП для систем зі спільною пам’яттю

Рис. 2.4. Залежність коефіцієнту прискорення для систем зі спільною пам’яттю при N=600

Рис. 2.5. Залежність коефіцієнту прискорення для систем зі спільною пам’яттю при N=1200

Рис. 2.6. Залежність коефіцієнту прискорення для систем зі спільною пам’яттю при N=1800

Рис. 2.7. Залежність коефіцієнту ефективності від кількості ЦП для систем зі спільною пам’яттю

Рис. 2.8. Залежність коефіцієнту ефективності для систем зі спільною пам’яттю при N=600

Рис. 2.9. Залежність коефіцієнту ефективності для систем зі спільною пам’яттю при N=1200

Рис. 2.10. Залежність коефіцієнту ефективності для систем зі спільною пам’яттю при N=1800

## 2.6 Висновки до розділу 2

1. Реалізація взаємодії процесів в програмі для ПКС зі спільною пам'яттю виконувалась за допомогою засобів мови програмування C#. Використання семафорів для синхронізації і замків для критичних секцій, що дозволило ефективно побудувати програму.

2. Тестування програми для ПКС зі спільною пам'яттю показало, що коефіцієнт прискорення лежить в діапазоні від 1.92 до 5,31; максимальний коефіцієнт прискорення отриманий при P = 6, N = 1800; коефіцієнт ефективності лежить в діапазоні від 83.13% до 99.7%; мінімальний коефіцієнт ефективності отриманий при P =5, N = 1200.

# РОЗДІЛ 3. Розробка програми ПРГ1 для ПКС з ЛП

## 3.1 Розробка алгоритмів процесів

Таблиця 3.1. Алгоритми потоків для системи зі спільною пам’яттю

|  |  |
| --- | --- |
| *№* | *Алгоритм процесу №1* |
| 1. | Прийняти Z2h, B2h від T3 |
| 2. | Надіслати Bh, Zh в T2 |
| 3. | Прийняти Ch, R, MO від Т2 |
| 4. | Обчислення a1 = BH \* CH |
| 5. | Надіслати a1 в Т2 |
| 6. | Прийняти MTh, а від Т2 |
| 7. | Обчислення AH = а \*ZH + R (MO\* MTH) |
| 8. | Прийняти Ah від Т2 |
| 9. | Передати A2h в Т3 |

|  |  |
| --- | --- |
| *№* | *Алгоритм процесу №2* |
| 1. | Прийняти Zh, Bh від T1 |
| 2. | Прийняти C2h, R, MO від Т4 |
| 3. | Передати Ch,R,MO в Т1 |
| 4. | Обчислення a2 = BH \* CH |
| 5. | Прийняти a1 від Т1 |
| 6. | Обчислення a2 = a2 + a1 |
| 7. | Надіслати a2 в Т4 |
| 8. | Прийняти MT2h, а від Т4 |
| 9. | Передати MTh, a в Т1 |
| 10. | Обчислення AH = а \*ZH + R (MO\* MTH) |
| 11. | Передати Ah в Т3 |

|  |  |
| --- | --- |
| *№* | *Алгоритм процесу №3* |
| 1. | Введення B,Z |
| 2. | Передати Zh, Bh в T1 |
| 3. | Передати Z3h, B3h в T4 |
| 4. | Прийняти Ch, R, MO від Т4 |
| 5. | Обчислення a3 = BH \* CH |
| 6. | Надіслати a3 в Т4 |
| 7. | Прийняти MTh, а від Т4 |
| 8. | Обчислення AH = а \*ZH + R (MO\* MTH) |
| 9. | Прийняти А2h від Т1 |
| 10. | Прийняти A3h від Т4 |
| 11. | Вивід А |

|  |  |
| --- | --- |
| *№* | *Алгоритм процесу №4* |
| 1. | Введення MT |
| 2. | Прийняти C4h, R, MO від Т5 |
| 3. | Передати Ch, R, MO в Т3 |
| 4. | Передати C2h, R, MO в Т2 |
| 5. | Прийняти Z3h,B3h від Т3 |
| 6. | Передати Z2h, B2h в Т5 |
| 7. | Прийняти a2 від Т2 |
| 8. | Прийняти a3 від Т3 |
| 9. | Прийняти a5 від Т5 |
| 10. | Обчислення a4 = BH \* CH |
| 11. | Обчислення а = a2 + a3 + a5 |
| 12. | Передати MTh,a в Т3 |
| 13. | Передати MT2h, a в Т5 |
| 14. | Передати MT2h, a в Т2 |
| 15. | Прийняти A2h від Т5 |
| 16. | Обчислення AH = а \*ZH + R (MO\* MTH) |
| 17. | Передати A3h в Т3 |

|  |  |
| --- | --- |
| *№* | *Алгоритм процесу №5* |
| 1. | Прийняти C5h, R, MO від Т6 |
| 2. | Передати C4h, R, MO в Т5 |
| 3. | Прийняти B2h, Z2h від Т4 |
| 4. | Передати Bh, Zh в Т6 |
| 5. | Прийняти a6 від Т6 |
| 6. | Обчислення a5 = BH \* CH |
| 7. | Обчислення a5 = a5 + a6 |
| 8. | Передати a5 в Т4 |
| 9. | Прийняти MT2h, a від Т4 |
| 10. | Передати MTh, a в Т6 |
| 11. | Прийняти Ah від Т6 |
| 12. | Обчислення AH = а \*ZH + R (MO\* MTH) |
| 13. | Передати A2h в Т4 |

|  |  |
| --- | --- |
| *№* | *Алгоритм процесу №6* |
| 1. | Введення C, R, MO |
| 2. | Передати С5h, R, MO в Т5 |
| 3. | Прийняти Bh,Zh від Т5 |
| 4. | Обчислення a6 = BH \* CH |
| 5. | Передати a6 в Т5 |
| 6. | Прийняти MTh, a від Т5 |
| 7. | Обчислення AH = а \*ZH + R (MO\* MTH) |
| 8. | Передати Ah в Т5 |

