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»

Соколовський С.В.

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

Соколовському Станіславу Валерійовичу

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 13 травня 2013 р.

3. Вихідні дані до роботи

- огляд і порівняння шестиядерних процесорів компаній AMD і Intel

- математична задача A=(B+C\*MO)\*(MZ \* MK\*α)

- структури ПКС ОП та ПКС ЛП

- мова програмування: Ada

- засоби організації взаємодії процесів: захищені модулі, механізм рандеву мови Ada

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

розробити)

- огляд і порівняння шестиядерних 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** \_\_\_\_\_\_\_\_\_ Соколовський С.В.

( підпис )

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

( підпис )

ТЕХНІЧНЕ ЗАВДАННЯ

Ада. Захищений модуль  


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\*MO)\*(MZ \* MK\*α)

Зміст

[ВВЕДЕННЯ 6](#_Toc356210210)

[РОЗДІЛ 1. ПОРІВНЯННЯ ШЕСТИЯДЕРНИХ ПРОЦЕСОРІВ INTEL І AMD. 7](#_Toc356210211)

[1.1. Огляд сучасних шестиядерних процесорів Intel. 7](#_Toc356210212)

[1.2. Огляд сучасних шестиядерних процесорів AMD. 15](#_Toc356210213)

[1.3. Тести на продуктивність. 19](#_Toc356210214)

[1.4. Висновки до розділу 1 23](#_Toc356210215)

[РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП 25](#_Toc356210216)

[2.1. Аналіз задачі на внутрішній паралелізм з використанням концепції необмеженого паралелізму. 25](#_Toc356210218)

[2.2. Розробка паралельного математичного алгоритму. 26](#_Toc356210219)

[2.3. Розробка алгоритмів процесів 26](#_Toc356210220)

[2.4. Розробка схеми взаємодії процесів 29](#_Toc356210221)

[2.5. Розробка програми ПРГ1 29](#_Toc356210222)

[2.6. Тестування програми ПРГ1 30](#_Toc356210223)

[2.7. Висновки до розділу 2 34](#_Toc356210224)

[РОЗДІЛ 3.РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП 35](#_Toc356210225)

[3.1. Розробка паралельного математичного алгоритму. 35](#_Toc356210229)

[3.2. Розробка алгоритмів процесів 35](#_Toc356210230)

[3.3. Розробка схеми взаємодії процесів 37](#_Toc356210231)

[3.4. Розробка програми ПРГ2 37](#_Toc356210232)

[3.5. Тестування програми ПРГ2 40](#_Toc356210233)

[3.6. Висновки до розділу 3 42](#_Toc356210234)

[ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ 43](#_Toc356210235)

[СПИСОК СКОРОЧЕНЬ 44](#_Toc356210236)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 45](#_Toc356210237)

[Додаток А 47](#_Toc356210238)

[Додаток Б 51](#_Toc356210239)

[Додаток В 57](#_Toc356210240)

[Додаток Г 66](#_Toc356210241)

# ВВЕДЕННЯ

В курсовій роботі представлено огляд сучасних шестиядерних процесорів компаній Intel та AMD. Розглянуто основні характеристики та переваги кожного розглянутого сімейства процесорів. Уточнено функції технологій, які підтримують процесори, а також основне призначення процесорів в цілому. Приведено декілька тестів для порівняння продуктивності даних процесорів (від незалежних джерел). А також проаналізовано можливі види діяльності, для яких доцільно використовувати той чи інший процесор.

В розділі 2 і 3 представлено реалізацію конкретної математичної задачі для паралельної комп'ютерної системи з СП та з ЛП. Створено відповідні алгоритми, схеми взаємодії задач, а також написано програми на мові Ada з використанням концепції посилки повідомлень( Rendezvous) та захищених модулів . [11]

Виконано дослідження ефективності задачі в залежності від розмірності N. Виконано тестування створеного ПЗ на реальній системі та визначено відповідні значення коефіцієнтів прискорення та ефективності програм.

# РОЗДІЛ 1. ПОРІВНЯННЯ ШЕСТИЯДЕРНИХ ПРОЦЕСОРІВ INTEL І AMD.

### Огляд сучасних шестиядерних процесорів Intel.

На сьогодні в продукції компанії Intel шестиядерні процесори представлені у сімействах процесорів для настільних комп’ютерів Intel® Core™ i7 та серверних процесорів Intel® Xeon®.

**Огляд та специфікація шестиядерних процесорів сімейства Intel® Core™ i7.**

**Архітектура Nehalem**

Nehalem - мікроархітектура процесорів компанії Intel, представлена ​​в 4 кварталі 2008 року. Мікропроцесори продаються під торговою маркою Core i7 і Core i5. Nehalem є кодову назву мікроархітектури процесорів Intel, наступник Core мікроархітектури. Nehalem процесори використовують 45-нм техпроцесу.

Nehalem архітектура в корені відрізняється від Netburst, зберігаючи при цьому деякі незначні особливості останнього. Мікропроцесори на основі Nehalem використовують більш високі тактові частоти і є більш енергоефективним, ніж Penryn мікропроцесори . Hyper-Threading технологія наявна​, поряд зі скороченням обсягу кеш-пам'яті L2, а також збільшенням кеш-пам'яті L3. Було повідомлено, що Nehalem має фокус на продуктивність, таким чином буде збільшенно частоту ядра у порівнянні з Penryn[12].

Розглянемо специфікацію процесорів Intel® Xeon E6540, Intel® Xeon E7530, Intel® Xeon E7540, Intel® Xeon X7542, Intel® Xeon L7545 (таб.1.1).

Таб.1.1. Специфікація шестиядерних процесорів сімейства Intel® Xeon.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Processor Number | E6540 | E7530 | E7540 | X7542 | L7545 |
| # of Cores | 6 | 6 | 6 | 6 | 6 |
| # of Threads | 12 | 12 | 12 | 6 | 12 |
| Clock Speed | 2 GHz | 1.866 GHz | 2 GHz | 2.666 GHz | 1.866 GHz |
| Max Turbo Frequency | 2.266 GHz | 2.133 GHz | 2.266 GHz | 2.8 GHz | 2.533 GHz |
| Cache | 18 MB L3 Cache | 12 MB L3 Cache | 18 MB L3 Cache | 18 MB L3 Cache | 18 MB L3 Cache |
| Bus Type | QPI | QPI | QPI | QPI | QPI |
| System Bus | 6.4 GT/s | 5.86 GT/s | 6.4 GT/s | 5.86 GT/s | 5.86 GT/s |
| Lithography | 45 nm | 45 nm | 45 nm | 45 nm | 45 nm |
| Max TDP | 105 W | 105 W | 105 W | 130 W | 95 W |
| Intel® Turbo Boost Technology | Yes | Yes | Yes | Yes | Yes |
| Intel® Hyper-Threading Technology | Yes | Yes | Yes |  | Yes |
| Intel® Virtualization Technology (VT-x) | Yes | Yes | Yes | Yes | Yes |

Бачимо, що дані процесори виготовлені по 45нм технології з підтримкою 64-розрядних систем, технології Intel® Virtualization, Intel® Turbo Boost, Intel® Hyper-Threading.

Шестиядерні процесори Intel® Xeon включають інноваційні процесорні технології:

* Технологія Intel® Turbo Boost максимально підвищує швидкість роботи ресурсномістких додатків, динамічно збільшуючи продуктивність у відповідності з навантаженням - більш висока продуктивність, коли це потрібно, і більш високе енергозбереження, коли збільшена продуктивність не потрібна.
* Технологія Intel® Hyper-Threading дозволяє багатопотокових додатків виконувати більше завдань паралельно. З 8 потоками, доступними операційній системі, багатозадачність стає ще легше.

**Архітектура Westmere**

Наступником Nehalem архітектура стала архітектура Westmere. Шестиядерні процесори побудовані на основі архітектури Westmere. Першим представником архітектури були процесори [Clarkdale](http://ru.wikipedia.org/wiki/Clarkdale), які володіли двома ядрами та графічним мікропроцесором, виробленим по 45-нм техпроцесу, що дозволить позбавиться від інтегрованої графіки в системній логіці.

Потім в серійне виробництво увійшов флагманський дизайн ядер даної архітектури - Gulftown, він володіє шістьма ядрами, дванадцятьма потоками, 12 Мб загального кеша третього рівня. У всьому іншому Gulftown по архітектурі дуже схожий на Bloomfield. Кожне ядро ​​отримує 32 кбайт кеша L1 для даних, 32 кбайт кеша L1 для інструкцій і виділений кеш L2 об'ємом 256 кбайт[1].

12 Мбайт загального кешу L3 потенційно можуть значно збільшити продуктивність. Оскільки кеш може динамічно розподілятися, програма, яка навантажує одне ядро, потенційно може монополізувати весь кеш. За інформацією Intel, від цього з'являються певні переваги в іграх, наприклад, але дуже важко оцінити, який приріст продуктивності пов'язаний із збільшеним числом ядер, а який - із збільшеним розміром кешу, тим більше, що ми використовуємо все менше однопоточних додатків в тестах.

Не дивлячись на додавання двох ядер і 4 Мбайт кешу L3, Gulftown за площею кристала виявився менше попередника (248 квадратних міліметрів проти 263 у Bloomfield). Число транзисторів збільшилася з 731 мільйона до 1,17 мільярда. Вражає, оскільки процесор Core i7-980X Extreme Edition відповідає тому ж тепловому пакету 130 Вт, що й існуюча лінійка процесорів Core i7-900.[5]

Контролер пам'яті Gulftown залишається незмінним, він як і раніше сертифікований під три канали пам'яті DDR3-1066. Це насправді дуже цікаво, оскільки 130-Вт процесори Westmere-EP, які Intel планує випустити Intel разом з Gulftown, підтримують пам'ять DDR3-1333 (причому до двох модулів на канал). У будь-якому випадку, ми отримаємо схожу продуктивність пам'яті, оскільки чотири ядра Bloomfield явно не відчували недостачі даних.

Серед інших додатків, про які варто згадати - підтримка інструкцій AES-NI, які забезпечують апаратне прискорення шифрування AES.

Розглянемо специфікацію процесорів Intel® Core™ i7-990X та Intel® Core™ i7-980X , Intel® Core™ i7-980 , Intel® Core™ i7-970 (таб.1.2)[2].

Таб.1.2. Специфікація шестиядерних процесорів сімейства Intel® Core™ i7.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Processor Number | i7-990X | i7-980X | i7-980 | i7-970 |
| # of Cores | 6 | 6 | 6 | 6 |
| # of Threads | 12 | 12 | 12 | 12 |
| Clock Speed | 3.46 GHz | 3.33 GHz | 3.33 GHz | 3.2 GHz |
| Max Turbo Frequency | 3.73 GHz | 3.6 GHz | 3.6 GHz | 3.46 GHz |
| Cache | 12 MB Intel® Smart Cache | 12 MB Intel® Smart Cache | 12 MB Intel® Smart Cache | 12 MB Intel® Smart Cache |
| Bus Type | QPI | QPI | QPI | QPI |
| System Bus | 6.4 GT/s | 6.4 GT/s | 4.8 GT/s | 4.8 GT/s |
| Instruction Set | 64-bit | 64-bit | 64-bit | 64-bit |
| Embedded Options Available | No | No | No | No |
| Lithography | 32 nm | 32 nm | 32 nm | 32 nm |
| Max TDP | 130 W | 130 W | 130 W | 130 W |
| Intel Turbo Boost Technology | Yes | Yes | Yes | Yes |
| Intel Hyper-Threading Technology | Yes | Yes | Yes | Yes |
| Intel Virtualization Technology | Yes | Yes | Yes | Yes |

Бачимо, що дані процесори виготовлені по 32нм технології з підтримкою 64-розрядних систем, технології Intel® Virtualization та функції Execute Disable Bit. 12 Мб Intel Smart Cache – загальної кеш-пам'яті третього рівня, що динамічно розподіляється між ядрами процесора залежно від навантаження, значно прискорюючи роботу та підвищуючи продуктивність.

Шестиядерні процесори Intel Core i7 мають рекордну продуктивність серед продукції компанії Intel та включають інноваційні процесорні технології:

* Технологія Intel® Turbo Boost максимально підвищує швидкість роботи ресурсномістких додатків, динамічно збільшуючи продуктивність у відповідності з навантаженням - більш висока продуктивність, коли це потрібно, і більш високе енергозбереження, коли збільшена продуктивність не потрібна[13].
* Технологія Intel® Hyper-Threading дозволяє багатопотокових додатків виконувати більше завдань паралельно. З 8 потоками, доступними операційній системі, багатозадачність стає ще проще.
* Технологія Intel® Smart Cache забезпечує високу продуктивність і ефективність кеш-пам'яті. Оптимізована для найсучасніших багатопоточних програм[14].
* Технологія Intel® QuickPath Interconnect розроблена для підвищення пропускної здатності і зниження часу затримки. Вона дозволяє досягти швидкості передачі даних до 25,6 ГБ / с з процесорами Extreme Edition.
* Вбудований контролер пам'яті підтримує три канали пам'яті DDR3, 1066МГц, завдяки чому пропускна здатність пам'яті досягає 25,6 ГБ / с. Низький час затримки і висока пропускна здатність контролера пам'яті забезпечують високу продуктивність додатків, що оперують великими обсягами даних.
* Технологія Intel® HD Boost значно підвищує продуктивність різноманітних мультимедійних і ресурсномістких додатків. 128-розрядні команди SSE запускаються по одній за тактовий цикл, дозволяючи досягти нового рівня ефективності з додатками, оптимізованими для набору команд SSE4.
* Технологія прискорення шифрування і дешифровки AES-NI включає 6 нових команд процесора, що підвищують продуктивність алгоритмів шифрування і дешифровки AES.

**Огляд та специфікація серверних шестиядерних процесорів сімейства Intel® Xeon®.**

У 2005 році Intel змінила свою стратегію розвитку настільних комп'ютерів, представивши першу двоядерні процесори Pentium. Зрозумівши, що битва гігагерц програна, і до рівня вище 10 ГГц прагнути навряд чи розумно, компанія змінила свою стратегію, націлившись на паралелізм[6].

Сьогодні ж ми можемо використовувати шестиядерні процесори з підтримкою Hyper-Threading, що дає нам 12 логічних процесорів для операційних систем.

Архітектурне перевага, яку AMD отримала завдяки використанню каналів HyperTransport, особливо сильно виявляло себе в многосокетних платформах, де Intel продовжувала спиратися на пропускну здатність шини FSB для обміну інформацією між процесорами. З лінійкою Xeon 5500 Intel, нарешті, вирішила цю проблему через інтерфейс QuickPath Interconnect, додавши підтримку технологій Hyper-Threading і Turbo Boost для подальшого поліпшення продуктивності в багатопотокових оточеннях і однопоточних додатках.

