Національний технічний університет України  
«Київський політехнічний інститут»

Кафедра обчислювальної техніки

(повна назва кафедри, циклової комісії)

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)

Студента 3 курсу групи ІО-01

напряму підготовки 050102

«Комп’ютерна інженерія»

Куріло С.А.

(прізвище та ініціали)

Керівник доцент Корочкін О.В.

Національна оцінка

Кількість балів:

Оцінка: ECTS

**Курсова робота**

з дисципліни «Паралельні та розподілені обчислення»

(назва дисципліни)

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

Київ – 2013 рік

Національний технічний університет України  
«Київський політехнічний інститут»

Факультет (інститут) інформатики та обчислювальної техніки

(повна назва)

Кафедра обчислювальної техніки

(повна назва)

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

(шифр і назва)

**Завдання**

на курсову роботу студенту

Куріло Сергій Андрійович

(прізвище, ім’я, по батькові)

1. Тема роботи «Розробка програмного забезпечення для паралельних комп’ютерних систем».
2. Керівник роботи к. т. н., доцент Корочкін Олександр Володимирович.

(вчене звання, науковий ступінь, прізвище та ініціали)

1. Строк подання студентом роботи 14 травня 2013 р.
2. Вихідні дані для роботи:

* огляд чотириядерних процесорів компанії Intel;
* математична задача ;
* структури паралельної комп’ютерної системи з спільною пам’яттю (ПКС СП) та паралельної комп’ютерної системи з локальною пам’яттю (ПКС ЛП);
* бібліотека програмування: Win32.
* Мови програмування: C++, Ada

1. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

* огляд чотириядерних процесорів компанії Intel;
* розробка і тестування програми ПРГ1 для ПКС СП;
* розробка і тестування програми ПРГ2 для ПКС ЛП.



Рисунок 1.1 – Структура ПКС СП



Рисунок 1.2 – Структура ПКС ЛП

1. Перелік графічного матеріалу:

* структурна схема ПКС СП;
* структурна схема ПКС ЛП;
* схеми алгоритмів процесів і головної програми для ПРГ1;
* схеми алгоритмів процесів і головної програми для ПРГ2.

1. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| № з/п | Назва етапів виконання КР | Строк виконання етапів КР |
|  | Виконання огляду для розділу 1 | 25.03.2013 |
|  | Розробка паралельного алгоритму рішення задачі | 06.04.2013 |
|  | Розробка алгоритмів процесів | 07.04.2013 |
|  | Розробка схем взаємодії процесів | 13.04.2013 |
|  | Розробка програм | 20.04.2013 |
|  | Тестування програм | 23.04.2013 |
|  | Оформлення КР | 08.05.2013 |
|  | Захист КР | 23.05.2013 |

Студент

(підпис) (прізвище та ініціали)

Керівник роботи

(підпис) (прізвище та ініціали)

**ПОЯСНЮВАЛЬНА  
ЗАПИСКА**

ЗМІСТ

[ВСТУП……… 6](#_Toc356288647)

[РОЗДІЛ 1. ОГЛЯД ЧОТИРИЯДЕРНИХ ПРОЦЕСОРІВ INTEL 7](#_Toc356288648)

[1.1 Багатоядерність як нова тенденція у розвитку потужностей 7](#_Toc356288649)

[1.2 Основні характеристики процесорів 9](#_Toc356288650)

[1.3 Загальна характеристика чотириядерних процесорів Intel Core і5 13](#_Toc356288651)

[1.3.1 Огляд процесорів Intel Core i5 Previous Gen[12] 14](#_Toc356288652)

[1.3.2 Огляд процесорів Intel Core i5 2nd Gen[13] 14](#_Toc356288653)

[1.3.3 Огляд процесорів Intel Core i5 3rd Gen[14] 15](#_Toc356288654)

[1.4 Загальна характеристика чотириядерних процесорів Intel Core i7 18](#_Toc356288655)