## Розробка схеми взаємодії задач

#### Схема взаємодії задач для системи з локальною представлена на рисунку 3.1

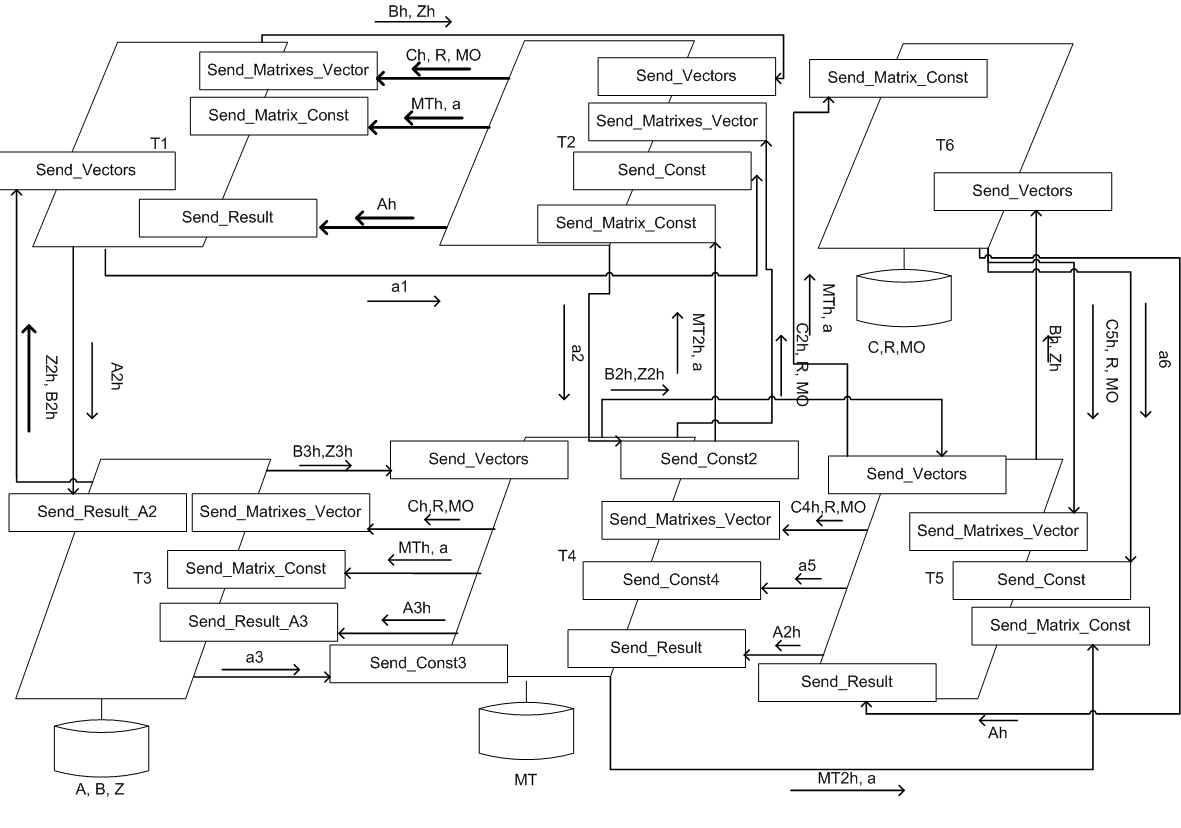


Рисунок 3.1. Схема взаємодії задач для системи з локальною пам’яттю

## 3.3. Розробка програми

Програмне забезпечення для системи з локальною пам’яттю було реалізовано за допомогою механізму рандеву в мові програмування Ада. Даний механізм призначений для прямого впливу двох задач, а інша приймає даний виклик. При цьому задачі синхронізуються, обмінюються даними і потім продовжують своє виконання [22].

Задача Т1 має такі оператори входу:

Send\_Matrixes\_Vector – приймає Ch, R, MO від Т2

Send\_Matrix\_Const – приймає MTh, a від Т2

Send\_Vectors – приймає Z2h, B2h від Т3

Send\_Result – приймає Ah від Т2

Задача Т2 має такі оператори входу:

Send\_Vectors – приймає Bh, Zh від Т1

Send\_Matrixes\_Vector – приймає C2h, R, MО від Т4

Send\_Const – приймає a1 від Т1

Send\_Matrix\_Const – приймає MT2h, a від Т4

Задача Т3 має такі оператори входу:

Send\_Matrixes\_Vector – приймає Ch, R, MO від Т4

Send\_Matrix\_Const – приймає MTh, a від Т4

Send\_Result\_A2 – приймає A2h від Т1

Send\_Result\_A3 – приймає A3h від Т4

Задача Т4 має такі оператори входу:

Send\_Vectors – приймає B3h, Z3h від Т3

Send\_Const2 – приймає а2 від Т2

Send\_Matrixes\_Vector – приймає C4h, R, MO від Т5

Send\_Const4 – приймає а5 від Т5

Send\_Result – приймає A2h від Т5

Send\_Const3 – приймає а3 від Т3

Задача Т5 має такі оператори входу:

Send\_Vectors – приймає B2h, Z2h від Т4

Send\_Matrixes\_Vector – приймає C5h, R, MO від Т6

Send\_Const – приймає а6 від Т6

Send\_Matrix\_Const – приймає MT2h, a від Т4

Send\_Result – приймає Ah від Т6

Задача Т6 має такі оператори входу:

Send\_Matrix\_Const – приймає MTh, a від Т5

Send\_Vectors – приймає Bh, Zh від Т5

## 3.4 Тестування програми ПРГ2

Тестування проводилося на системі з характеристиками:

Центральний процесор:

Оперативна пам’ять:

Операційна система:

Версії програмного забезпечення:

Таблиця 3.2. Час виконання програм обчислення на системі зі ЛП

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N / P | 1 | 2 | 3 | 4 | 5 | 6 |
| 600 | 16,4268 | 8,5332 | 5,6784 | 4,1184 | 3,8376 | 3,26 |
| 1200 | 130,8374 | 65,876 | 45,2012 | 35,9364 | 31,4776 | 24,923 |
| 1800 | 437,5339 | 219,9231 | 149,455 | 111,5873 | 98,5405 | 82,2464 |

Коефіцієнт прискорення обраховується за формулою [22]:



Таблиця 3.3. Значення коефіцієнтів прискорення для системи з ЛП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N / P | 2 | 3 | 4 | 5 | 6 |
| 600 | 1,925046 | 2,892857 | 3,988636 | 4,280488 | 5,038896 |
| 1200 | 1,986116 | 2,894556 | 3,640804 | 4,156524 | 5,249665 |
| 1800 | 1,989486 | 2,927529 | 3,921001 | 4,440143 | 5,319794 |

Коефіцієнт ефективності обраховується за формулою [22]:



Таблиця 3.4. Значення коефіцієнтів ефективності для системи з ЛП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N / P | 2 | 3 | 4 | 5 | 6 |
| 600 | 96,25229 | 96,42857 | 99,71591 | 85,60976 | 83,9816 |
| 1200 | 99,30582 | 96,4852 | 91,02011 | 83,13048 | 87,49442 |
| 1800 | 99,47429 | 97,58431 | 98,02502 | 88,80286 | 88,66323 |

Рис. 3.2. Залежність коефіцієнту прискорення від кількості ЦП для систем з локальною пам’яттю

Рис. 3.3. Залежність коефіцієнту прискорення для систем з локальною пам’яттю при N=600

Рис. 3.4. Залежність коефіцієнту прискорення для систем з локальною пам’яттю при N=1200

Рис. 3.5. Залежність коефіцієнту прискорення для систем з локальною пам’яттю при N=1800

Рис. 3.6. Залежність коефіцієнту ефективності від кількості ЦП для систем з локальною пам’яттю

Рис. 3.7. Залежність коефіцієнту ефективності для систем з локальною пам’яттю при N=600

Рис. 3.8. Залежність коефіцієнту ефективності для систем з локальною пам’яттю при N=1200

Рис. 3.9. Залежність коефіцієнту ефективності для систем з локальною пам’яттю при N=1800

## 3.5 Висновки до Розділу 3

1. Реалізація концепції пересилки повідомлень в мові Ада виконана у вигляді механізму рандеву. Дана концепція дозволила ефективно обмінюватися даними між процесами.

2. Тестування програми для ПКС з локальною пам'яттю показало, що коефіцієнт прискорення лежить в діапазоні від 1.96 до 5.02; максимальний коефіцієнт прискорення отриманий при P = 6, N = 1200; коефіцієнт ефективності лежить в діапазоні від 72% до 99%; мінімальний коефіцієнт ефективності отриманий при P = 6, N = 600 та P = 5, N = 1200

# ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ

1. останні роки виробники процесорів не прагнуть до досягнення максимальної тактової частоти - замість цього вони нарощують міць CPU, збільшуючи кількість ядер.
2. На зміну застарілому сімейства Intel Core 2 Duo прийшли нові чотирьох ядерні процесори Intel серій Core i3, i5 і i7. Вони обзавелися мікроархітектурою Sandy Bridge і виробляються за 32-нанометровим техпроцесом. Тим часом AMD суттєво доопрацювала свій чотирьохядерний Phenom X4, збільшивши об'єм кеш-пам'яті і освоївши 45-нанометровий технологічний процес.
3. Одним з найважливіших нововведень є модульний принцип розташування ядер у системі х86 - по два на кожному модулі. Завдяки цій особливості компанії нескладно вибудувати модельний ряд, пропонуючи рішення з різними кількістю обчислювальних блоків і тактовими частотами.
4. Максимальний час виконання програми для ПКС зі спільною пам'яттю склав 437,5339 с. Для ПКС з локальною пам'яттю – 453,9751с. Таким чином програма для ПКС зі СП виявилась швидшою. Це пояснюється тим, що багато часу було виділено на проходження етапів збирання і передачі даних

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Багатоядрений процесор [Електронний ресурс] – Режим доступу: <http://uk.wikipedia.org/wiki/Багатоядерний_процесор> (дата звернення: 23.03.13). – Назва з екрану.
2. Intel Corporation [Електронний ресурс] – Режим доступу: <http://uk.wikipedia.org/wiki/Intel_Corporation> (дата звернення: 23.03.2013). – Назва з екрану.
3. Advanced Micro Devices [Електронний ресурс]: Торгові марки – Режим доступу: <http://uk.wikipedia.org/wiki/AMD> (дата звернення:23.03.13). – Назва з екрану
4. Валентин Соломенчук. Железо ПК 2010 / В.Г. Соломенчук, П.В. Соломенчук. – СПб.: БХВ-Петербург, 2010. – 448с. – ISBN 978-5-9775-0515-4
5. Turbo boost [Електронний ресурс] – Режим доступу: <http://ru.wikipedia.org/wiki/Intel_Turbo_Boost> (дата звернення: 23.03.13). – Назва з екрану
6. Супертест [Електронний ресурс]: 36 процессоров в свободном тестировании – Режим доступу: <http://itc.ua/articles/supertest_36_processorov_v_svodnom_testirovanii_50066/> (дата звернення 23.03.2013). – Назва з екрану
7. Intel® Core™ i7-975 Processor Extreme Edition (8M Cache, 3.33 GHz, 6.40 GT/s Intel® QPI)[Електронний ресурс] – Режим доступу: <http://ark.intel.com/products/37153/Intel-Core-i7-975-Processor-Extreme-Edition-8M-Cache-3_33-GHz-6_40-GTs-Intel-QPI> (дата звернення 23.03.2013). – Назва з екрану
8. Intel® Core™ i7-920 Processor (8M Cache, 2.66 GHz, 4.80 GT/s Intel® QPI) [Електронний ресурс] – Режим доступу: <http://ark.intel.com/products/37147/Intel-Core-i7-920-Processor-8M-Cache-2_66-GHz-4_80-GTs-Intel-QPI> (дата звернення 23.03.13). – Назва з екрану
9. Core i5 [Електронний ресурс] – Режим доступу: <http://ru.wikipedia.org/wiki/Intel_Core_i5> (дата звернення: 23.03.13). – Назва з екрану
10. Список микропроцессоров Core i5 [Електронний ресурс] – Режим доступу: <http://ru.wikipedia.org/wiki/Список_микропроцессоров_Core_i5> (дата звернення:23.03.13). – Назва з екрану.
11. Intel® Dynamic Virtual Client (DVC) Model for Education [Електронний ресурс] – Режим доступу: <http://www.intel.ru/content/www/ru/ru/education/education-core-vpro-intel-dynamic-virtual-client-brief.html> (дата звернення: 23.03.2013). – Назва з екрану
12. List of Intel Core 2 microprocessors [Електронний ресурс] – Режим доступу: <http://en.wikipedia.org/wiki/List_of_Intel_Core_2_microprocessors> (дата звернення 23.03.13). – Назва з екрану
13. AMD HyperTransport™ Technology [Електронний ресурс] – Режим доступу: <http://www.amd.com/ru/products/technologies/hypertransport-technology/Pages/hypertransport-technology.aspx> (дата звернення: 23.03.2013). – Назва з екрану
14. 3DNow! [Електронний ресурс] – Режим доступу: <http://uk.wikipedia.org/wiki/3DNow!> (дата звернення 23.03.2013). – Назва з екрану
15. AMD [Електронний ресурс] – Режим доступу: <http://uk.wikipedia.org/wiki/AMD64> (дата звернення 23.03.2013). – Назва з екрану
16. Enhanced Virus Protection [Електронний ресурс] – Режим доступу: <http://www.amd.com/us/products/technologies/enhanced-virus-protection/Pages/enhanced-virus-protection.aspx> (дата звернення 23.03.2013). – Назва з екрану
17. PowerNow! [Електронний ресурс] – Режим доступу: <http://uk.wikipedia.org/wiki/PowerNow!> (дата звернення 23.03.2013). – Назва з екрану
18. Список процесорів AMD Phenom [Електронний ресурс] – Режим доступу: [http://uk.wikipedia.org/wiki/ Список\_процесорів\_AMD\_Phenom](http://uk.wikipedia.org/wiki/%20Список_процесорів_AMD_Phenom%20%20)  (дата звернення 23.03.2013). – Назва з екрану
19. Модельный ряд процессоров AMD FX и сравнение характеристик [Електронний ресурс] – Режим доступу:  [<http://www.amd.com/ru/products/desktop/processors/amdfx/Pages/amdfx-model-number-comparison.aspx>](http://uk.wikipedia.org/wiki/%20Список_процесорів_AMD_Phenom%20%20)  (дата звернення 23.03.2013). – Назва з екрану
20. Тестирование процессоров AMD FX-6200 BE, FX-4170 BE, FX-4130 BE и Phenom II X4 960T BE в играх [Електронний ресурс] - Режим доступу: <http://www.overclockers.ru/lab/50122_3/Testirovanie_processorov_AMD_FX-6200_BE_FX-4170_BE_FX-4130_BE_i_Phenom_II_X4_960T_BE_v_igrah.html> (дата звернення 23.03.2013). – Назва з екрану
21. Эволюция четырехъядерности в играх: Intel. Yorkfield - Bloomfield - Lynnfield - Sandy Bridge - Ivy Bridge: <http://www.overclockers.ru/lab/49008_3/Evoljuciya_chetyrehyadernosti_v_igrah_Intel._Yorkfield_-_Bloomfield_-_Lynnfield_-_Sandy_Bridge_-_Ivy_Bridge.html> (дата звернення 23.03.2013). – Назва з екрану
22. Жуков І. А., Корочкін О. В. Паралельні та розподілені обчислення. Навч. посіб. – К.: «Корнійчук», 2005. – 226 с.