Звичайно, прогрес не стоїть на місці. У цьому році перехід на 32-нм техпроцес дав Intel прекрасну можливість збільшити складність своїх процесорів для малого і середнього бізнесу, не виходячи за межі теплового пакету. Зустрічайте лінійку Xeon 5600, яка оснащена до шести фізичних ядер на CPU і 12 Мбайт загального кешу L3 - і все це при збереженні такого ж теплового пакета 130 Вт, що й у випадку лінійки Xeon 5500.

Процесор Gulftown (шестиядерний процесор для настільних ПК) сумісний по інтерфейсу з процесорами Bloomfield (оригінальна лінійка Core i7-900). Точно так само процесори сімейства Xeon 5600 встановлюються в той же інтерфейс LGA 1366, що і Xeon 5500.

Ці процесори виготовлені для 64-розрядних однопроцесорних і двопроцесорних серверів і робочих станцій на базі багатоядерних процесорів для забезпечення передового рівня продуктивності і максимальної енергоефективності. Вони добре підходять для різноманітних інфраструктур, облачних, обчислювальних систем високої щільності і високопродуктивних обчислювальних систем.[3]

Розглянемо специфікацію шестиядерних процесорів Intel® Xeon® подана на таб.1.3.

Таб. 1.3. Специфікація шестиядерних процесорів сімейства Intel® Xeon®.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Processor Number | X5690 | X5680 | X5675 | X5670 | X5660 | X5650 | E5649 | E5645 |
| # of Cores | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| # of Threads | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| Clock Speed  (GHz) | 3.46 | 3.33 | 3.06 | 2.93 | 2.8 | 2.66 | 2.53 | 2.4 |
| Max Turbo  Frequency  (GHz) | 3.73 | 3.6 | 3.46 | 3.33 | 3.2 | 3.06 | 2.93 | 2.67 |
| Cache | 12 MB Smart Cache | 12 MB Smart Cache | 12 MB Smart Cache | 12 MB Smart Cache | 12 MB Intel® Smart Cache | 12 MB Intel® Smart Cache | 12 MB Intel® Smart Cache | 12 MB Intel® Smart Cache |
| System Bus  GT/s | 6.4 | 6.4 | 6.4 | 6.4 | 6.4 | 6.4 | 5.86 | 5.86 |
| Instruction Set | 64-bit | 64-bit | 64-bit | 64-bit | 64-bit | 64-bit | 64-bit | 64-bit |
| Lithography | 32 nm | 32 nm | 32 nm | 32 nm | 32 nm | 32 nm | 32 nm | 32 nm |
| Max TDP | 130 W | 130 W | 95 W | 95 W | 95 W | 95 W | 80 W | 80 W |
| Intel® Turbo Boost Technology | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Intel® Hyper-Threading Technology | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

Ці процесори виготовлені для 64-розрядних однопроцесорних і двопроцесорних серверів і робочих станцій на базі багатоядерних процесорів для забезпечення передового рівня продуктивності і максимальної енергоефективності. Вони добре підходять для різноманітних інфраструктур, облачних, обчислювальних систем високої щільності і високопродуктивних обчислювальних систем.

Процесори сімейства Intel® Xeon® виготовлені по 32нм технології з 12Мб Smart Cache, технології Intel ® Turbo Boost та Intel® Hyper-Threading.

* Інтелектуальні процесори, адаптуються до робочого навантаження, забезпечують до 15 разів більш високу продуктивність у порівнянні з серверами на базі одноядерних процесорів.
* Технологія Intel® Turbo Boost динамічно і автоматично збільшує до максимуму продуктивність серверних додатків за рахунок підвищення робочих частот ядер, що забезпечує прискорену обробку конкретних обчислювальних потоків і робочих навантажень з великою кількістю завдань.
* Технологія Intel® QuickPath надає масштабовану архітектуру загальної пам'яті з високошвидкісними міжпроцесорних каналами типу «точка-точка», а також збільшений об'єм кеш-пам'яті і системної пам'яті в двопроцесорних серверах.
* Великий обсяг пам'яті - до 18 роз'ємів DIMM і підтримка до 288 ГБ системної пам'яті забезпечують більш високу продуктивність додатків з інтенсивним використанням даних.

### Огляд сучасних шестиядерних процесорів AMD.

Сьогодні компанія AMD виготовляє шестиядерні процесори у сімействі процесорів для настільних ПК AMD Phenom™ II та .

**Огляд та специфікація шестиядерних процесорів сімейства AMD Phenom™ II.**

AMD Phenom II X6 1090T виконаний по техпроцесу 45 нм, містить шість обчислювальних ядер, по 128 Кб і 512 Кб кеш-пам'яті першого і другого рівня на кожне ядро, відповідно. Також є загальна для всіх ядер кеш-пам'ять третього рівня об'ємом 6 Мб, як і у чотирьохядерних попередників[4].

Специфікація процесорів AMD Phenom™ II X6 Black приведена на таб.1.4.[7]

Таб.1.4. Специфікація процесорів AMD Phenom™ II X6 Black.

|  |  |  |  |
| --- | --- | --- | --- |
| Processor Type: | [AMD Phenom™ II X6 Black](http://shop.amd.com/us/All/LinesPerFamily/Desktop/Processor?Family=phenom&Line=phenom%2Fphenomiix6black) | Processor Type: | [AMD Phenom™ II X6 Black](http://shop.amd.com/us/All/LinesPerFamily/Desktop/Processor?Family=phenom&Line=phenom%2Fphenomiix6black) |
| Model: | [1090T](http://shop.amd.com/us/All/ModelsPerLine/Desktop/Processor?Line=phenom%2Fphenomiix6black&Model=phenom%2Fphenomiix6black%2F1090t) | Model: | [1100T](http://shop.amd.com/us/All/ModelsPerLine/Desktop/Processor?Line=phenom%2Fphenomiix6black&Model=phenom%2Fphenomiix6black%2F1100t) |
| Part Number: | HDT90ZFBGRBOX | Part Number: | HDE00ZFBGRBOX |
| Socket Type: | AM3 | Socket Type: | AM3 |
| Power Wattage: | 125 | Power Wattage: | 125 |
| Frequency: | 3200 | Frequency: | 3300 |
| Processor Bus Speed: | 2000 | Processor Bus Speed: | 4000 |
| Processor L2 Cache Size: | 3072 | Processor L2 Cache Size: | 3072 |
| CMOS: | 45nm SOI | CMOS: | 45nm SOI |
| Stepping: | E0 | Stepping: | E0 |
| Fusion Control Hubs: | Not Specified | Fusion Control Hubs: | Not Specified |
| Direct X Version: | Not Specified | Direct X Version: | Not Specified |
| GPU Clock Speed: | Not Specified | GPU Clock Speed: | Not Specified |

На сьогоднішній день нові шестиядерні процесори AMD Phenom ™ II X6, побудовані на технології Vision Black, найшвидші процесори компанії AMD. Це також єдині в світі шестиядерні процесори, що закупівельна ціна яких не перевищує $ 200. В центрі платформи технології Vision Black, передових процесорів AMD Phenom II X6 є технологія AMD Turbo, що підлаштовує процесор до динамічно мінливих потреб користувачів.

Підтримують наступні технології та має такі переваги:

* технологія AMD64 з архітектурою Direct Connect:
  + підвищує продуктивність системи за допомогою прямого підключення процесорів, контролерів пам'яті і інтерфейсу введення-виведення процесора;
* Інтегрований контролер пам'яті:
  + підвищення продуктивності додатків завдяки значному скороченню часу відгуку пам'яті;
* відповідність продуктивності і пропускної здатності пам'яті обчислювальним потребам;
* технологія HyperTransport:
  + забезпечує можливість виділення пікової пропускної спроможності каналу (до 16,0 ГБ / с) на кожен процесор, скорочуючи кількість «вузьких місць» в підсистемі введення-виведення даних;
  + загальна пропускна здатність каналу зв'язку процесора з системою - до 37 ГБ / с (шина HyperTransport і шина пам'яті)[15];
* технологія AMD Balanced Smart Cache:
  + загальний кеш третього рівня (6 МБ або 4 МБ);
  + кеш другого рівня 512 КБ на ядро;

ПЕРЕВАГА: підвищення продуктивності за рахунок скорочення часу доступу до часто використовуваних даних.

* Технологія AMD Wide Floating Point Accelerator:
  + 128-розрядний модуль для виконання операцій з плаваючою комою;
  + окремий високопродуктивний модуль для виконання операцій з плаваючою комою (128-розрядний внутрішній канал передачі даних) для кожного ядра;

ПЕРЕВАГА: підвищення продуктивності завдяки розширеному каналу передачі даних і прискореного виконання операцій з плаваючою комою.

* Технологія HyperTransport ™:
  + один 16-розрядний канал до 4000 млн передач в секунду;
  + Пропускна здатність каналу введення-виведення HyperTransport ™ - до 8 Гб / с, у режимі HyperTransport Generation 3.0 - до 16 Гб / с.
  + Загальна пропускна здатність каналу зв'язку процесора з системою - до 37 ГБ / с (шина HyperTransport і шина пам'яті).

ПЕРЕВАГА: підвищення продуктивності завдяки скороченню часу доступу до системного інтерфейсу введення-виведення.

* Вбудований контролер DRAM з технологією AMD Memory Optimizer.
  + широкосмуговий вбудований контроллер пам'яті з малим часом відгуку.
  + Підтримка модулів небуферізованной пам'яті SDRAM DIMM: PC2-8500 (DDR2-1066); PC2-6400 (DDR2-800), PC2-5300 (DDR2-667), PC2-4200 (DDR2-533) і PC2-3200 (DDR2-400 ) - АМ2 +.
  + Підтримка безрегістровой пам'яті DIMM до PC2 8500 (DDR2-1066 МГц) і PC3 10600 (DDR3-1333 МГц) - АМ3.
  + Пропускна здатність пам'яті DDR2 - 17,1 ГБ / с, пропускна здатність пам'яті DDR3 - до 21 ГБ / с.

ПЕРЕВАГА: підвищення продуктивності за рахунок швидкого доступу до системної пам'яті.

* Технологія AMD Virtualization ™ (AMD-V ™) з функцією Rapid Virtualization Indexing
  + поліпшення кремнієвих компонентів спрямоване на підвищення продуктивності, а також рівня надійності і безпеки існуючих та майбутніх середовищ віртуалізації за рахунок надання віртуалізованих прикладних прямого швидкого доступу до виділеної пам'яті.

ПЕРЕВАГА: підвищення рівня безпеки та ефективності роботи програмного забезпечення для віртуалізації завдяки поліпшенню їх взаємодії з віртуальними системами.

* Технологія Cool'n'Quiet ™ 3.0:
* поліпшені засоби управління енергоспоживанням з автоматичною миттєвої регулюванням рівнів продуктивності і функцій відповідно до вимог до продуктивності процесора;
* безшумна робота і знижені вимоги до електроживлення;

ПЕРЕВАГА: можливість створення високопродуктивних систем з низьким рівнем енергоспоживання, тепловиділення і шуму.

* Технологія AMD CoolCore ™:
* зниження енергоспоживання за рахунок відключення невикористовуваних блоків процесора. Наприклад, контролер пам'яті може відключити ланцюг запису при читанні з пам'яті, що дозволить знизити рівень загального споживання електроенергії системою;
* автоматична робота без встановлення додаткових драйверів або включення функції в BIOS;
* живлення може включатися і виключатися протягом одного такту, забезпечуючи економію електроенергії без зниження продуктивності;

ПЕРЕВАГА: підвищення продуктивності за допомогою динамічного включення і виключення компонентів процесора.

* Технологія Dual Dynamic Power Management ™:
* більш диференційоване управління живленням дозволяє знизити енергоспоживання процесора;
* завдяки окремим рівням живлення ядер і контролера пам'яті досягається оптимальний рівень енергоспоживання та продуктивності, що призводить до економії електроенергії.

ПЕРЕВАГА: підвищення ефективності платформи за рахунок забезпечення роботи пам'яті на вимогу при збереженні низького рівня енергоспоживання системи.

### Тести на продуктивність.

Для об’єктивного порівняння процесорів Intel і AMD представимо розгорнуті тести кращих процесорів обох компаній[8].

**Сімейство Intel Xeon.Архітектура Nehalem**

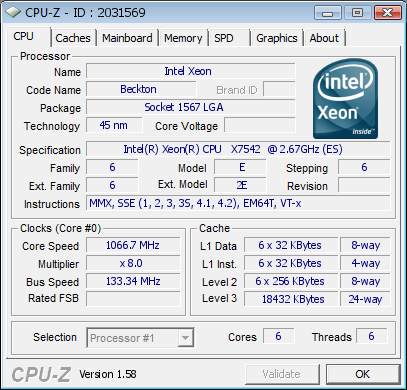
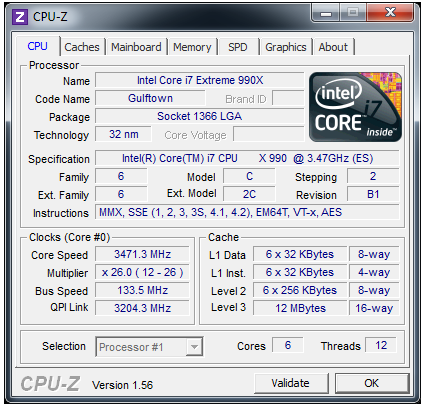


Рис.1.1. Результат тестування шестиядерних процесорів Intel Xeon X7542

**Сімейство Intel Core i7.**



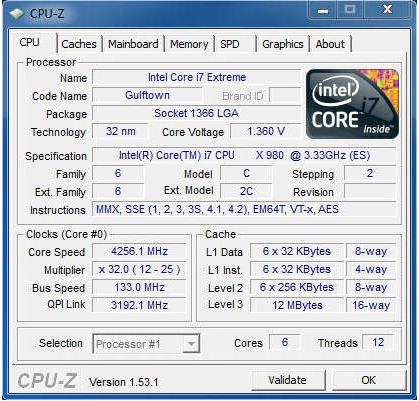
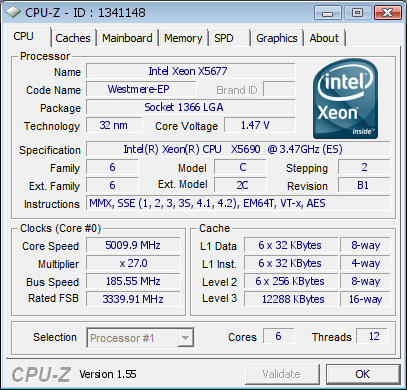


Рис.1.2. Результати тестування шестиядерних процесорів Intel Core i7-990х та Intel Core i7-980х.