[1.4.1 Огляд процесорів Intel Core і7 Previous Gen[17] 19](#_Toc356288656)

[1.4.2 Огляд процесорів Intel Core і7 Previous Gen Extreme Edition[18] 20](#_Toc356288657)

[1.4.3 Огляд процесорів Intel Core і7 2nd Gen[19] 21](#_Toc356288658)

[1.4.4 Огляд процесорів Intel Core і7 2nd Gen Extreme Edition[20] 21](#_Toc356288659)

[1.4.5 Огляд процесорів Intel Core і7 3rd Gen[21] 22](#_Toc356288660)

[1.4.6 Огляд процесорів Intel Core і7 3rd Gen Extreme Edition[22] 22](#_Toc356288661)

[1.5 Висновки до розділу 1 25](#_Toc356288662)

[РОЗДІЛ 2. РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП 26](#_Toc356288663)

[2.1 Розробка паралельного математичного алгоритму 26](#_Toc356288664)

[2.2 Розробка алгоритмів процесів 27](#_Toc356288665)

[2.3 Розробка схеми взаємодії процесів 29](#_Toc356288666)

[2.4 Розробка програми ПРГ1 33](#_Toc356288667)

[2.5 Тестування програми ПРГ1 33](#_Toc356288668)

[2.6 Висновки до розділу 2 39](#_Toc356288669)

[РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП 40](#_Toc356288670)

[3.1 Розробка паралельного математичного алгоритму 40](#_Toc356288671)

[3.2 Розробка алгоритмів процесів 41](#_Toc356288672)

[3.3 Розробка схеми взаємодії процесів 42](#_Toc356288673)

[3.4 Розробка програми ПРГ2 43](#_Toc356288674)

[3.5 Тестування програми ПРГ2 45](#_Toc356288675)

[3.6 Висновки до розділу 3 49](#_Toc356288676)

[ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ПО РОБОТІ 51](#_Toc356288677)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52](#_Toc356288678)

[ДОДАТКИ………. 56](#_Toc356288679)

[Додаток А. Структурна схема ПКС СП 57](#_Toc356288680)

[Додаток Б. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ1 58](#_Toc356288681)

[Додаток В. Схема алгоритму процесів для програми ПРГ1 61](#_Toc356288682)

[Додаток Г. Лістинг програми ПРГ1 63](#_Toc356288683)

[Додаток Д. Код скрипта для тестування ПРГ1 70](#_Toc356288684)

[Додаток Е. Структурна схема ПКС ЛП 51](#_Toc356288685)

[Додаток Ж. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ2 52](#_Toc356288686)

[Додаток З. Схема алгоритму процесів для програми ПРГ2 75](#_Toc356288687)

[Додаток К. Лістинг програми ПРГ2 78](#_Toc356288688)

[Додаток Л. Код скрипта для тестування ПРГ2 84](#_Toc356288689)

ВСТУП

Курсова робота по дисципліні «Паралельні і розподілені обчислення» складається з трьох розділів.

В першому розділі «Огляд чотириядерних процесорів компанії Intel» описані передумови виникнення багатоядерності, загальна характеристика процесорів Intel та основні обчислювальні якості процесорів Intel лінійок i5 та i7.

Другий та третій розділи присвячені розробці програми для обчислення математичної задачі в паралельній комп’ютерній системі зі спільною та локальною пам’яттю відповідно. Програмне забезпечення для комп’ютерної системи зі спільною пам’яттю розроблено на мові C++ з використанням бібліотеки Win32, а для комп’ютерної системи з локальною пам’яттю – розроблено на мові Ada з використанням механізму організації взаємодії Rendezvous. Проведено тестування отриманих програмних продуктів і зроблено висновки по їх ефективності.

Лістинги та алгоритми розроблених програм наведено у додатках.