# ДОДАТКИ

# Додаток А. Схеми алгоритмів процесів

# Додаток Б. Схеми алгоритмів основних програм

# Додаток В. Лістинг програм

## Лістинг програми для системи зі спільною пам’яттю

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* PRO-2. Курсова робота \*

\* C# \*

\* A = (B\*C)\*Z + R\*MO\*MT \*

\* Seredenko Andrii, group IO-01 \*

\* 09.04.2013 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace CW

{

class Program

{

static DateTime startTime = DateTime.Now;

static int N = 1800;

static int P = 6;

static int H = N / P;

static Matrix MT = new Matrix(N);

static Matrix MO = new Matrix(N);

static Vector B = new Vector(N);

static Vector C = new Vector(N);

static Vector Z = new Vector(N);

static Vector R = new Vector(N);

static Vector A = new Vector(N);

static volatile int a = 0;

static EventWaitHandle E1 = new ManualResetEvent(false);

static EventWaitHandle E2 = new ManualResetEvent(false);

static EventWaitHandle E6 = new ManualResetEvent(false);

static Semaphore SemaphoreCalcAtomic1 = new Semaphore(0, 6);

static Semaphore SemaphoreCalcAtomic2 = new Semaphore(0, 6);

static Semaphore SemaphoreCalcAtomic3 = new Semaphore(0, 6);

static Semaphore SemaphoreCalcAtomic4 = new Semaphore(0, 6);

static Semaphore SemaphoreCalcAtomic5 = new Semaphore(0, 6);

static Semaphore SemaphoreCalcAtomic6 = new Semaphore(0, 6);

static Semaphore SemaphoreCalcA = new Semaphore(0, P);

static Object lck = new Object();

static int multiplyVV(Vector V1, Vector V2)

{

int r = 0;

for (int i = 0; i < V1.getLength(); i++)

{

r += V1.get(i) \* V2.get(i);

}

return r;

}

static Vector multiplyVM(Vector V, Matrix MM)

{

Vector R = new Vector(N);

for (int i = 0; i < V.getLength(); i++)

{

R.set(i, 0);

for (int j = 0; j < V.getLength(); j++)

{

int Ri = R.get(i);

R.set(i, Ri + V.get(j) \* MM.get(i, j));

}

}

return R;

}

static int multiplyConstV(int a, Vector V)

{

int r = 0;

for (int i = 0; i < V.getLength(); i++)

{

r += a \* V.get(i);

}

return r;

}

//--------------Thread 1---------------

public static void T1()

{

Console.WriteLine("Thread 1 has started.");

int start = 0;

int end = H;

// 1.

Vector.FillWithOnes(B);

Vector.FillWithOnes(Z);

// 2.

E1.Set();

// 3.

WaitHandle[] E26 = new WaitHandle[] { E2, E6 };

WaitHandle.WaitAll(E26);

// 4.

int a1 = 0;

for (int i = start; i < end; i++)

{

a1 += B.get(i) \* C.get(i);

}

// 5.

a += a1;

// 6.

SemaphoreCalcAtomic1.Release(6);

// 7.

Semaphore[] s1 = { SemaphoreCalcAtomic2, SemaphoreCalcAtomic3, SemaphoreCalcAtomic4, SemaphoreCalcAtomic5, SemaphoreCalcAtomic6 };

WaitHandle.WaitAll(s1);

// 8.

a1 = 0;

Vector R1 = new Vector(N);

Matrix MO1 = new Matrix(N);

lock (lck)

{

R1 = R.clone();

MO1 = MO.copy();

}

a1 = a;

// 9.

Vector ZH = new Vector(N);

Vector MTH = new Vector(N);

Vector tmp = new Vector(H);

for (int i = start; i < end; i++)

{

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

{

MTH.set(j, MT.get(i, j));

}

ZH.set(i, Z.get(i));

int dd = a1 \* ZH.get(i) + multiplyVV(R1, multiplyVM(MTH, MO1));

A.set(i, dd);

}

// 10.