**Сімейство Intel Xeon. Архітектура Westmere**

З усієї лінійки процесорів Intel Xeon найпотужніші шестиядерні процесори Intel Xeon Х5690 та Intel Xeon Х5680. Результати їх тестування наведено на рис.1.3.[9]



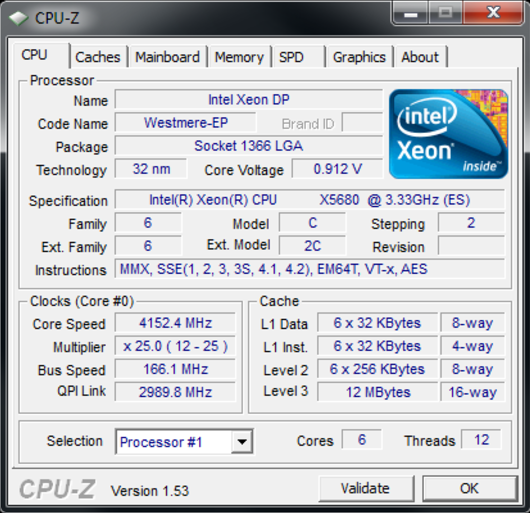
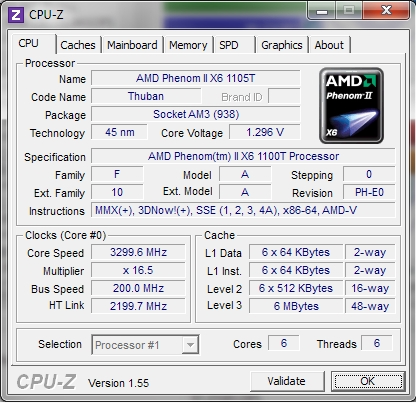


Рис.1.3. Результати тестування процесорів Intel Xeon Х5690 та Intel Xeon Х5680.

**Сімейство AMD Phenom™ II.**

З усієї лінійки процесорів Intel Xeon найпотужніші шестиядерні процесори Intel Xeon Х5690 та Intel Xeon Х5680. Результати їх тестування наведено на рис.1.4.[10]



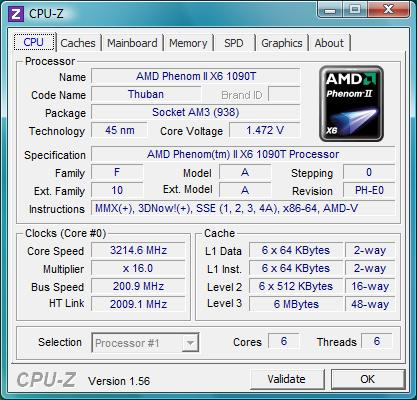


Рис.1.4. Результати тестування процесорів AMD Phenom™ II 1100Т та AMD Phenom™ II 1090Т.

### Висновки до розділу 1

1.Зроблено аналіз основних характеристики та переваги кожного сімейства процесорів. При цьому вияснилося, що в арсеналі компанії Intel є дещо потужніші та продуктивніші процесори з широким спектром підтримуваних технологій. Так за допомогою технології Intel® Turbo Boost процесори лінійки Intel Core i7 досягають тактової частоти 3,8 ГГц, недоступної для процесорів аналогічного спрямування компанії AMD - AMD Phenom™ II. Так як обидва сімейства (Intel Core i7 та AMD Phenom™ II) виготовлені для настільних ПК, вони мають вбудовану підтримку графіки: у сімействі Intel Core i7 за це відповідає технологія Intel® Virtualization (Intel® VT), у сімействі AMD Phenom™ II - AMD Virtualization ™ (AMD-V ™) з функцією Rapid Virtualization Indexing.

2.На основі аналізу пам’яті, було показано, що процесори обох компаній використовують усі три види кеша, причому третій рівень динамічно роз приділяється між процесорами по потребі за допомогою технологій Intel® Smart Cache та AMD Balanced Smart Cache. Суттєва різниця полягає лише в об’ємі кеш пам’яті: Intel Core i7 мають менше кеш пам’яті першого рівня(32Кб проти 64Кб процесорів AMD Phenom™ II), але більше кеш пам’яті третього рівня – 12Мб проти 6Мб.

3.Проаналізувавши арсенал випуску шестиядерних процесорів сімейства Intel Xeon не мають шестиядерних аналогів в продукції AMD, тому конкурують на ринку з чотирьох- та восьмиядерними AMD Opteron™. Як будь-які серверні процесори вони (процесори Intel Xeon) оптимізовані для виконання великої кількості обчислень. Підтримують, крім звичайних для сучасних процесорів Intel технологій Технологія Intel® Turbo Boost та Intel® Smart Cache, технологію Intel® QuickPath, що надає масштабовану архітектуру загальної пам'яті з високошвидкісними міжпроцесорних каналами типу «точка-точка», а також мають збільшений об'єм кеш-пам'яті і системної пам'яті в двопроцесорних серверах. Великий обсяг пам'яті - до 18 роз'ємів DIMM і підтримка до 288 ГБ системної пам'яті забезпечують більш високу продуктивність додатків з інтенсивним використанням даних.

4.Розглянувши використання електроенергії перевагами процесорів AMD Phenom™ II є більш економне використання її за допомогою технологій Dual Dynamic Power Management, AMD CoolCore та Cool'n'Quiet 3.0. Хоча в специфікації до продуктів вказана однакова швидкість обміну з пам’яттю, на тестах побачили, що за допомогою технології HyperTransport ™ частота системної шини процесорів AMD близька до 200МГц, порівняно з близько 180МГц продуктів компанії Intel. А також незаперечною перевагою є ціна, яка в Україні для AMD Phenom™ II становить близько $ 200, проти близько $ 1100 за Intel Core i7-990Х та Intel Core i7-980Х чи до $ 1750 за Intel Six-Core Xeon X5680.

# РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП



### 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\*MO)\*(MZ \* MK\*α)наступним чином:

C:\Users\stas\Desktop\курсач про\Безымянный.png

C:\Users\stas\Desktop\курсач про\й.png

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:
2. 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 .

### Розробка паралельного математичного алгоритму.

Вираз математичного завдання може бути обчислений в 2 етапи:

1. XH=BH+C\*MOH
2. АH = X\*(MZ\*MKH \*α)

Загальні ресурси: X,C, MZ, α.

### Розробка алгоритмів процесів

Задача Т1:

1. Введення B,C
2. Сигнал про завершення вводу задачам T2..T6
3. Очікувати сигнал про завершення введення в задачах T3,T6
4. Копіювати C1=C, MZ1=MZ, α1=α
5. Рахунок XH=BH+C1\*MOH
6. Сигнал задачам T2..T6 про завершення рахунку XH
7. Очікувати завершення рахунку в задачах T2..T6
8. Копіювати X1=X
9. Рахунок АH = X1\*(MZ1\*MKH \*α1)
10. Очікувати завершення рахунку в задачах T2..T6
11. Вивід А

Задача Т2:

1. Очікувати сигнал про завершення введення в задачах T1, T3,T6
2. Копіювати C2=C, MZ2=MZ, α2=α
3. Рахунок XH=BH+C2\*MOH
4. Сигнал задачам T2..T6 про завершення рахунку XH
5. Очікувати завершення рахунку в задачах T2..T6
6. Копіювати X2=X
7. Рахунок АH = X2\*(MZ2\*MKH \*α2)
8. Сигнал задачі T1 про завершення рахунку АH

Задача Т3:

1. Введення MO,MZ
2. Сигнал про завершення вводу задачам T1, T2, T4..T6
3. Очікувати сигнал про завершення введення в задачах T1,T6
4. Копіювати C3=C, MZ3=MZ, α3=α
5. Рахунок XH=BH+C3\*MOH
6. Сигнал задачам T2..T6 про завершення рахунку XH
7. Очікувати завершення рахунку в задачах T2..T6
8. Копіювати X3=X
9. Рахунок АH = X3\*(MZ3\*MKH \*α3)
10. Сигнал задачі T1 про завершення рахунку АH

Задача Т4:

1. Очікувати сигнал про завершення введення в задачах T1, T3,T6
2. Копіювати C4=C, MZ4=MZ, α4=α
3. Рахунок XH=BH+C4\*MOH
4. Сигнал задачам T2..T6 про завершення рахунку XH
5. Очікувати завершення рахунку в задачах T2..T6
6. Копіювати X4=X
7. Рахунок АH = X4\*(MZ4\*MKH \*α4)
8. Сигнал задачі T1 про завершення рахунку АH

Задача Т5:

1. Очікувати сигнал про завершення введення в задачах T1, T3,T6
2. Копіювати C5=C, MZ5=MZ, α5=α
3. Рахунок XH=BH+C5\*MOH
4. Сигнал задачам T2..T6 про завершення рахунку XH
5. Очікувати завершення рахунку в задачах T2..T6
6. Копіювати X5=X
7. Рахунок АH = X5\*(MZ5\*MKH \*α5)
8. Сигнал задачі T1 про завершення рахунку АH

Задача Т6:

1. Введення MK,α
2. Сигнал про завершення вводу задачам T1..T5
3. Очікувати сигнал про завершення введення в задачах T1,T3
4. Копіювати C6=C, MZ6=MZ, α6=α
5. Рахунок XH=BH+C6\*MOH
6. Сигнал задачам T2..T6 про завершення рахунку XH
7. Очікувати завершення рахунку в задачах T2..T6
8. Копіювати X6=X
9. Рахунок АH = X6\*(MZ6\*MKH \*α6)
10. Сигнал задачі T1 про завершення рахунку АH

Схема алгоритму для кожного процесора ПРГ1 знаходиться в Додатку А.

### Розробка схеми взаємодії процесів

Структурна схема взаємодії процесів зображена на рис. 2.2.

### Розробка програми ПРГ1

ПЗ для ПКС зі спільною пам'яттю реалізоване на мові програмування Ada з використанням захищених модулів та складається з наступних модулів:

1. Модуль KP\_PRO1.adb містить реалізацію алгоритмів основної програми та кожного з потоків. Потоки створюються за допомогою інструментів мови програмування Ada. Механізми синхронізації , який використовується, описується за допомогою захищеного модулю(Monitor) .

Для вирішення задачі взаємного виключення на основі схеми взаємодії процесів розроблено модуль Monitor з такими процедурами:

procedure Set\_C(V: in Vector);

procedure Set\_MZ(MM: in Matrix);

procedure Set\_L(InL: in Integer);

procedure Set\_X(V: in Vector; from: in integer; to: in integer);

Для вирішення задачі синхронізації на основі схеми взаємодії процесів розроблено Monitor з такими синхронізованими процедурами і входами:

procedure Inc\_FT1;

procedure Inc\_FT3;

procedure Inc\_FT6;

procedure Inc\_FX;

procedure Inc\_FA;

entry Wait\_Input\_T1;

entry Wait\_Input\_T3;

entry Wait\_Input\_T6;

entry Wait\_X;

entry Wait\_A;

Для копіювання даних на основі схеми взаємодії процесів розроблено слідуючі функції:

function Copy\_X return Vector;

function Copy\_C return Vector;

function Copy\_L return Integer;

function Copy\_MZ return Matrix;

Лістинг програми наведений в Додатку В.

### Тестування програми ПРГ1

Тестування проводилося на системі з характеристиками:

Центральний процесор: AMD Phenom I I X6 2,80 ГГц;

Оперативна пам’ять: 4 Гб

Операційна система: 6.1.7601 SP1

Значення часу Т1-Т6 для різних N відображаються в таблиці 2.1.

Таблиця 2.1. Час виконання програми в ПКС зі СП

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N / P | 1 | 2 | 3 | 4 | 5 | 6 |
| 800 | 10,55 | 5,351 | 3,751 | 3,366 | 2,966 | 2,086 |
| 1400 | 67,45 | 34,122 | 23,39 | 19,5 | 18,752 | 13,11 |
| 2200 | 265,678 | 134,817 | 93,271 | 83,792 | 74,672 | 52,217 |

На підставі значень часу Т1-Т6 виконуються розрахунки коефіцієнтів прискорення Кпn = Т1/Тn, які відображаються в таблиці 2.2.

Таблиця 2.2. Значення коефіцієнту прискорення для ПКС зі СП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N / P | 2 | 3 | 4 | 5 | 6 |
| 600 | 1,971594 | 2,812583 | 3,134284 | 3,556979 | 5,057526 |
| 1200 | 1,976731 | 2,883711 | 3,458974 | 3,59695 | 5,144928 |
| 1800 | 1,970657 | 2,848452 | 3,170685 | 3,557933 | 5,08796 |

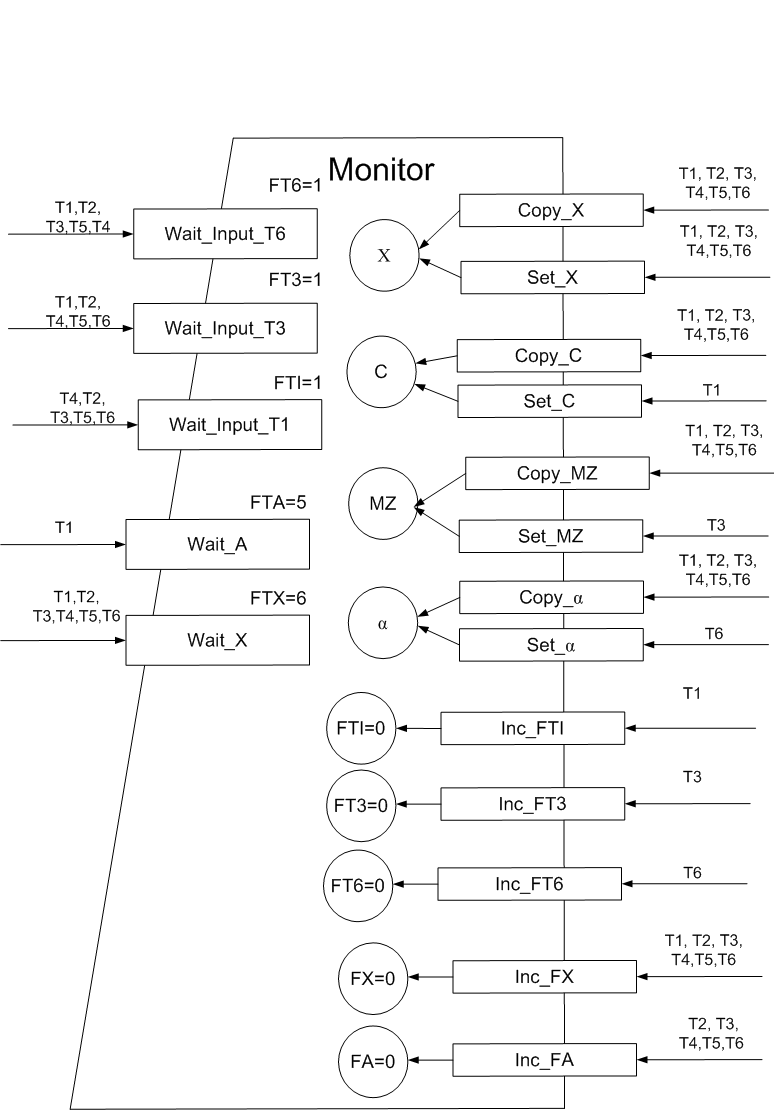


Рис.2.2. Структурна схема взаємодії процесів

Графіки зміни Кп в залежності від значень Р = (1 - 6) для різних N зображені на рис. 2.3 - рис. 2.5.