# ОГЛЯД ЧОТИРИЯДЕРНИХ ПРОЦЕСОРІВ INTEL

## Багатоядерність як нова тенденція у розвитку потужностей

Закон Мура прогнозує подвоєння кiлькостi транзисторів на кристалі мікропроцесора кожні 2 роки. [1] Збільшення кiлькостi транзисторів означає збільшення продуктивності. Але збільшення кiлькостi транзисторів вимагає постійного вдосконалення технологічного процесу виробництва інтегральних схем. Останнім часом однією з проблем стало досягнення фундаментальних фізичних обмежень напiвпровiдникової технології. Зокрема, фiзичнi обмеження створюють проблеми тепловиділення та синхронiзацiї даних. [2, 3] Ці проблеми в першу чергу пов’язані з максимальною швидкістю передачі сигналів через неiдеальнi лiнiї з ненульовою ємністю. Перезарядження цієї паразитної ємності i є основна причина тепловиділення в напiвпровiдникових інтегральних схемах, а час перезаряду причина затримок передачі даних, більших, ніж в ідеальному випадку (коли єдиним обмеженням є швидкість світла). Мiнiмально можливий час передачі даних звичайно зумовлює максимально можливу тактову частоту мікропроцесора. Таким чином, можна зробити висновок, що необмежене  
зростання продуктивності за рахунок збільшення тактової частоти та інженерних рушень, продуктивність яких залежить вiд частоти, неможливе.

Iнженери-розробники мiкропроцесорiв шукають рiзнi шляхи обходу вищеназваних фізичних обмежень. Серед ранніх інженерних рішень варто виділити введення конвеєра та суперскалярних iнструкцiй. Але ці розробки самі по собі не можуть дати необмежений приріст продуктивності без одночасного збільшення тактової частоти мікропроцесора. Таким чином, вони можуть забезпечити збільшення продуктивності тільки в константну кiлькiсть разів, а  
провідним фактором все ж залишається тактова частота. З іншої сторони, ці розробки підходять не однаково гарно для різних програмних застосувань. Дійсно, якщо в програмному коді є багато переходів, напрямок яких важно передбачити, то переваги конвеєра зникають (мікропроцесор вимушений скидати конвеєр при виконанні неправильно передбаченого переходу). Якщо ж програмний код важко ефективно реалізувати з огляду на особливості суперскалярних архiтектур, то i застосування суперскалярної архітектури переваг отримати не зможе.

Сучасним підходом до вирішення згаданих проблем є створення багатоядерних мiкропроцесорiв. [4] Даний пiдхiд передбачає розміщення декількох функціонально незалежних обчислювальних ядер разом з комунiкацiйним середовищем у спільному корпусi. [5] Окремі ядра є рiвноцiнними з точки зору програміста прикладних програм, що дозволяє проводити процес програмування так само, як i для класичних багатопроцесорних систем. При цьому існує кілька варiантiв реалiзацiї багатоядерностi: [5]

* Незалежні обчислювальні ядра з iндивiдуальною кеш-пам’яттю, розмiщенi на одному кристалі та пiд’єднанi до системної шини.
* Незалежні обчислювальні ядра розмiщенi на різних кристалах.
* Обчислювальні ядра, що використовують спiльнi вузли, наприклад, кеш-память або системну шину.

Звичайно, такий пiдхiд дозволяє забезпечити суб-лiнiйну масштабованість для застосункiв, що розроблені з метою виконання на багатопроцесорних системах. Звідси випливає недолік багатопроцесорних систем: програмне забезпечення має бути розроблене з урахуванням їх особливостей. В той же час, оптимiзацiї, які використовують паралелізм на рiвнi iнструкцiй (наприклад, вищезгаданий конвеєр), можна проводити повністю в автоматичному режимі, хоча i це вимагає наявності вiдповiдного оптимiзуючого компілятора. Наприклад, компілятор GCC може в автоматичному режимі оптимізувати код з використанням супер-скалярних iнструкцiй, наявних у цільовому процесорі. [6, 7]

Порівняно зі звичайними багатопроцесорними системами, багатоядерні коштують в середньому менше. Це пов’язано з тим, що встановлення на системну плату додаткових процесорних роз’ємiв та реалiзацiя комунiкацiйного середовища суттєво підвищують її ціну.