SemaphoreCalcA.WaitOne();

if (N < 8)

Vector.print(A);

Console.WriteLine();

Console.WriteLine("Thread 1 has finished");

TimeSpan sp = DateTime.Now - startTime;

Console.WriteLine("Working time: " + sp);

}

///////////////////////////////////

//Thread 2/////////////////////////

static void T2()

{

Console.WriteLine("Thread 2 has started.");

int start = H;

int end = 2 \* H;

// 1.

Vector.FillWithOnes(C);

Vector.FillWithOnes(R);

Matrix.FillWithOnes(MO);

// 2.

E2.Set();

// 3.

WaitHandle[] E16 = new WaitHandle[] { E1, E6 };

WaitHandle.WaitAll(E16);

// 4.

int a2 = 0;

for (int i = start; i < end; i++)

{

a2 += B.get(i) \* C.get(i);

}

// 5.

a += a2;

// 6.

SemaphoreCalcAtomic2.Release(6);

// 7.

Semaphore[] s1 = { SemaphoreCalcAtomic1, SemaphoreCalcAtomic3, SemaphoreCalcAtomic4, SemaphoreCalcAtomic5, SemaphoreCalcAtomic6 };

WaitHandle.WaitAll(s1);

// 8.

a2 = 0;

Vector R2 = new Vector(N);

Matrix MO2 = new Matrix(N);

lock (lck)

{

R2 = R.clone();

MO2 = MO.copy();

}

a2 = a;

// 9.

Vector ZH = new Vector(N);

Vector MTH = new Vector(N);

for (int i = start; i < end; i++)

{

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

{

MTH.set(j, MT.get(i, j));

}

ZH.set(i, Z.get(i));

int dd = a2 \* ZH.get(i) + multiplyVV(R2, multiplyVM(MTH, MO2));

A.set(i, dd);

}

// 10.

SemaphoreCalcA.Release();

Console.WriteLine("Thread 2 has finished.");

}

//--------------------Thread 3-----------------------

static void T3()

{

Console.WriteLine("Thread 3 has started.");

int start = 2 \* H;

int end = 3 \* H;

// 3.

WaitHandle[] E126 = new WaitHandle[] { E1, E2, E6 };

WaitHandle.WaitAll(E126);

// 4.

int ai = 0;

for (int i = start; i < end; i++)

{

ai += B.get(i) \* C.get(i);

}

// 5.

a += ai;

// 6.

SemaphoreCalcAtomic3.Release(6);

// 7.

Semaphore[] s1 = { SemaphoreCalcAtomic1, SemaphoreCalcAtomic2, SemaphoreCalcAtomic4, SemaphoreCalcAtomic5, SemaphoreCalcAtomic6 };

WaitHandle.WaitAll(s1);

// 8.

ai = 0;

Vector Ri = new Vector(N);

Matrix MOi = new Matrix(N);

lock (lck)

{

Ri = R.clone();

MOi = MO.copy();

}

ai = a;

// 9.

Vector ZH = new Vector(N);

Vector MTH = new Vector(N);

for (int i = start; i < end; i++)

{

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

{

MTH.set(j, MT.get(i, j));

}

ZH.set(i, Z.get(i));

int dd = ai \* ZH.get(i) + multiplyVV(Ri, multiplyVM(MTH, MOi));

A.set(i, dd);

}

// 10.

SemaphoreCalcA.Release();

Console.WriteLine("Thread 3 has finished.");

}

//--------------------Thread 4-----------------------

static void T4()

{

Console.WriteLine("Thread 4 has started.");

int start = 3 \* H;

int end = 4 \* H;

// 3.

WaitHandle[] E126 = new WaitHandle[] { E1, E2, E6 };

WaitHandle.WaitAll(E126);

// 4.

int ai = 0;

for (int i = start; i < end; i++)

{

ai += B.get(i) \* C.get(i);

}

// 5. Calc

a += ai;

// 6.

SemaphoreCalcAtomic4.Release(6);

// 7.

Semaphore[] s1 = { SemaphoreCalcAtomic1, SemaphoreCalcAtomic2, SemaphoreCalcAtomic3, SemaphoreCalcAtomic5, SemaphoreCalcAtomic6 };

WaitHandle.WaitAll(s1);

// 8.

ai = 0;

Vector Ri = new Vector(N);

Matrix MOi = new Matrix(N);

lock (lck)

{

Ri = R.clone();

MOi = MO.copy();

}

ai = a;

// 9.

Vector ZH = new Vector(N);

Vector MTH = new Vector(N);

for (int i = start; i < end; i++)

{

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

{

MTH.set(j, MT.get(i, j));

}

ZH.set(i, Z.get(i));

int dd = ai \* ZH.get(i) + multiplyVV(Ri, multiplyVM(MTH, MOi));

A.set(i, dd);

}

// 10.

SemaphoreCalcA.Release();

Console.WriteLine("Thread 3 has finished.");

}

//--------------------Thread 5-----------------------

static void T5()

{

Console.WriteLine("Thread 5 has started.");

int start = 4 \* H;

int end = 5 \* H;

// 3

WaitHandle[] E126 = new WaitHandle[] { E1, E2, E6 };

WaitHandle.WaitAll(E126);

// 4.

int ai = 0;

for (int i = start; i < end; i++)

{

ai += B.get(i) \* C.get(i);

}

// 5.

a += ai;

// 6.

SemaphoreCalcAtomic5.Release(6);

// 7.

Semaphore[] s1 = { SemaphoreCalcAtomic1, SemaphoreCalcAtomic2, SemaphoreCalcAtomic3, SemaphoreCalcAtomic4, SemaphoreCalcAtomic6 };

WaitHandle.WaitAll(s1);

// 8.

ai = 0;

Vector Ri = new Vector(N);

Matrix MOi = new Matrix(N);

lock (lck)

{

Ri = R.clone();

MOi = MO.copy();

}

ai = a;

// 9.

Vector ZH = new Vector(N);

Vector MTH = new Vector(N);

for (int i = start; i < end; i++)

{

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

{

MTH.set(j, MT.get(i, j));

}

ZH.set(i, Z.get(i));

int dd = ai \* ZH.get(i) + multiplyVV(Ri, multiplyVM(MTH, MOi));

A.set(i, dd);

}

// 10.