Рис. 2.3. Залежність коефіцієнту прискорення від кількості ядер для ПКС з спільною пам’яттю при N = 800

Рис. 2.4. Залежність коефіцієнту прискорення від кількості ядер для ПКС з спільною пам’яттю при N = 1400

Рис. 2.5. Залежність коефіцієнту прискорення від кількості ядер для ПКС з спільною пам’яттю при N = 2200

Виконаємо розрахунки коефіцієнтів ефективності Кеn = Т1 / (Р \* Тn), які відображаються у таблиці 2.3.

Таблиця 2.3.Значення коефіцієнту ефективності для ПКС з СП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N / P | 2 | 3 | 4 | 5 | 6 |
| 800 | 98,5797 | 93,75278 | 78,3571 | 71,13958 | 84,29211 |
| 1400 | 98,83653 | 96,1237 | 86,47436 | 71,93899 | 85,74879 |
| 2200 | 98,53283 | 94,94841 | 79,26711 | 71,15867 | 84,79933 |

Графіки зміни Ке в залежності від значень Р = (1 - 6) для різних N зображені на рис. 2.6.

Рис. 2.6. Залежність коефіцієнту ефективності від кількості ядер для ПКС з спільною пам’яттю

### Висновки до розділу 2

1. Реалізація взаємодії процесів в програмі для ПКС з СП виконувалась за допомогою засобів мови програмування Ada. Використання захищеного модуля дозволило ефективно вирішити задачі синхронізації, при очікуванні сигналу від багатьох процесів, а також використання процедур при зверненні до даних вирішило проблему задачі взаємовиключення.

2. Тестування програми для ПКС зі спільною пам'яттю показало, що коефіцієнт прискорення лежить в діапазоні від 1.98 до 5,14; максимальний коефіцієнт прискорення отриманий при P = 6, N = 1400; коефіцієнт ефективності лежить в діапазоні від 0.71 до 0,99; мінімальний коефіцієнт ефективності отриманий при P =5, N = 800 та P = 5, N = 2200.

# РОЗДІЛ 3.РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП



### Розробка паралельного математичного алгоритму.

Вираз математичного завдання може бути обчислений в 2 етапи:

1.XH=BH+C\*MOH

2.АH = X\*(MZ\*MKH \*α).

### Розробка алгоритмів процесів

Задача Т1:

1. Введення B,C,MO
2. Переслати задачі T2 C, B4h, MO4h
3. Прийняти від задачі T2 MZ, MK4h, α
4. Переслати T4 C, MZ, α, Bh, MOh, MKh
5. Рахунок Xh=Bh+C\*MOh
6. Прийняти від T4 Xh
7. Переслати T2 X2h
8. Прийняти від T2 X4h
9. Переслати T4 X4h
10. Рахунок Ah
11. Прийняти від T4 Ah
12. Прийняти від T2 A4h
13. Вивід А

Задача Т2:

1. Прийняти від задачі T1 C, B4h, MO4h
2. Прийняти від задачі T3 MZ, MK4h, α
3. Переслати T3 C, B2h, MO2h
4. Переслати T1 MZ, α, MK2h
5. Переслати T5 C, MZ, α, Bh, MOh, MKh
6. Рахунок Xh=Bh+C\*MOh
7. Прийняти від T5 Xh
8. Прийняти від T3 X2h
9. Прийняти від T1 X2h
10. Переслати T1 X4h
11. Переслати T3 X4h
12. Переслати T5 X5h
13. Рахунок Ah
14. Прийняти від T5 Аh
15. Прийняти від T3 А2h
16. Переслати T1 А4h

Задача Т3:

1. Введення MZ, MK, α
2. Переслати задачі T2 MZ, MK4h, α
3. Прийняти від задачі T2 C, B2h, MO2h
4. Переслати T6 C, MZ, α, Bh, MOh, MKh
5. Рахунок Xh=Bh+C\*MOh
6. Прийняти від T6 Xh
7. Переслати T2 X2h
8. Прийняти від T2 X4h
9. Переслати T6 X5h
10. Рахунок Ah
11. Прийняти від T6 Ah
12. Переслати T2 A2h

Задача Т4:

1. Прийняти від задачі T1 C, MZ, α, Bh, MOh, MKh
2. Рахунок Xh=Bh+C\*MOh
3. Переслати задачі T1 Xh
4. Прийняти від задачі T1 X5h
5. Рахунок Ah
6. Переслати задачі T1 Ah

Задача Т5:

1. Прийняти від задачі T2 C, MZ, α, Bh, MOh, MKh
2. Рахунок Xh=Bh+C\*MOh
3. Переслати задачі T2 Xh
4. Прийняти від задачі T2 X5h
5. Рахунок Ah
6. Переслати задачі T2 Ah

Задача Т6:

1. Прийняти від задачі T3 C, MZ, α, Bh, MOh, MKh
2. Рахунок Xh=Bh+C\*MOh
3. Переслати задачі T3 Xh
4. Прийняти від задачі T3 X5h
5. Рахунок Ah
6. Переслати задачі T3 Ah

Схема алгоритму для кожного процесора ПРГ2 знаходиться в Додатку Б.

### Розробка схеми взаємодії процесів

Структурна схема взаємодії процесів зображена на рис. 3.1.

### Розробка програми ПРГ2

ПЗ для ПКС з локальною пам'яттю реалізоване на мові програмування Ada з використанням концепції посилки повідомлень та складається з наступних модулів:

1. Модулю KP\_PRO2.adb містить реалізацію алгоритмів основної програми та кожного з процесів. Для реалізації посилки повідомлень використовувалися наступні функції концепції посилки повідомлень в мові програмування Ada: entry.

Задача Т1 має такі оператори входу:

SendT2\_T1\_Data – приймає MZ, α, MKh, MK2h від Т2

SendT4\_T1\_X– приймає Xh від Т4

SendT2\_T1\_X– приймає X4h від Т2

SendT4\_T1\_A– приймає Ah від Т4

SendT2\_T1\_A - приймає A2h від Т2

Задача Т2 має такі оператори входу:

SendT1\_T2\_Data– приймає C, B4h, MO4h від Т1

SendT3\_T2\_Data– приймає MZ, α, MK4h від Т3

SendT5\_T2\_X– приймає Xh від T5

SendT3\_T2\_X– приймає X2h від Т3

SendT1\_T2\_X – приймає X2h від Т1

SendT5\_T2\_A – приймає Ah від Т5

SendT3\_T2\_A - приймає A2h від Т3

Задача Т3 має такі оператори входу:

SendT2\_T3\_Data– приймає C, B2h, MO2h від Т2

SendT6\_T3\_X– приймає Xh від Т6

SendT2\_T3\_X– приймає X4h від Т2

SendT4\_T3\_A– приймає Ah від Т4

SendT6\_T3\_A - приймає Ah від Т6

Задача Т4 має такі оператори входу:

SandT1\_T4\_Data– приймає C,MZ, α, Bh, MOh від Т1

SandT1\_T4\_X– приймає A5h від Т1

Задача Т5 має такі оператори входу:

SandT2\_T5\_Data– приймає C,MZ, α, Bh, MOh від Т2

SandT2\_T5\_X– приймає A5h від Т2

Задача Т6 має такі оператори входу:

SandT3\_T6\_Data– приймає C,MZ, α, Bh, MOh від Т3

SandT3\_T6\_X– приймає A5h від Т3

Лістинг програми наведений в Додатку Г.



Рис.3.1. Структурна схема взаємодії процесів

### Тестування програми ПРГ2

Тестування проводилося на системі з характеристиками:

Центральний процесор: Intel core 2 quad 6600 2,4Ghz

Оперативна пам’ять: 2

Операційна система: версія 6.1.7601 sp1

Версії програмного забезпечення:

Значення часу Т1-Т4 для різних N відображаються в таблиці 3.1.

Таблиця 3.1. Час виконання програми в ПКС зі СП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N / P | 1 | 2 | 3 | 4 |
| 800 | 10,431 | 5,43 | 3,605 | 2,81 |
| 1400 | 55,811 | 28,153 | 19,201 | 16,315 |
| 2200 | 216,595 | 112,601 | 74,317 | 63,528 |

На підставі значень часу Т1-Т6 виконуються розрахунки коефіцієнтів прискорення Кпn = Т1/Тn, які відображаються в таблиці 3.2.

Таблиця 3.2. Значення коефіцієнту прискорення для ПКС зі СП

|  |  |  |  |
| --- | --- | --- | --- |
| N / P | 2 | 3 | 4 |
| 800 | 1,920994 | 2,893481 | 3,7121 |
| 1400 | 1,982418 | 2,906672 | 3,42084 |
| 2200 | 1,923562 | 2,914474 | 3,409442 |

Графіки зміни Кп в залежності від значень Р = (1 - 4) для різних N зображені на рис. 3.2 - рис. 3.4.

Рис. 3.2. Залежність коефіцієнту прискорення від кількості ядер для ПКС з спільною пам’яттю при N = 800

Рис. 3.3. Залежність коефіцієнту прискорення від кількості ядер для ПКС з спільною пам’яттю при N = 1400

Рис. 3.4. Залежність коефіцієнту прискорення від кількості ядер для ПКС з спільною пам’яттю при N = 2200

Виконаємо розрахунки коефіцієнтів ефективності Кеn = Т1 / (Р \* Тn), які відображаються у таблиці 3.3.

Таблиця 3.3.Значення коефіцієнту ефективності для ПКС з СП

|  |  |  |  |
| --- | --- | --- | --- |
| N / P | 2 | 3 | 4 |
| 800 | 96,04972 | 96,44938 | 92,80249 |
| 1400 | 99,12088 | 96,88905 | 85,52099 |
| 2200 | 96,1781 | 97,14915 | 85,23604 |

Графіки зміни Ке в залежності від значень Р = (1 - 4) для різних N зображені на рис. 3.5.

Рис. 3.5. Залежність коефіцієнту ефективності від кількості ядер для ПКС з спільною пам’яттю

### Висновки до розділу 3

1. Реалізація організації рішення задачі взаємодії в програмі для ПКС з ЛП виконувалась за допомогою концепції посилки повідомлень Мови програмування Ada. Направленість концепції на пересилку повідомлень дозволила ефективно обмінюватися даними між процесами та не втратити багато часу на пересилку.

2. Тестування програми для ПКС з локальною пам'яттю показало, що коефіцієнт прискорення лежить в діапазоні від 1.92 до 3.7; максимальний коефіцієнт прискорення отриманий при P = 4, N = 800; коефіцієнт ефективності лежить в діапазоні від 0.85 до 0.99; мінімальний коефіцієнт ефективності отриманий при P = 4, N = 2200 та P = 4, N = 1400.

# ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ДО РОБОТИ

1. В цілому на ринку шестиядерних процесорів лідирує переважно продукція компанії Intel. Ці процесори потужніші та підтримують набагато більший спектр технологій. Проте суттєва різниця полягає в ціні. Процесори сімейства AMD Phenom™ II набагато дешевші, а тому для настільних ПК - це хороший варіант за оптимальну ціну. Процесори ж сімейства Intel Сore I7 – використовують важливі та корисні технології, що реалізують можливість розпаралелювання задач в кожному з ядер та мають вбудовану підтримку графіки на високому рівні. Ці процесори оптимально використовувати для настільних ПК для ігор та програм з використанням складних графічних модулів.

2. Внутрішня будова процесорів Intel та AMD також значно відрізняється. Основною і найбільш значною відмінністю є використання різної технології виготовлення кристала – 32нм в Intel та 45нм в AMD.

3. Внутрішня пам’ять розглядуваних процесорів також відрізняється. Проглядається різний підхід до організації кеш-пам’яті: процесори Intel мають більше кеш-пам’яті третього рівня, але разом з тим менше кеш-пам’яті першого та другого рівнів.

4. Максимальний час виконання ПЗ для ПКС зі спільною пам'яттю склав 265,678 с. Для ПКС з локальною пам'яттю – 216,595 с. Таким чином ПЗ для ПКС зі СЛ виявилось швидшим. Це пояснюється тим, що багато часу займає запис спільних даних, оскільки задачі можуть записувати їх тільки по черзі.

5. За результатами тестування можемо стверджувати, що в системах з СП та ж задача виконується довше: через необхідність запису спільних даних кожною задачами, який відбувається по черзі, кожною задачею. За результатами тестування було показано, що ПЗ для ПКС зі локальною пам'яттю виконується швидше, ніж ПЗ для ПКС з спільною пам'яттю.

# СПИСОК СКОРОЧЕНЬ

ЗВВ - задача взаємного виключення.

КНП - концепція необмеженого паралелізму.

КР - курсова робота.

ЛП - локальна пам'ять.

МПКС - масштабована паралельна комп'ютерна система.

ПЗ - програмне забезпечення.

СП - спільна пам'ять.

ЯПФ - ярусно-паралельна форма.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Процессоры Intel® Core™ i7 – Intel – URL: http://www.intel.com/ru\_RU/ products/processor/corei7/index.htm

2. Сімейство процессорів Intel® Core™ i7 Extreme Edition на офіційному сайті компанії Intel. - URL: http://www.intel.com/cd/products/services/emea/rus /processors/corei7ee/overview/406058.htm.

3.Сімейство процессорів Intel® Xeon® серии 5000. - URL: http://www.intel.com/ru\_RU/products/server/processor/xeon5000/index.htm.

4. Сімейство процессорів AMD Phenom™ II X6. - URL: http://products.amd. com/en-us/desktopcpuresult.aspx?f1=AMD+Phenom%e2%84%a2+II+X6.

5. Техническое описание процессора Intel® Core™ i7-900 Extreme Edition для настольных ПК на базе 32-нанометровой производственной технологии - URL: http://download.intel.com/design/processor/datashts/323252.pdf.

6. Краткое описание процессоров Intel® Xeon® серии 5600 - URL: http://www.intel.com/Assets/en\_US/PDF/prodbrief/323501.pdf.

7. Специфікація процесора AMD Phenom II X6 1055T - URL: http://products.amd.com/en-us/DesktopCPUDetail.aspx

8. Тести продуктивності програмою CPU-Z процесорів сімейства Intel Core I7 - URL: http://forum.antichat.ru/thread270548.html

9. Тести продуктивності програмою CPU-Z процесорів сімейства Intel Xeon серії 5600 - URL: http://www.thg.ru/cpu/intel\_xeon\_x5680/intel\_xeon\_x5680-03.html

10. Тести продуктивності програмою CPU-Z процесорів сімейства AMD Phenom II X6 - URL: http://www.pcgameshardware.com/aid,746125/AMD-Phenom-II-X6-1090T-and-1055T-Six-cores-on-a-budget/Reviews/

11. Жуков І. Корочікн О. Паралельні та розподілені обчислення. - Київ, Корнійчук, 2005. - 226 c.