SemaphoreCalcA.Release();

Console.WriteLine("Thread 5 has finished.");

}

//-----------------Thread 6-------------------

static void T6()

{

int start = 5 \* H;

int end = N;

Console.WriteLine("Thread 6 has started.");

// 1.

Matrix.FillWithOnes(MT);

// 2.

E6.Set();

// 3.

//E2.WaitOne();

WaitHandle[] E12 = new WaitHandle[] { E1, E2 };

WaitHandle.WaitAll(E12);

// 4.

int a6 = 0;

for (int i = start; i < end; i++)

{

a6 += B.get(i) \* C.get(i);

}

// 5.

a += a6;

// 6.

SemaphoreCalcAtomic6.Release(6);

// 7.

Semaphore[] s1 = { SemaphoreCalcAtomic1, SemaphoreCalcAtomic2, SemaphoreCalcAtomic3, SemaphoreCalcAtomic4, SemaphoreCalcAtomic5 };

WaitHandle.WaitAll(s1);

// 8.

a6 = 0;

Vector R6 = new Vector(N);

Matrix MO6 = new Matrix(N);

lock (lck)

{

R6 = R.clone();

MO6 = MO.copy();

}

a6 = a;

// 9.

Vector ZH = new Vector(N);

Vector MTH = new Vector(N);

for (int i = start; i < end; i++)

{

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

{

MTH.set(j, MT.get(i, j));

}

ZH.set(i, Z.get(i));

int dd = a6 \* ZH.get(i) + multiplyVV(R6, multiplyVM(MTH, MO6));

A.set(i, dd);

}

// 10.

SemaphoreCalcA.Release();

Console.WriteLine("Thread 6 has finished.");

}

//------------Main thread--------------

static void Main(string[] args)

{

new Thread(T1).Start();

new Thread(T2).Start();

new Thread(T3).Start();

new Thread(T4).Start();

new Thread(T5).Start();

new Thread(T6).Start();

Console.ReadLine();

}

//-----------------------------------------

}

}

## Лістинг програми для системи з локальною пам’яттю

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- ПРО-2. Курсова робота \*

-- Ада. Рандеву \*

-- A = (B\*C)\*Z + R\*MO\*MT \*

-- Середенко Андрій, гр. IO-01 \*

-- 19.04.2013 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Synchronous\_Task\_Control;

use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Synchronous\_Task\_Control;

procedure Main is

N: integer := 6;

P: integer := 6;

H: integer := N / P;

type Vector\_General is array(integer range <>) of integer;

subtype Vector is Vector\_General(1..N);

subtype Vector\_5H is Vector\_General(1..(5 \* H));

subtype Vector\_4H is Vector\_General(1..(4 \* H));

subtype Vector\_3H is Vector\_General(1..(3 \* H));

subtype Vector\_2H is Vector\_General(1..(2 \* H));

subtype Vector\_H is Vector\_General(1..H);

type Matrix\_General is array(integer range <>) of Vector;

subtype Matrix is Matrix\_General(1..N);

subtype Matrix\_4H is Matrix\_General(1..(4 \* H));

subtype Matrix\_3H is Matrix\_General(1..(3 \* H));

subtype Matrix\_2H is Matrix\_General(1..(2 \* H));

subtype Matrix\_H is Matrix\_General(1..H);

--Task specification T1-----------------------------------------------

task T1 is

entry Send\_Matrixes\_Vector(Ch: in Vector\_H; R: in Vector; MO: in Matrix);

entry Send\_Matrix\_Const(MTh:in Matrix\_H; a:in Integer);

entry Send\_Vectors(Z2h:in Vector\_2H; B2h: in Vector\_2H);

entry Send\_Result(Ah: in Vector\_H);

end T1;

----------------------------------------------------------------------

--Task specification T2-----------------------------------------------

task T2 is

entry Send\_Vectors(Bh: in Vector\_H; Zh:in Vector\_H);

entry Send\_Matrixes\_Vector(C2h:in Vector\_2H; R:Vector; MO: in Matrix);

entry Send\_Const(t1:in Integer);

entry Send\_Matrix\_Const(MT2h: in Matrix\_2H; a: in Integer);

end T2;

----------------------------------------------------------------------

--Task specification T3-----------------------------------------------

task T3 is

entry Send\_Matrixes\_Vector(Ch:in Vector\_H; R: in Vector; MO:in Matrix);

entry Send\_Matrix\_Const(MTh: in Matrix\_H; a: in Integer);

entry Send\_Result\_A3(A3h: in Vector\_3H);

entry Send\_Result\_A2(A2h: in Vector\_2H);

end T3;

----------------------------------------------------------------------

--Task specification T4-----------------------------------------------

task T4 is

entry Send\_Matrixes\_Vector(C4h: in Vector\_4H; R: in Vector; MO: in Matrix);

entry Send\_Vectors(B3h: in Vector\_3H; Z3h: in Vector\_3H);

entry Send\_Const2(t: in Integer);

entry Send\_Const3(t: in Integer);

entry Send\_Const4(t: in Integer);

entry Send\_Result(A2h: in Vector\_2H);

end T4;

----------------------------------------------------------------------

--Task specification T5-----------------------------------------------

task T5 is

entry Send\_Vectors(B2h: in Vector\_2H; Z2h:in Vector\_2H);

entry Send\_Matrixes\_Vector(C5h:in Vector\_5H; R: in Vector; MO: in Matrix);

entry Send\_Const(t: in Integer);

entry Send\_Matrix\_Const(MT2h: in Matrix\_2H; a: in Integer);

entry Send\_Result(Ah: in Vector\_H);

end T5;

----------------------------------------------------------------------

--Task specification T6-----------------------------------------------

task T6 is

entry Send\_Vectors(Bh: in Vector\_H; Zh : in Vector\_H);

entry Send\_Matrix\_Const(MTh: in Matrix\_H; a: in Integer);

end T6;

----------------------------------------------------------------------

--Task body T1--------------------------------------------------------

task body T1 is

ti: integer;

Zh\_in, Bh\_in: Vector\_H;

Zh\_out, Bh\_out: Vector\_H;

C\_in,A1:Vector\_H;

R1,TV: Vector;

MO1: Matrix;

t1,alpha1:Integer;

MT1 : Matrix\_H;

A2:Vector\_2H;

begin

put\_line("Task T1 has started.");

-- 1.

accept Send\_Vectors (Z2h : in Vector\_2H; B2h : in Vector\_2H) do

Zh\_in := Z2h(1..H);

Bh\_in := Z2h(1..H);

Zh\_out := Z2h((H+1)..2\*H);

Bh\_out := Z2h((H+1)..2\*H);

end Send\_Vectors;

-- 2.

T2.Send\_Vectors(Bh\_out,Zh => Zh\_out);

-- 3.

accept Send\_Matrixes\_Vector (Ch : in Vector\_H; R : in Vector; MO : in Matrix) do

C\_in := Ch;

R1 := R;

MO1 := MO;

end Send\_Matrixes\_Vector;

-- 4.

t1:=0;

for i in 1..Bh\_in'Length loop

t1 := t1 + Bh\_in(i)\*C\_in(i);

end loop;

-- 5.

T2.Send\_Const(t1);

--5.

accept Send\_Matrix\_Const (MTh : in Matrix\_H; a : in Integer) do

MT1 := MTh(1..H);

alpha1 := a;

end Send\_Matrix\_Const;

-- 4.

for i in 1..H loop

for j in 1..N loop

TV(j) := 0;

for k in 1..N loop

TV(j) := TV(j) + MO1(k)(j) \* MT1(i)(k);

end loop;

end loop;

ti := 0;

for j in 1..N loop

ti := ti + R1(j) \* TV(j);

end loop;

A1(i) := Zh\_in(i)\*alpha1 + ti;

end loop;

accept Send\_Result(Ah: in Vector\_H) do

for i in 1..H loop

A2(i):=A1(i);

end loop;

ti:=0;

for i in H+1..2\*H loop

A2(i):= A1(ti);

ti:=t1+1;

end loop;

end Send\_Result;

T3.Send\_Result\_A2(A2);

put\_line("Task T1 has finished.");

end T1;

----------------------------------------------------------------------

--Task body T2--------------------------------------------------------

task body T2 is

t2,t\_1, ti, alpha2: integer;

B2, Z2: Vector\_H;

C\_in, C\_out: Vector\_H;

R2:Vector;

MO2: Matrix;

MT\_in, MT\_out:Matrix\_H;

A2: Vector\_H;

TV: Vector;

begin

put\_line("Task T2 has started.");

-- 1.

accept Send\_Vectors (Bh : in Vector\_H; Zh : in Vector\_H) do

B2 := Bh;

Z2 := Zh;

end Send\_Vectors;

accept Send\_Matrixes\_Vector (C2h : in Vector\_2H; R : Vector; MO : in Matrix) do

C\_in := C2h(1..H);

C\_out := C2h((H+1)..2\*H);

R2 := R;

MO2 := MO;

end Send\_Matrixes\_Vector;

-- 2.

T1.Send\_Matrixes\_Vector(C\_out,R => R2,MO => MO2 );

-- 3.

accept Send\_Const (t1 : in Integer) do

t\_1:=t1;

end Send\_Const;

-- 4.

t2:=0;

for i in 1..C\_in'Length loop

t2:= t2 + C\_in(i) \* B2(i);

end loop;

--5.

T4.Send\_Const2(t2+t\_1);

--6.

accept Send\_Matrix\_Const (MT2h : in Matrix\_2H; a : in Integer) do

MT\_in := MT2h(1..H);

MT\_out := MT2h((H+1)..2\*H);

alpha2 := a;

end Send\_Matrix\_Const;

--7.

T1.Send\_Matrix\_Const(MTh => MT\_out,

a => alpha2);

-- 8.

for i in 1..H loop

for j in 1..N loop

TV(j) := 0;

for k in 1..N loop

TV(j) := TV(j) + MO2(k)(j) \* MT\_in(i)(k);

end loop;

end loop;

ti := 0;

for j in 1..N loop

ti := ti + R2(j) \* TV(j);

end loop;

A2(i) := Z2(i)\*alpha2 + ti;

end loop;

-- 9.

T1.Send\_Result(A2);

put\_line("Task T3 has finished.");

end T2;

----------------------------------------------------------------------

--Task body T3--------------------------------------------------------

task body T3 is

ti, t3: integer;

A3, B3,Z3: Vector;

C3 :Vector\_H;

A: Vector\_3H;

R3, TV: Vector;

MT3: Matrix\_H;

MO3: Matrix;

alpha3:Integer;

begin

put\_line("Task T3 has started.");

-- 1.

for i in 1..N loop

B3(i):=1;

Z3(i):=1;

end loop;

-- 2.

T1.Send\_Vectors(Z2h => Z3((H+1)..3\*H), B2h => B3((H+1)..3\*H));

-- 3.

T4.Send\_Vectors(B3h => B3((3\*H+1)..N),Z3h => Z3((3\*H+1)..N));

--4.

accept Send\_Matrixes\_Vector (Ch : in Vector\_H; R : in Vector; MO : in Matrix) do

C3:=Ch;

R3:=R;

MO3:=MO;

end Send\_Matrixes\_Vector;

-- 5.

t3:=0;

for i in 1..C3'Length loop

t3:= t3 + C3(i) \* B3(i);

end loop;

-- 6.

T4.Send\_Const4(t3);

--7.

accept Send\_Matrix\_Const (MTh : in Matrix\_H; a : in Integer) do

MT3 := MTh;

alpha3:=a;

end Send\_Matrix\_Const;

--8.

accept Send\_Result\_A2 (A2h : in Vector\_2H) do

A(h+1..3\*H):=A2h;

end Send\_Result\_A2;

accept Send\_Result\_A3 (A3h : in Vector\_3H) do

A((2\*H+1)..N) := A3h;

end Send\_Result\_A3;

-- 10.

for i in 1..H loop

for j in 1..N loop

TV(j) := 0;

for k in 1..N loop

TV(j) := TV(j) + MO3(k)(j) \* MT3(i)(k);

end loop;

end loop;

ti := 0;

for j in 1..N loop

ti := ti + R3(j) \* TV(j);

end loop;

A3(i) := Z3(i)\*alpha3 + ti;

end loop;

put("A = (");

for i in 1..(N - 1) loop

put(A1(i));

put(", ");

end loop;

put(A3(N));

put\_line(")");

put\_line("Task T3 has finished.");

end T3;

----------------------------------------------------------------------

--Task body T4--------------------------------------------------------

task body T4 is

ti, z,t\_3,t4,result\_t4,t2t1,t5t6: integer;

A4: Vector\_H;

A3h,B4,Z4: Vector\_3H;

TV,R4: Vector;

C4: Vector\_4H;

MO4,MT: Matrix;

begin

put\_line("Task T4 has started.");