12. SDK для процессорів сімейства Intel® Core i7 - URL: http://software.intel.com/ru-ru/articles/intel-next-generation-intel-core-i7-processor-family-sdk/

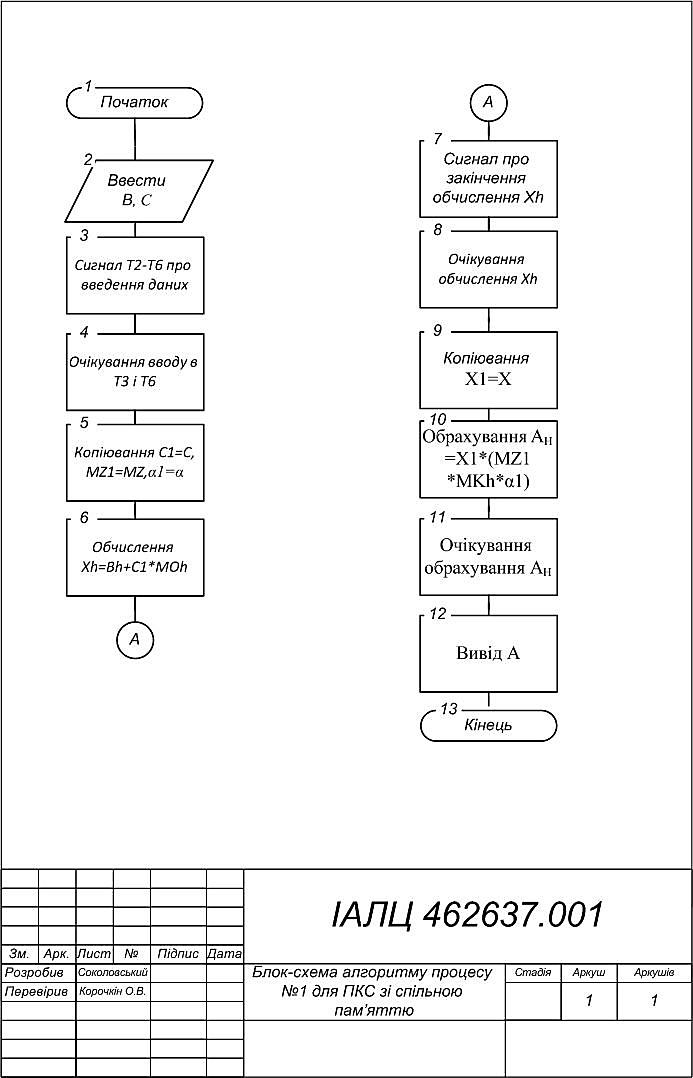
13. Intel® Turbo Boost Technology - URL: http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html

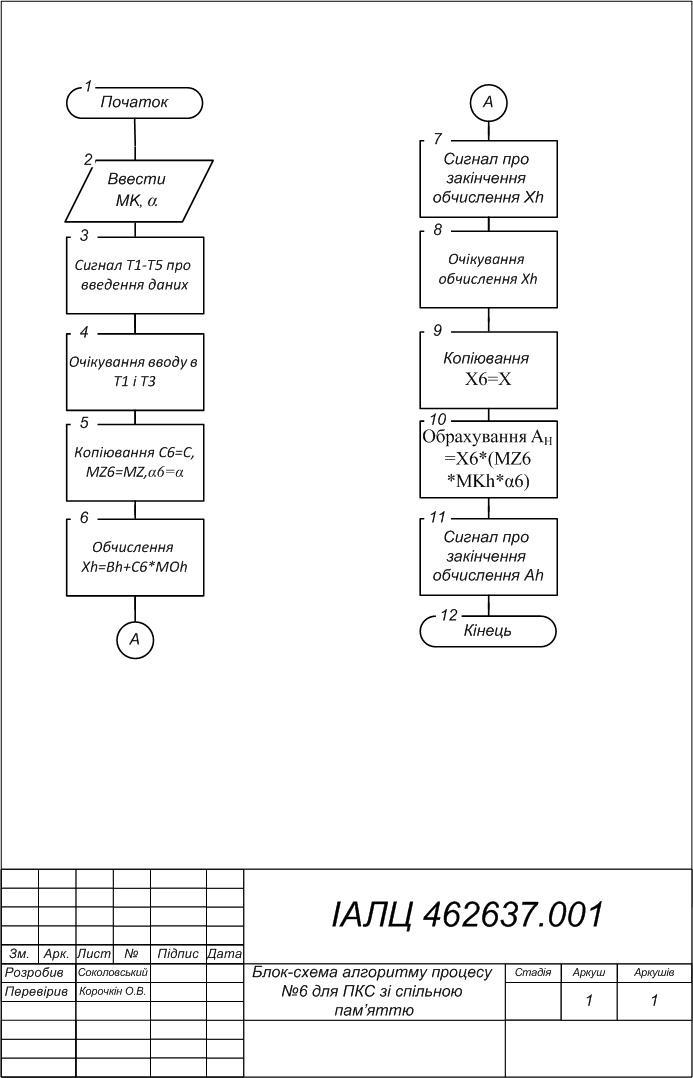
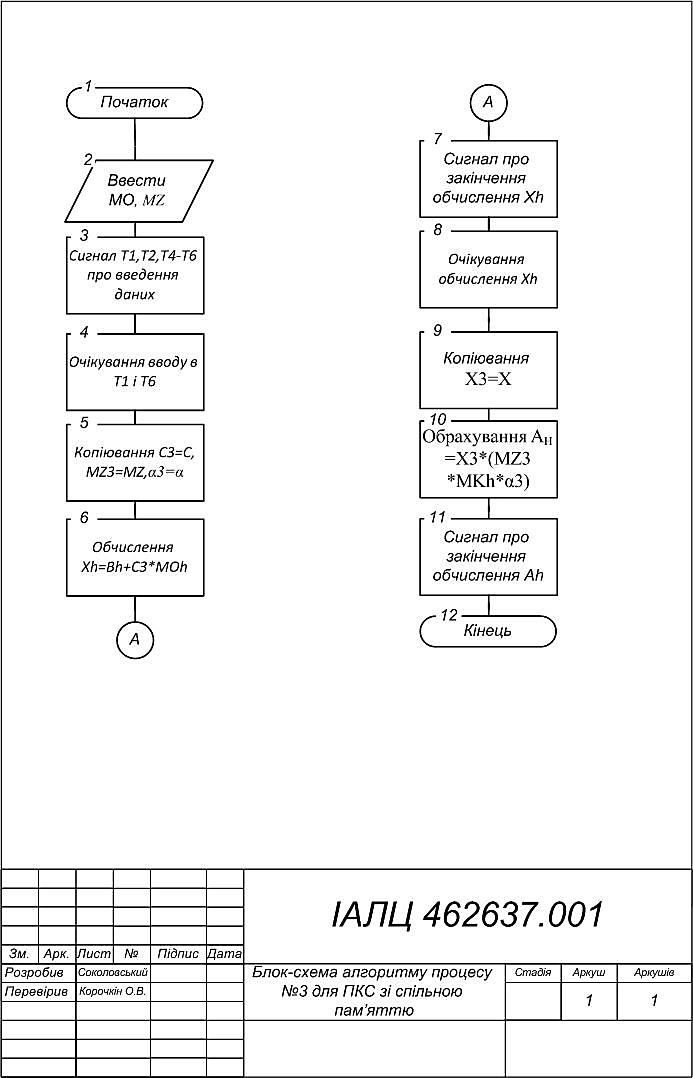
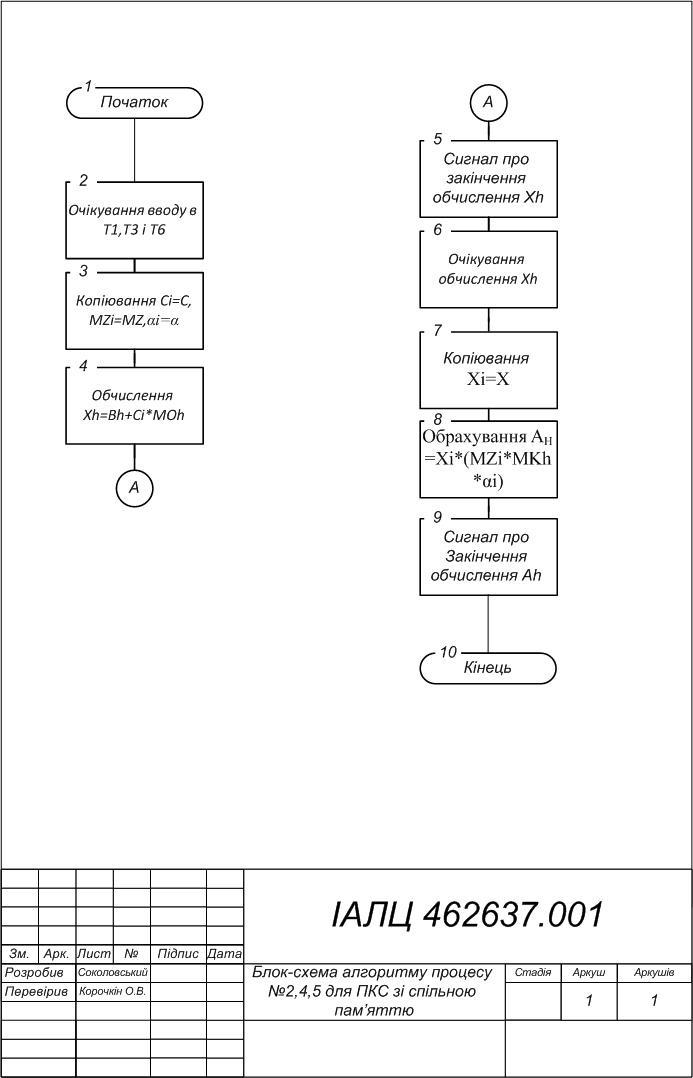
14. Програмні технології для багатоядеоних систем - URL:

http://software.intel.com/en-us/articles/software-techniques-for-shared-cache-multi-core-systems

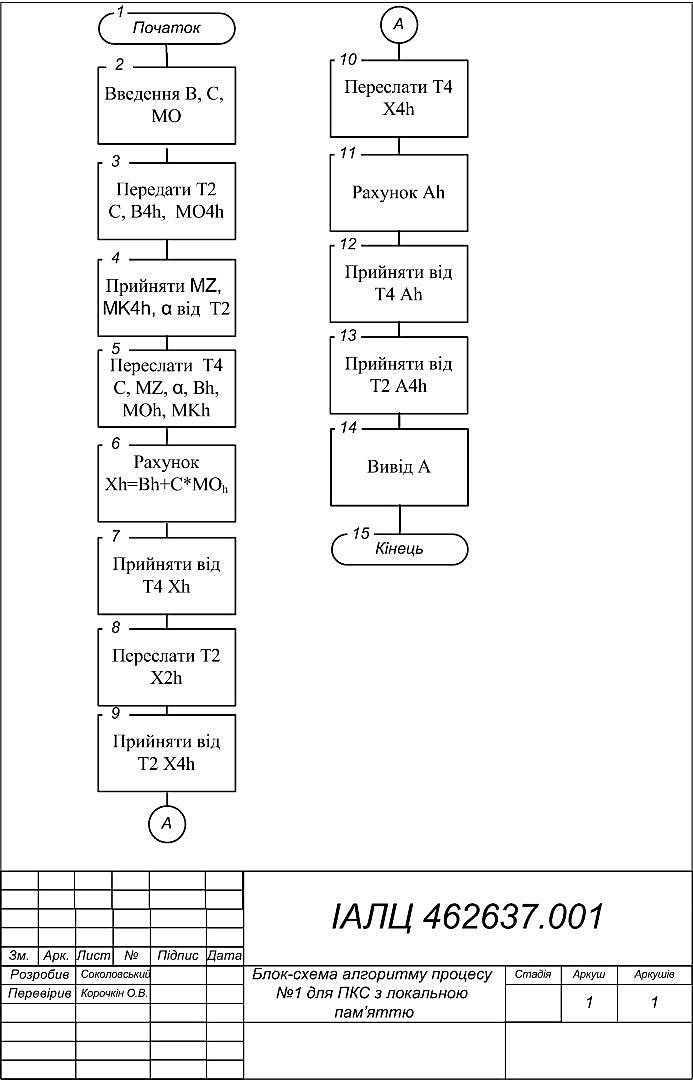
15. AMD HyperTransport™ Технологія - URL: http://www.amd.com/us/products/technologies/hypertransport-technology/Pages/hypertransport-technology.aspx

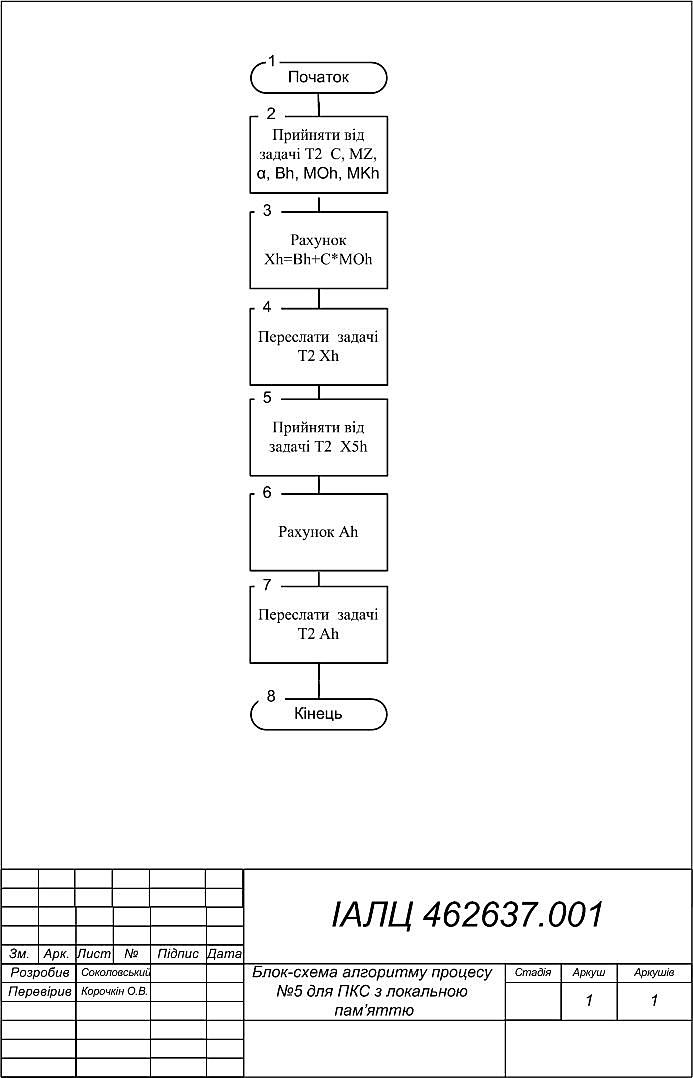
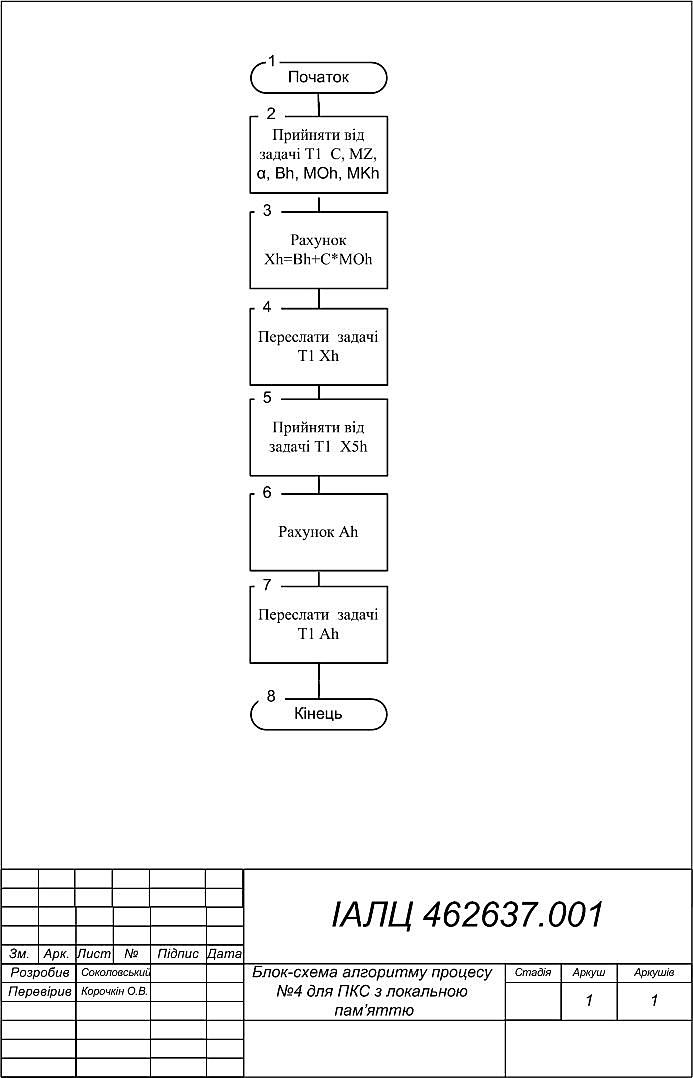
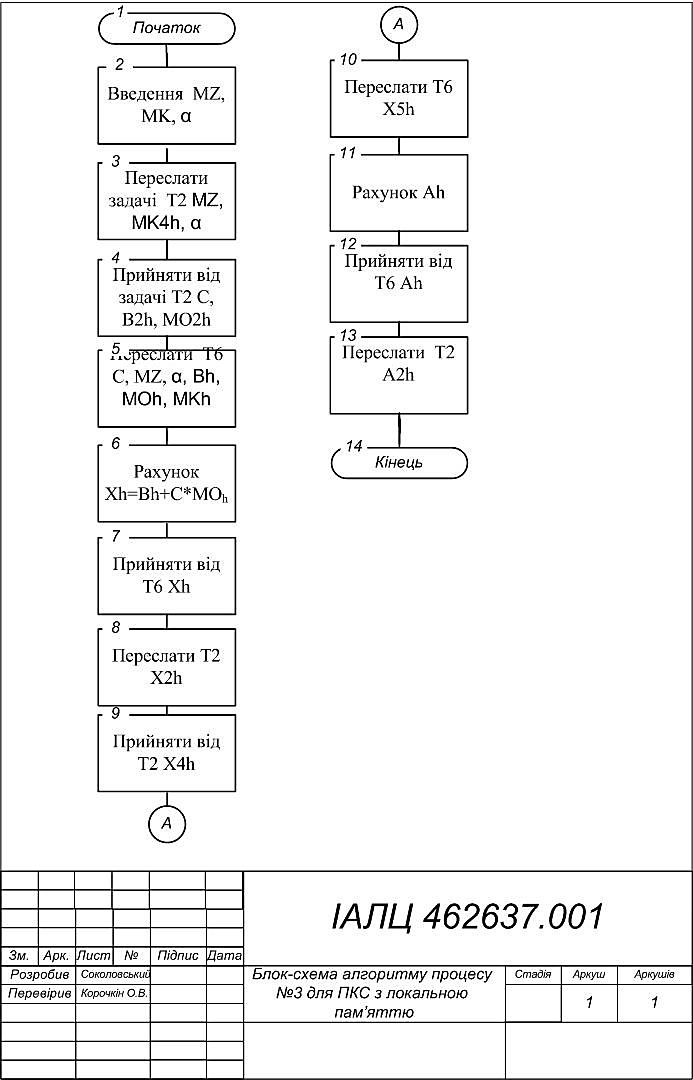
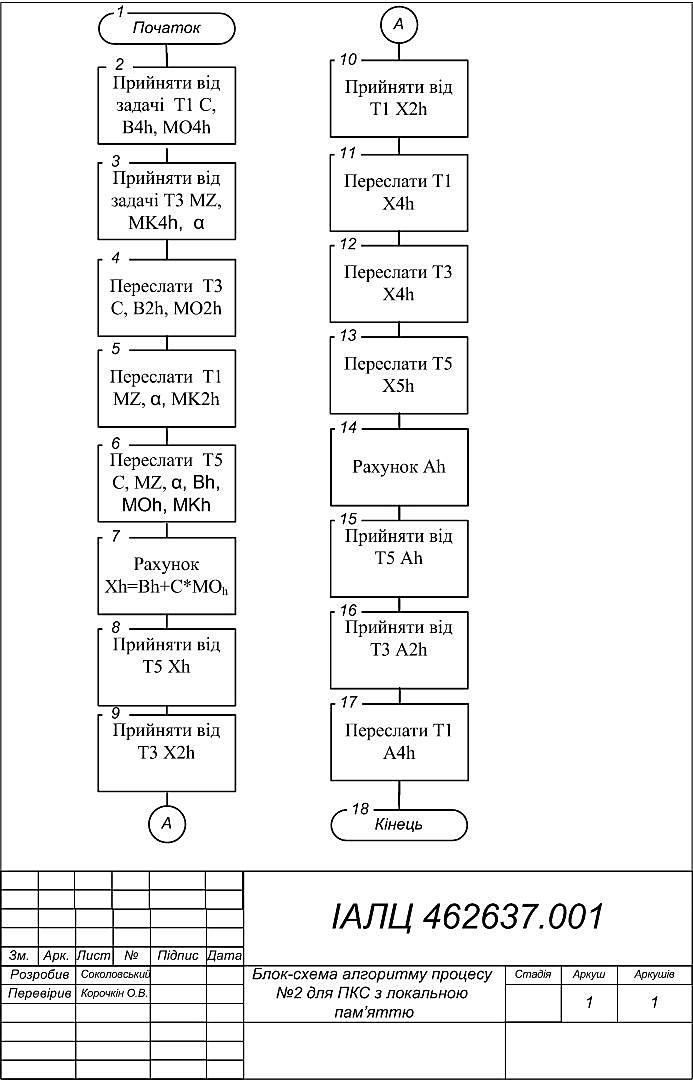
# Додаток А

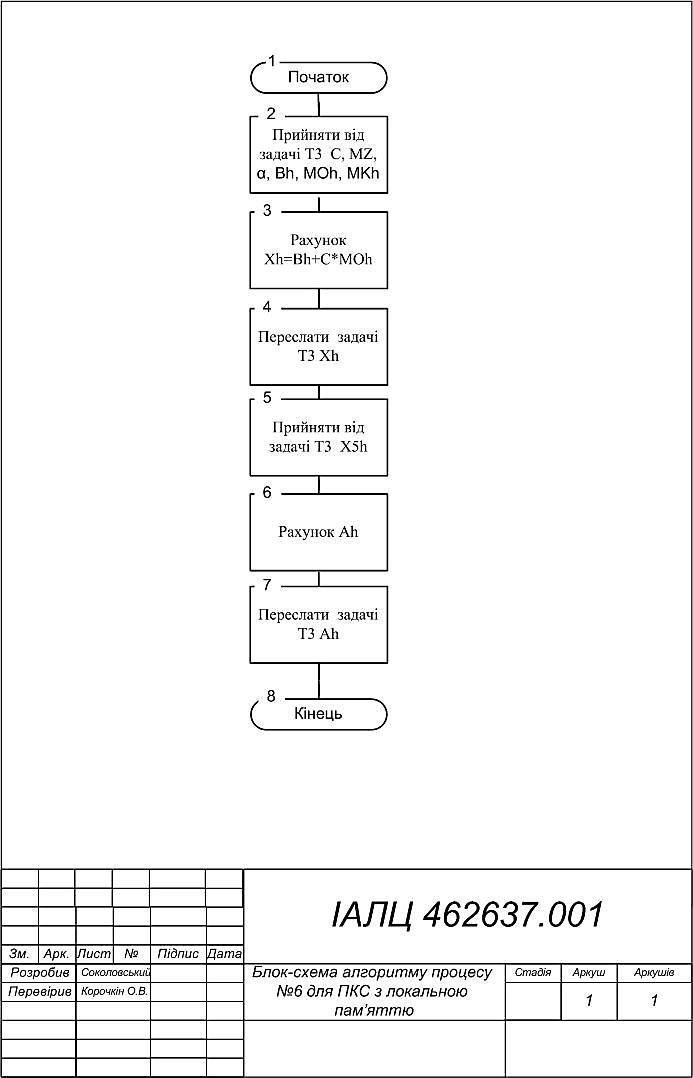




# Додаток Б







# Додаток В

with Ada.Text\_IO, Ada.Integer\_Text\_IO;

use Ada.Text\_IO, Ada.Integer\_Text\_IO;

with Ada.Calendar; use Ada.Calendar;

with ada.float\_text\_io; use ada.float\_text\_io;

procedure Main is

N: integer := 1000; -- Size of vectors and matrixes

P: integer := 6; -- Number of tasks

H: integer := N / P;

H2: integer := 2 \* H;

H3: integer := 3 \* H;

H4: integer := 4 \* H;

H5: integer := 5 \* H;

--Types

type Vector is array(1..N) of integer;

type Matrix is array(1..N) of Vector;

Time\_all: duration;

time1,time2: time;

--Monitor---------------------------------------

protected Monitor is

--Functions

function Copy\_X return Vector;

function Copy\_C return Vector;

function Copy\_L return Integer;

function Copy\_MZ return Matrix;

--Procedures

procedure Set\_C(V: in Vector);

procedure Set\_MZ(MM: in Matrix);

procedure Set\_L(InL: in Integer);

procedure Set\_X(V: in Vector; from: in integer; to: in integer);

procedure Inc\_FT1;

procedure Inc\_FT3;

procedure Inc\_FT6;

procedure Inc\_FX;

procedure Inc\_FA;

--Entries

entry Wait\_Input\_T1;

entry Wait\_Input\_T3;

entry Wait\_Input\_T6;

entry Wait\_X;

entry Wait\_A;

private

FT1: integer := 0; -- Input flag for T1

FT3: integer := 0; -- Input flag for T3

FT6: integer := 0; -- Input flag for T6

FX: integer := 0; -- T flag

FA: integer := 0; -- A flag

--Common resources

X,c: Vector;

MZ: Matrix;

L: Integer;

end Monitor;

protected body Monitor is

--copy

function Copy\_L return Integer is

begin

return L;

end Copy\_L;

function Copy\_X return Vector is

begin

return X;

end Copy\_X;

function Copy\_MZ return Matrix is

begin

return MZ;

end Copy\_MZ;

function Copy\_C return Vector is

begin

return C;

end Copy\_C;

--set procedures--

procedure Set\_L(inL: in Integer) is

begin

L := inL;

end Set\_L;

procedure Set\_C(V: in Vector) is

begin

C := V;

end Set\_C;

procedure Set\_MZ(MM: in Matrix) is

begin

MZ := MM;

end Set\_MZ;

procedure Set\_X(V: in Vector; from: in integer; to: in integer) is

begin

for i in from..to loop

X(i) := V(i);

end loop;

end Set\_X;

--flags for input

procedure Inc\_FT1 is

begin

FT1 := FT1 + 1;

end Inc\_FT1;

procedure Inc\_FT3 is

begin

FT3 := FT3 + 1;

end Inc\_FT3;

procedure Inc\_FT6 is

begin

FT6 := FT6 + 1;

end Inc\_FT6;

--another flags

procedure Inc\_FX is

begin

FX := FX + 1;

end Inc\_FX;

procedure Inc\_FA is

begin

FA := FA + 1;

end Inc\_FA;

--wait for inputing

entry Wait\_Input\_T1 when FT1 = 1 is

begin

null;

end Wait\_Input\_T1;

entry Wait\_Input\_T3 when FT3 = 1 is

begin

null;

end Wait\_Input\_T3;

entry Wait\_Input\_T6 when FT6 = 1 is

begin

null;

end Wait\_Input\_T6;

--wait another

entry Wait\_X when FX = 6 is

begin

null;

end Wait\_X;

entry Wait\_A when FA = 5 is

begin

null;

end Wait\_A;

end Monitor;

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

B,a: Vector;

MO, MK: Matrix;

--Task 1----------------------------------------

task T1 is

pragma Storage\_Size(30\_000\_000);

end T1;

task body T1 is

MZ1,v: Matrix;

C1,X1,InC,XH:Vector;

L1:Integer;

begin

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

--Input data

for i in 1..N loop

B(i) := 1;

InC(I):=1;

end loop;

Monitor.set\_C(InC);

--Signal T2, T3, T4,T5,T6

Monitor.Inc\_FT1;

--Wait T6

Monitor.Wait\_Input\_T6;

--Wait T3

Monitor.Wait\_Input\_T3;

--Copying common resource - B

C1 := Monitor.Copy\_C;

L1 := Monitor.Copy\_L;

MZ1 := Monitor.Copy\_MZ;

--Calculating XH

for i in 1..H loop

XH(i) := 0;

for j in 1..N loop

XH(i) := XH(i) + C1(j) \* MO(i)(j);

end loop;

XH(i) := XH(i) + B(i);

end loop;

Monitor.Set\_X(XH, 1, H);

--Signal T2, T3, T4

Monitor.Inc\_FX;

-- Wait X

Monitor.Wait\_X;

--Copying common resources - T, MZ

X1 := Monitor.Copy\_X;

--Calculating Ah

for i in 1..H loop

for j in 1..N loop

V(j)(i) := 0;

A(i) := 0;

for k in 1..N loop

V(j)(i) := V(j)(i) + MZ1(j)(k) \* MK(k)(i);

end loop;

V(J)(I):=V(J)(I)\*L1;

A(I):=A(i)+X1(j)\*V(J)(I);

end loop;

end loop;