-- 1. Receive B3h, C, a, MX3h

for i in 1 .. n loop

for j in 1 .. n loop

MT(i)(j):=1;

end loop;

end loop;

accept Send\_Matrixes\_Vector (C4h : in Vector\_4H; R : in Vector; MO : in Matrix) do

MO4:=MO;

R4:=R;

C4:=C4h;

end Send\_Matrixes\_Vector;

t3.Send\_Matrixes\_Vector(Ch => C4(3\*h+1 .. 4\*h),R => R4,MO => MO4);

t2.Send\_Matrixes\_Vector(C2h => C4(1+2\*h.. 4\*h),R => R4,MO => MO4);

accept Send\_Vectors(B3h: in Vector\_3H; Z3h: in Vector\_3H) do

B4 := B3h;

Z4 := Z3h;

end Send\_Vectors;

t5.Send\_Vectors(B2h => B4(h+1 .. 3\*h),

Z2h => C4(h+1 .. 3\*h));

accept Send\_Const4 (t : in Integer) do

t\_3:=t;

end Send\_Const4;

accept Send\_Const3 (t : in Integer) do

t5t6:=t;

end Send\_Const3;

accept Send\_Const2 (t : in Integer) do

t2t1:=t;

end Send\_Const2;

t4:=0;

for i in 1..C4'Length loop

t4:= t4 + C4(i) \* B4(i);

end loop;

result\_t4:=t4+t5t6+t2t1+t\_3;

t3.Send\_Matrix\_Const(MTh =>MT(2\*h+1 .. 3\*h) ,

a => result\_t4);

t5.Send\_Matrix\_Const(MT2h => MT(4\*h+1 .. N),

a => result\_t4);

t2.Send\_Matrix\_Const(MT2h => MT(1 .. 2\*h),

a => result\_t4);

-- 5.

for i in 1..H loop

for j in 1..N loop

TV(j) := 0;

for k in 1..N loop

TV(j) := TV(j) + MO4(k)(j) \* MT(i)(k);

end loop;

end loop;

ti := 0;

for j in 1..N loop

ti := ti + R4(j) \* TV(j);

end loop;

A4(i) := Z4(i)\*result\_t4 + ti;

end loop;

-- 6. Receive A2h from T5

accept Send\_Result(A2h: in Vector\_2H) do

z := 1;

for i in (H + 1)..(3 \* H) loop

A3h(i) := A2h(z);

z := z + 1;

end loop;

end Send\_Result;

--

T3.Send\_Result(A3h);

put\_line("Task T4 has finished.");

end T4;

----------------------------------------------------------------------

--Task body T5--------------------------------------------------------

task body T5 is

ti, z,t5,t\_6,t5t6,a\_5: integer;

A5: Vector\_H;

A2h,B5,Z5: Vector\_2H;

C: Vector\_5H;

TV,R5: Vector;

MO5 : Matrix;

MT: Matrix\_2H;

begin

put\_line("Task T5 has started.");

-- 1.

accept Send\_Matrixes\_Vector (C5h : in Vector\_5H; R : in Vector; MO : in Matrix) do

MO5:=MO;

R5:=R;

C:=C5h;

end Send\_Matrixes\_Vector;

t4.Send\_Matrixes\_Vector(C4h =>C (1 .. 4\*h), R => R5, MO => MO5);

accept Send\_Vectors (B2h : in Vector\_2H; Z2h : in Vector\_2H) do

B5:=B2h;

Z5:=Z2h;

end Send\_Vectors;

t6.Send\_Vectors(Bh => B5(h+1 .. 2\*h),

Zh => Z5(h+1 .. 2\*h));

-- 2.

accept Send\_Const (t : in Integer) do

t\_6:=t;

end Send\_Const;

-- 4.

t5:=0;

for i in 1..C'Length loop

t5:= t5 + C(i) \* B5(i);

end loop;

t5t6:=t5+t\_6;

--5.

T4.Send\_Const3(t5t6);

accept Send\_Matrix\_Const (MT2h : in Matrix\_2H; a : in Integer) do

MT:= MT2h;

A\_5:=a;

end Send\_Matrix\_Const;

-- 5.

for i in 1..H loop

for j in 1..N loop

TV(j) := 0;

for k in 1..N loop

TV(j) := TV(j) + MO5(k)(j) \* MT(i)(k);

end loop;

end loop;

ti := 0;

for j in 1..N loop

ti := ti + R5(j) \* TV(j);

end loop;

A5(i) := Z5(i)\*A\_5+ ti;

end loop;

-- 6.

accept Send\_Result(Ah: in Vector\_H) do

z := 1;

for i in (H + 1)..(2 \* H) loop

A2h(i) := Ah(z);

z := z + 1;

end loop;

end Send\_Result;

--Send A2h to task T4

T4.Send\_Result(A2h);

put\_line("Task T5 has finished.");

end T5;

----------------------------------------------------------------------

--Task body T6--------------------------------------------------------

task body T6 is

ti,t6,a\_6: integer;

A6, B6,Z6: Vector\_H;

TV,C,R: Vector;

MT: Matrix\_H;

MO: Matrix;

begin

put\_line("Task T6 has started.");

for i in 1.. n loop

for j in 1 .. n loop

MO(i)(j):=1;

end loop;

C(i):=1;

R(i):=1;

end loop;

t5.Send\_Matrixes\_Vector(C(1 .. 5\*h),R => R,MO => MO);

accept Send\_Vectors (Bh : in Vector\_H; Zh : in Vector\_H) do

B6:= Bh;

Z6:=Zh;

end Send\_Vectors;

t6:=0;

for i in 1..C'Length loop

t6 := t6 + B6(i)\*C(i);

end loop;

-- 5.

T5.Send\_Const(t6);

accept Send\_Matrix\_Const (MTh : in Matrix\_H; a : in Integer) do

MT:=MTh;

a\_6:=a;

end Send\_Matrix\_Const;

-- 2.

for i in 1..H loop

for j in 1..N loop

TV(j) := 0;

for k in 1..N loop

TV(j) := TV(j) + MO(k)(j) \* MT(i)(k);

end loop;

end loop;

ti := 0;

for j in 1..N loop

ti := ti + R(j) \* TV(j);

end loop;

A6(i) := Z6(i)\*a\_6 + ti;

end loop;

-- 3.

T5.Send\_Result(A6);

put\_line("Task T6 has finished.");

end T6;

----------------------------------------------------------------------

begin

null;

end Main;