--Wait T2, T3, T4, T5, T6

Monitor.Wait\_A;

--Output A

if(N<=8) then

put("A = ");

for i in 1..N loop

put(A(i));

end loop;

Put\_Line("");

end if;

time2:=clock;

time\_all:=time2-time1;

put\_line("TIME");

put (Float(Time\_all), 4, 3, 0);

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

end T1;

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

--Task 2----------------------------------------

task T2 is

pragma Storage\_Size(30\_000\_000);

end T2;

task body T2 is

MZ2,V: Matrix;

C2,X2,XH:Vector;

L2:Integer;

begin

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

--Wait T1

Monitor.Wait\_Input\_T1;

--Wait T6

Monitor.Wait\_Input\_T6;

--Wait T3

Monitor.Wait\_Input\_T3;

--Copying common resource - B

C2 := Monitor.Copy\_C;

L2 := Monitor.Copy\_L;

MZ2 := Monitor.Copy\_MZ;

--Calculating XH

for i in H..H2 loop

XH(i) := 0;

for j in 1..N loop

XH(i) := XH(i) + C2(j) \* MO(i)(j);

end loop;

XH(i) := XH(i) + B(i);

end loop;

Monitor.Set\_X(XH, H, H2);

--Signal T2, T3, T4

Monitor.Inc\_FX;

-- Wait X

Monitor.Wait\_X;

--Copying common resources - T, MZ

X2 := Monitor.Copy\_X;

--Calculating Ah

for i in H..H2 loop

for j in 1..N loop

V(j)(i) := 0;

A(i) := 0;

for k in 1..N loop

V(j)(i) := V(j)(i) + MZ2(j)(k) \* MK(k)(i);

end loop;

V(J)(I):=V(J)(I)\*L2;

A(I):=A(i)+X2(j)\*V(J)(I);

end loop;

end loop;

--Signal T4

Monitor.Inc\_FA;

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

end T2;

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

----------Task 3--------------------------------

task T3 is

pragma Storage\_Size(30\_000\_000);

end T3;

task body T3 is

MZ3,InMZ,v: Matrix;

C3,X3,XH:Vector;

L3:Integer;

begin

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

--Input data

for i in 1..N loop

for J in 1..N loop

MO(i)(J):= 1;

InMZ(I)(J):=1;

end loop;

end loop;

Monitor.set\_MZ(InMZ);

--Signal T2, T3, T4,T5,T6

Monitor.Inc\_FT3;

--Wait T6

Monitor.Wait\_Input\_T6;

--Wait T1

Monitor.Wait\_Input\_T1;

--Copying common resource - B

C3 := Monitor.Copy\_C;

L3 := Monitor.Copy\_L;

MZ3 := Monitor.Copy\_MZ;

--Calculating XH

for i in H2..H3 loop

XH(i) := 0;

for j in 1..N loop

XH(i) := XH(i) + C3(j) \* MO(i)(j);

end loop;

XH(i) := XH(i) + B(i);

end loop;

Monitor.Set\_X(XH, H2, H3);

--Signal T2, T3, T4

Monitor.Inc\_FX;

-- Wait X

Monitor.Wait\_X;

--Copying common resources - T, MZ

X3 := Monitor.Copy\_X;

--Calculating Ah

for i in H2..H3 loop

for j in 1..N loop

V(j)(i) := 0;

A(i) := 0;

for k in 1..N loop

V(j)(i) := V(j)(i) + MZ3(j)(k) \* MK(k)(i);

end loop;

V(J)(I):=V(J)(I)\*L3;

A(I):=A(i)+X3(j)\*V(J)(I);

end loop;

end loop;

--Signal T4

Monitor.Inc\_FA;

Put\_Line("Task T3 has finished");

end T3;

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

----------Task 4--------------------------------

task T4 is

pragma Storage\_Size(30\_000\_000);

end T4;

task body T4 is

MZ4,v: Matrix;

C4,X4,XH:Vector;

L4:Integer;

begin

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

--Wait T1

Monitor.Wait\_Input\_T1;

--Wait T6

Monitor.Wait\_Input\_T6;

--Wait T3

Monitor.Wait\_Input\_T3;

--Copying common resource - B

C4 := Monitor.Copy\_C;

L4 := Monitor.Copy\_L;

MZ4 := Monitor.Copy\_MZ;

--Calculating XH

for i in H3..H4 loop

XH(i) := 0;

for j in 1..N loop

XH(i) := XH(i) + C4(j) \* MO(i)(j);

end loop;

XH(i) := XH(i) + B(i);

end loop;

Monitor.Set\_X(XH, H3, H4);

--Signal T2, T3, T4

Monitor.Inc\_FX;

-- Wait X

Monitor.Wait\_X;

--Copying common resources - T, MZ

X4 := Monitor.Copy\_X;

--Calculating Ah

for i in H3..H4 loop

for j in 1..N loop

V(j)(i) := 0;

A(i) := 0;

for k in 1..N loop

V(j)(i) := V(j)(i) + MZ4(j)(k) \* MK(k)(i);

end loop;

V(J)(I):=V(J)(I)\*L4;

A(I):=A(i)+X4(j)\*V(J)(I);

end loop;

end loop;

Monitor.Inc\_FA;

Put\_Line("Task T4 has finished");

end T4;

----------Task 5--------------------------------

task T5 is

pragma Storage\_Size(30\_000\_000);

end T5;

task body T5 is

MZ5,v: Matrix;

C5,X5,XH:Vector;

L5:Integer;

begin

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

--Wait T1

Monitor.Wait\_Input\_T1;

--Wait T6

Monitor.Wait\_Input\_T6;

--Wait T3

Monitor.Wait\_Input\_T3;

--Copying common resource - B

C5 := Monitor.Copy\_C;

L5 := Monitor.Copy\_L;

MZ5 := Monitor.Copy\_MZ;

--Calculating XH

for i in H4..H5 loop

XH(i) := 0;

for j in 1..N loop

XH(i) := XH(i) + C5(j) \* MO(i)(j);

end loop;

XH(i) := XH(i) + B(i);

end loop;

Monitor.Set\_X(XH, H4, H5);

--Signal T2, T3, T4

Monitor.Inc\_FX;

-- Wait X

Monitor.Wait\_X;

--Copying common resources - T, MZ

X5 := Monitor.Copy\_X;

--Calculating Ah

for i in H4..H5 loop

for j in 1..N loop

V(j)(i) := 0;

A(i) := 0;

for k in 1..N loop

V(j)(i) := V(j)(i) + MZ5(j)(k) \* MK(k)(i);

end loop;

V(J)(I):=V(J)(I)\*L5;

A(I):=A(i)+X5(j)\*V(J)(I);

end loop;

end loop;

--Signal T4

Monitor.Inc\_FA;

Put\_Line("Task T5 has finished");

end T5;

----------Task 6--------------------------------

task T6 is

pragma Storage\_Size(30\_000\_000);

end T6;

task body T6 is

MZ6,v: Matrix;

C6,X6,XH:Vector;

L6,InL:Integer;

begin

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

--Input data

for i in 1..N loop

for J in 1..N loop

MK(I)(J):=1;

end loop;

end loop;

InL := 1;

Monitor.set\_L(InL);

--Signal T2, T3, T4,T5,T6

Monitor.Inc\_FT6;

--Wait T1

Monitor.Wait\_Input\_T1;

--Wait T3

Monitor.Wait\_Input\_T3;

--Copying common resource - B

C6 := Monitor.Copy\_C;

L6 := Monitor.Copy\_L;

MZ6 := Monitor.Copy\_MZ;

--Calculating XH

for i in h5..n loop

XH(i) := 0;

for j in 1..N loop

XH(i) := XH(i) + C6(j) \* MO(i)(j);

end loop;

XH(i) := XH(i) + B(i);

end loop;

Monitor.Set\_X(XH, h5, N);

--Signal T2, T3, T4

Monitor.Inc\_FX;

-- Wait X

Monitor.Wait\_X;

--Copying common resources - T, MZ

X6 := Monitor.Copy\_X;

--Calculating Ah

for i in h5..n loop

for j in 1..N loop

V(j)(i) := 0;

A(i) := 0;

for k in 1..N loop

V(j)(i) := V(j)(i) + MZ6(j)(k) \* MK(k)(i);

end loop;

V(J)(I):=V(J)(I)\*L6;

A(I):=A(i)+X6(j)\*V(J)(I);

end loop;

end loop;

--Signal T4

Monitor.Inc\_FA;

Put\_Line("Task T6 has finished");

end T6;

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

begin

time1:=clock;

end Main;

# Додаток Г

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

-- PRO-2 --

-- Course. Ada. Rendezvous --

-- --

-- Sokolovskiy Stanislav,gr IO-01 --

-- 19.04.2013 --

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

with Ada.Text\_IO; use Ada.Text\_IO;

with Ada.Integer\_Text\_IO; use Ada.Integer\_Text\_IO;

procedure CoursePRO is

N: integer := 2400; --size of structures

P: integer := 6; --count of processors

H: Integer := N / P;

type vector is array(integer range <>) of integer;

Subtype Vector\_N is Vector(1..N);

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

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

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

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

Subtype Vector\_H is Vector(1..H);

Subtype Vector\_HN is Vector(1..(N-5\*h));

Subtype Vector\_2HN is Vector(1..(N-4\*h));

Subtype Vector\_3HN is Vector(1..(N-3\*h));

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

Subtype Matrix\_N is Matrix(1..N);

Subtype Matrix\_5H is Matrix(1..5\*H);

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

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

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

Subtype Matrix\_H is Matrix(1..H);

Subtype Matrix\_4HN is Matrix(1..(N-2\*h));

Subtype Matrix\_3HN is Matrix(1..(N-3\*h));

Subtype Matrix\_2HN is Matrix(1..(N-4\*h));

Subtype Matrix\_HN is Matrix(1..(N-5\*h));

task T1 is

ENTRY SandT2\_T1\_Data(MZ: in Matrix\_n; Alpha: in Integer; MK1: in Matrix\_H; MK2: in Matrix\_H );

ENTRY SandT4\_T1\_X(XH: in Vector\_H);

ENTRY SandT2\_T1\_X(XH1: in Vector\_2H; XH2: in Vector\_2HN);

ENTRY SandT4\_T1\_A(AH: in Vector\_H);

ENTRY SandT2\_T1\_A(AH1: in Vector\_2H; AH2: in Vector\_2HN);

end T1;

task T2 is

ENTRY SandT1\_T2\_Data(C: in Vector\_N; B1: in Vector\_2H; B2: in Vector\_2HN; MO1: in Matrix\_2H; MO2: in Matrix\_2HN);

ENTRY SandT3\_T2\_Data(MZ: in Matrix\_N; Alpha:in Integer; MK1: in Matrix\_2h; MK2: in Matrix\_2h);

ENTRY SandT5\_T2\_X(XH: in Vector\_H);

ENTRY SandT3\_T2\_X(XH1: in Vector\_H; XH2: in Vector\_HN);

ENTRY SandT1\_T2\_X(XH1: in Vector\_H ; XH2: in Vector\_H );

ENTRY SandT5\_T2\_A(AH: in Vector\_H);

ENTRY SandT3\_T2\_A(AH1: in Vector\_H; AH2: in Vector\_HN);

end T2;

task T3 is

ENTRY SandT2\_T3\_Data(C: in Vector\_N; B1: in Vector\_H; B2: in Vector\_HN; MO1: in Matrix\_H; MO2: in Matrix\_HN);

ENTRY SandT6\_T3\_X(XH: in Vector\_HN);

ENTRY SandT2\_T3\_X(XH1: in Vector\_2H; XH2: in Vector\_2H);

ENTRY SandT4\_T3\_A(AH: in Vector\_H);

ENTRY SandT6\_T3\_A(AH: in Vector\_HN);

end T3;

task T4 is

ENTRY SandT1\_T4\_Data(C: in Vector\_N;MZ: in Matrix\_n; Alpha: in Integer; B: in Vector\_H; MO: in Matrix\_h; MK: in Matrix\_H);

ENTRY SandT1\_T4\_X(XH1: in Vector\_3H; XH2: in Vector\_2HN);

end T4;

task T5 is

ENTRY SandT2\_T5\_Data(C: in Vector\_N;MZ: in Matrix\_n; Alpha: in Integer; B: in Vector\_H; MO: in Matrix\_h; MK: in Matrix\_H);

ENTRY SandT2\_T5\_X(XH1: in Vector\_4H; XH2: in Vector\_HN);

end T5;

task T6 is

ENTRY SandT3\_T6\_Data(C: in Vector\_N;MZ: in Matrix\_n; Alpha: in Integer; B: in Vector\_HN; MO: in Matrix\_hN; MK: in Matrix\_HN);

ENTRY SandT3\_T6\_X(XH: in Vector\_5H);

end T6;

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

task body T1 is

--inAll

B\_1,C\_1:Vector\_N;

MO\_1,MZ\_1:Matrix\_n;

X23,A23:Vector\_2H;

X123:Vector\_3h;

X56,A56:Vector\_2HN;

X4,X1:Vector\_h;

Alpha\_1,Res: Integer;

MK\_1:Matrix\_h;

MK\_2:Matrix\_h;

A4,A1: Vector\_h;

x\_1,a\_all:Vector\_N;

begin

put\_Line("T1 started!");

for i in 1 .. N loop

for j in 1 .. n loop

MO\_1(i)(j):=1;

end loop;

B\_1(i):=1;

C\_1(i):=1;

end loop;

t2.SandT1\_T2\_Data(C\_1, B\_1(H+1..3\*H), B\_1(4\*H+1..n),MO\_1(H+1..3\*H),MO\_1(4\*H+1..n));

ACCEPT SandT2\_T1\_Data(MZ: in Matrix\_n; Alpha: in Integer; MK1: in Matrix\_H; MK2: in Matrix\_H) DO

MZ\_1:=MZ;

Alpha\_1:=Alpha;

MK\_1:=MK1;

MK\_2:=MK2;

END SandT2\_T1\_Data;

T4.SandT1\_T4\_Data(C\_1,MZ\_1,Alpha\_1, B\_1(3\*H+1..4\*H), MO\_1(3\*H+1..4\*H), MK\_2);

--calculation

for i in 1..H loop

res:=0;

for j1 in 1..N loop

res:=Res+C\_1(j1)\*MO\_1(I)(J1);

END LOOP;

res:=res+B\_1(I);

X1(i):=res;

END LOOP;

accept SandT4\_T1\_X(XH: in Vector\_H) do

X4:=XH;

end SandT4\_T1\_X;

t2.SandT1\_T2\_X(X1, X4);

accept SandT2\_T1\_X(XH1: in Vector\_2H; XH2: in Vector\_2HN) do

X23:=XH1;

X56:=XH2;

end SandT2\_T1\_X;

for i in 1..h loop

X123(i):=X1(i);

END LOOP;

for i in 1..2\*h loop

X123(i+h):=X23(i);

END LOOP;

T4.SandT1\_T4\_X(X123,X56);

--Calculation A

for i in 1..3\*h loop

X\_1(i):=X123(i);

END LOOP;

for i in 1..h loop

X\_1(i+3\*h):=X4(i);

END LOOP;

for i in 1..(N-4\*H) loop

X\_1(i+4\*h):=X56(i);

END LOOP;

for i in 1..H loop

res:=0;

for j in 1..N loop

res:=0;

for j1 in 1..N loop

res:=Res+MK\_1(i)(j)\*MZ\_1(j)(j1);

END LOOP;

res:=res\*Alpha\_1\*X\_1(j);

END LOOP;

A1(i):=res;

END LOOP;

accept SandT4\_T1\_A(AH: in Vector\_H) do

A4:=AH;

end SandT4\_T1\_A;

accept SandT2\_T1\_A(AH1: in Vector\_2H; AH2: in Vector\_2HN) do

A23:=AH1;

A56:=AH2;

end SandT2\_T1\_A;

PUT\_LINE("END CALC T1");

for i in 1..h loop

A\_all(i+3\*h):=A4(i);

A\_all(i):=A1(i);

END LOOP;

for i in 1..2\*h loop

A\_all(i+h):=A23(i);

END LOOP;

for i in 1..N-4\*H loop

A\_all(i+4\*h):=A56(i);

END LOOP;

if N < 40 then

for i in 1..N loop

PUT(A\_all(i));

END LOOP;

end if;

end T1;

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

task body T2 is

C\_2:Vector\_N;

MZ\_2:Matrix\_n;

Alpha\_2,Res: Integer;

B1\_2,X12,X45,x23,A23: Vector\_2H;

B2\_2,A56: Vector\_2HN;

X56:Vector\_2hN;

MO2\_2: Matrix\_2HN;

MO1\_2: Matrix\_2H;

MK1\_2,MK2\_2: Matrix\_2h;

X1234:Vector\_4H;

X2,X5,x3,x1,x4,a5,a3,a2:Vector\_H;

X6,a6: Vector\_HN;

x\_2:Vector\_N;

begin

put\_Line("T2 started!");

ACCEPT SandT1\_T2\_Data(C: in Vector\_N; B1: in Vector\_2H; B2: in Vector\_2HN; MO1: in Matrix\_2H; MO2: in Matrix\_2HN) DO

C\_2:=C;

B1\_2:=B1;

B2\_2:=b2;

MO1\_2:=MO1;

MO2\_2:=MO2;

END SandT1\_T2\_Data;

t3.SandT2\_T3\_Data(C\_2,B1\_2(h+1..2\*h),B2\_2(h+1..(n-(4\*h))), MO1\_2(h+1..2\*h),MO2\_2(h+1..(n-(4\*h))));

ACCEPT SandT3\_T2\_Data(MZ: in Matrix\_N; Alpha:in Integer; MK1: in Matrix\_2h; MK2: in Matrix\_2h) DO

MZ\_2:=MZ;

Alpha\_2:=Alpha;

MK1\_2:=MK1;

MK2\_2:=MK2;

END SandT3\_T2\_Data;

T1.SandT2\_T1\_Data(MZ\_2, Alpha\_2, MK1\_2(1..h), MK2\_2(1..h));

T5.SandT2\_T5\_Data(C\_2, MZ\_2, Alpha\_2, B2\_2(1..h), MO2\_2(1..h), MK2\_2(h+1..2\*h));

--calculation

for i in 1..H loop

res:=0;

for j1 in 1..N loop

res:=Res+C\_2(j1)\*MO1\_2(i)(J1);

END LOOP;

res:=res+B1\_2(I);

X2(i):=res;

END LOOP;

accept SandT5\_T2\_X(XH: in Vector\_H) do

X5:=XH;

end SandT5\_T2\_X;

accept SandT3\_T2\_X(XH1: in Vector\_H; XH2: in Vector\_HN) do

X3:=XH1;

X6:=XH2;

end SandT3\_T2\_X;

accept SandT1\_T2\_X(XH1: in Vector\_H; XH2: in Vector\_H) do

X1:=XH1;

X4:=XH2;

end SandT1\_T2\_X;

for i in 1..h loop

X23(i):=X2(i);

X56(i):=X5(i);

X12(i):=X1(i);

X45(i):=X4(i);

END LOOP;

for i in 1..h loop

X23(i+h):=X3(i);

X12(i+h):=X2(i);

X45(i+h):=X5(i);

X1234(i+3\*h):=X4(i);

X1234(i+2\*h):=X3(i);

END LOOP;

for i in 1..N-5\*h loop

X56(i+h):=X6(i);

END LOOP;

for i in 1..2\*h loop

X1234(i):=X12(i);

END LOOP;

T1.SandT2\_T1\_X(X23,X56);

T3.SandT2\_T3\_X(X12,X45);

T5.sandT2\_T5\_X(X1234,X6);

--Calculation A

for i in 1..4\*h loop

X\_2(i):=X1234(i);

END LOOP;

for i in 1..(n-4\*h) loop

X\_2(i+4\*h):=X56(i);

END LOOP;

for i in 1..H loop

res:=0;

for j in 1..N loop

res:=0;

for j1 in 1..N loop

res:=Res+MK1\_2(H+i)(j)\*MZ\_2(j)(j1);

END LOOP;

res:=res\*Alpha\_2\*X\_2(j);

END LOOP;

A2(i):=res;

END LOOP;

accept SandT5\_T2\_A(AH: in Vector\_H) do

A5:=AH;

end SandT5\_T2\_A;

accept SandT3\_T2\_A(AH1: in Vector\_H; AH2: in Vector\_HN) do

A3:=AH1;

A6:=AH2;

end SandT3\_T2\_A;

for i in 1..h loop

A23(i):=A2(i);

A56(i):=A5(I);

END LOOP;

for i in 1..h loop

A23(H+i):=A3(i);

END LOOP;

for i in 1..n-5\*h loop

A56(i+h):=A6(i);

END LOOP;

T1.SandT2\_T1\_a(a23,A56);

PUT\_LINE("END CALC T2");

end T2;

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

task body T3 is

--inAll

MZ\_3,MK\_3 : Matrix\_n;

alpha\_3,res:Integer:=1;

C\_3:Vector\_n;

X12,X45:Vector\_2h;

B1\_3,X3,A3:Vector\_h;

B2\_3,X6,A6:Vector\_hN;

MO1\_3:Matrix\_h;

MO2\_3:Matrix\_hN;

x12345:Vector\_5h;

x\_3:Vector\_N;

begin

put\_Line("T3 started!");

for i in 1 .. N loop

for j in 1 .. n loop

MZ\_3(i)(j):=1;

MK\_3(i)(j):=1;

end loop;

end loop;

ACCEPT SandT2\_T3\_Data(C: in Vector\_N; B1: in Vector\_H; B2: in Vector\_HN; MO1: in Matrix\_H; MO2: in Matrix\_HN) DO

C\_3:=C;

B1\_3:=B1;

B2\_3:=B2;

MO1\_3:=MO1;

MO2\_3:=MO2;

END SandT2\_T3\_Data;

T2.SandT3\_T2\_Data(MZ\_3,Alpha\_3,MK\_3(1..2\*H),MK\_3(3\*H+1..5\*H));

T6.SandT3\_T6\_Data(C\_3,MZ\_3,Alpha\_3,B2\_3,MO2\_3,MK\_3(5\*H+1..N));

--calculation

for i in 1..H loop

res:=0;

for j1 in 1..N loop

res:=Res+C\_3(j1)\*MO1\_3(I)(J1);

END LOOP;

res:=res+B1\_3(I);

X3(i):=res;

END LOOP;

accept SandT6\_T3\_X(XH: in Vector\_HN) do

X6:=XH;

end SandT6\_T3\_X;

T2.SandT3\_T2\_X(X3,x6);

accept SandT2\_T3\_X(XH1: in Vector\_2H; XH2: in Vector\_2H) do

X12:=XH1;

X45:=XH2;

end SandT2\_T3\_X;

for i in 1..2\*h loop

X12345(i):=X12(i);

X12345(i+3\*H):=X45(i);

END LOOP;

for i in 1..h loop

X12345(2\*h+I):=X3(i);

END LOOP;

T6.Sandt3\_T6\_X(x12345);

--Calculation A

for i in 1..5\*h loop

X\_3(i):=X12345(i);

END LOOP;

for i in 1..N-5\*h loop

X\_3(i+5\*h):=X6(i);

END LOOP;

for i in 1..H loop

res:=0;

for j in 1..N loop

res:=0;

for j1 in 1..N loop

res:=Res+MK\_3(2\*h+i)(j)\*MZ\_3(j)(j1);

END LOOP;

res:=res\*Alpha\_3\*X\_3(j);

END LOOP;

A3(i):=res;

END LOOP;

accept SandT6\_T3\_A(AH: in Vector\_HN) do

A6:=AH;

end SandT6\_T3\_A;

T2.SandT3\_t2\_a(A3,A6);

PUT\_LINE("END CALC T3");

end T3;

------------------task 4--------------------------------------

task body T4 is

C\_4:Vector\_N;

MZ\_4:Matrix\_n;

Alpha\_4,res:Integer;

MO\_4,MK\_4:Matrix\_h;

B\_4,X4,A4:Vector\_h;

X\_4:Vector\_N;

x123:Vector\_3h;

x56:Vector\_2hN;

begin

put\_Line("T4 started!");

ACCEPT SandT1\_T4\_Data(C: in Vector\_N;MZ: in Matrix\_n; Alpha: in Integer; B: in Vector\_H; MO: in Matrix\_h; MK: in Matrix\_H) DO

C\_4:=C;

MZ\_4:=MZ;

Alpha\_4:=Alpha;

B\_4:=b;

MO\_4:=MO;

MK\_4:=MK;

END SandT1\_T4\_Data;

--calculation

for i in 1..H loop

res:=0;

for j1 in 1..N loop

res:=Res+C\_4(j1)\*MO\_4(I)(J1);

END LOOP;

res:=res+B\_4(I);

X4(i):=res;

END LOOP;

T1.SandT4\_T1\_X(X4);

accept SandT1\_T4\_X(XH1: in Vector\_3H; XH2: in Vector\_2HN) do

X123:=XH1;

X56:=XH2;

end SandT1\_T4\_X;

--Calculation A

for i in 1..3\*h loop

X\_4(i):=X123(i);

END LOOP;

for i in 1..(N-H\*4) loop

X\_4(i+4\*h):=X56(i);

END LOOP;

for i in 1..H loop

X\_4(i+3\*h):=X4(i);

END LOOP;

for i in 1..H loop

res:=0;

for j in 1..N loop

res:=0;

for j1 in 1..N loop

res:=Res+MK\_4(i)(j)\*MZ\_4(j)(j1);

END LOOP;

res:=res\*Alpha\_4\*X\_4(j);

END LOOP;

A4(i):=res;

END LOOP;

T1.SandT4\_t1\_a(A4);

PUT\_LINE("END CALC T4");

end T4;

-----------------task 5--------------------------------------

task body T5 is

C\_5:Vector\_N;

MZ\_5:Matrix\_n;

Alpha\_5,res:Integer;

MO\_5,MK\_5:Matrix\_h;

X\_5:Vector\_N;

B\_5,X5,A5:Vector\_h;

x1234:Vector\_4h;

X6:Vector\_HN;

begin

put\_Line("T5 started!");

ACCEPT SandT2\_T5\_Data(C: in Vector\_N;MZ: in Matrix\_n; Alpha: in Integer; B: in Vector\_H; MO: in Matrix\_h; MK: in Matrix\_H) DO

C\_5:=C;

MZ\_5:=MZ;

Alpha\_5:=Alpha;

B\_5:=b;

MO\_5:=MO;

MK\_5:=MK;

END SandT2\_T5\_Data;

--calculation

for i in 1..H loop

res:=0;

for j1 in 1..N loop

res:=Res+C\_5(j1)\*MO\_5(I)(J1);

END LOOP;

res:=res+B\_5(I);

X5(i):=res;

END LOOP;

T2.SandT5\_T2\_X(X5);

accept SandT2\_T5\_X(XH1: in Vector\_4H; XH2: in Vector\_HN) do

X1234:=XH1;

X6:=XH2;

end SandT2\_T5\_X;

--Calculation A

for i in 1..4\*h loop

X\_5(i):=X1234(i);

END LOOP;

for i in 1..(N-H\*5) loop

X\_5(i+5\*h):=X6(i);

END LOOP;

for i in 1..H loop

X\_5(i+4\*h):=X5(i);

END LOOP;

for i in 1..H loop

res:=0;

for j in 1..N loop

res:=0;

for j1 in 1..N loop

res:=Res+MK\_5(i)(j)\*MZ\_5(j)(j1);

END LOOP;

res:=res\*Alpha\_5\*X\_5(j);

END LOOP;

A5(i):=res;

END LOOP;

T2.SandT5\_t2\_a(A5);

PUT\_LINE("END CALC T5");

end T5;

----------------task 6--------------------------------------

task body T6 is

C\_6:Vector\_N;

MZ\_6:Matrix\_n;

Alpha\_6,res:Integer;

MO\_6,MK\_6:Matrix\_hN;

X\_6:Vector\_N;

B\_6,X6,A6:Vector\_hN;

x12345:Vector\_5h;

begin

put\_Line("T6 started!");

ACCEPT SandT3\_T6\_Data(C: in Vector\_N;MZ: in Matrix\_n; Alpha: in Integer; B: in Vector\_HN; MO: in Matrix\_hN; MK: in Matrix\_HN) DO

C\_6:=C;

MZ\_6:=MZ;

Alpha\_6:=Alpha;

B\_6:=b;

MO\_6:=MO;

MK\_6:=MK;

END SandT3\_T6\_Data;

--calculation

for i in 1..(n-5\*h) loop

res:=0;

for j1 in 1..N loop

res:=Res+C\_6(j1)\*MO\_6(I)(J1);

END LOOP;

res:=res+B\_6(I);

X6(i):=res;

END LOOP;

T3.SandT6\_T3\_X(X6);

accept SandT3\_T6\_X(XH: in Vector\_5H) do

X12345:=XH;

end SandT3\_T6\_X;

--Calculation A

for i in 1..5\*h loop

X\_6(i):=X12345(i);

END LOOP;

for i in 1..(n-5\*h) loop

X\_6(i+5\*h):=X6(i);

END LOOP;

for i in 1..(n-5\*h) loop

res:=0;

for j in 1..N loop

res:=0;

for j1 in 1..N loop

res:=Res+MK\_6(i)(j)\*MZ\_6(j)(j1);

END LOOP;

res:=res\*Alpha\_6\*X\_6(j);

END LOOP;

A6(i):=res;

END LOOP;

T3.SandT6\_t3\_a(A6);

PUT\_LINE("END CALC T6");

end T6;

begin

null;

end CoursePRO;