В 1999 році IBM анонсувала перший в свiтi двоядерний процесор загального призначення IBM Power4. Випущений у 2001 році, цей серверний процесор мав два ядра, тактову частоту в 1.1 ГГц і суперскалярну 64‑бітну архітектуру.[9] Так як Power4 призначався в першу чергу для високопродуктивних серверів, то IBM виробляла також багатопроцесорні модулі з кількома двоядерними процесорами Power4. Для архітектури x86 перші двоядерні процесори були представлені світу у 2005 році компаніями Intel та Advanced Micro Devices.

Важливо вiдзначити, що на сьогоднiшнiй день багатоядернiсть використовується не тільки при розробці процесорів для настільних комп’ютерiв та серверів [5], а також для управляючих мiкроконтролерiв. Одним з комерційно успішних багатоядерних процесорів, призначених не для високопродуктивних обчислень, а в першу чергу для вирішення задач управління, є Parallax Propeller [9] 8-ядерний 32-бiтний RISC процесор.

## Основні характеристики процесорів

* Лінійка

Модельний ряд, або лінійка, до якої належить процесор.

* Сокет

Тип роз'єма для установки процесора на материнській платі. Як правило, тип сокета характеризується кількістю ніжок і виробником процесора. Різні сокети відповідають різним типам процесорів.

* Ядро

Ядро - це головна частина центрального процесора (CPU). Воно визначає більшість параметрів CPU, перш за все - тип сокета (гнізда, в яке вставляється процесор), діапазон робочих частот і частоту роботи внутрішньої шини передачі даних (FSB). Ядро процесора характеризується наступними параметрами: технологічний процес, обсяг внутрішнього кеша першого і другого рівня, напруга і тепловіддача.

* Кількість ядер

Число ядер в процесорі. Нова технологія виготовлення процесорів дозволяє розмістити в одному корпусі більше одного ядра. Наявність декількох ядер значно збільшує продуктивність процесора.

* Техпроцес

Техпроцес - це масштаб технології, яка визначає розміри напівпровідникових елементів, що складають основу внутрішніх ланцюгів процесора (ці ланцюги складаються із з'єднаних відповідним чином між собою транзисторів). Удосконалення технології та пропорційне зменшення розмірів транзисторів сприяють поліпшенню характеристик процесорів.

* Тактова частота процесора.

Тактова частота - це кількість тактів (операцій) процесора в секунду. Тактова частота процесора пропорційна частоті шини (FSB). Як правило, чим вище тактова частота процесора, тим вище його продуктивність. Але таке порівняння доречно тільки для моделей однієї лінійки, оскільки, крім частоти, на продуктивність процесора впливають такі параметри, як розмір кеша другого рівня (L2), наявність і частота кеша третього рівня (L3), наявність спеціальних інструкцій та інші.

* Коефіцієнт множення

Значення коефіцієнта множення процесора, на підставі якого проводиться розрахунок кінцевої тактовою частоти процесора. Тактова частота процесора обчислюється як добуток частоти шини (FSB) на коефіцієнт множення. Майже у всіх сучасних процесорів даний параметр є заблокованим на рівні ядра і не піддається зміні.

* Напруга на ядрі

Номінальна напруга живлення ядра процесора. Цей параметр вказує напругу, яка необхідна процесору для роботи (вимірюється в вольтах). Він характеризує енергоспоживання процесора і особливо важливий при виборі CPU для мобільного, нестаціонарної системи.

* Частота шини

Частота шини даних (Front Side Bus, або FSB). Шина даних - це набір сигнальних ліній для передачі інформації в процесор і з нього. Частота шини - це тактова частота, з якою відбувається обмін даними між процесором і системною шиною комп'ютера.

* Об'єм кешу L1

Кеш-пам'ять першого рівня - це блок високошвидкісної пам'яті, розташований прямо на ядрі процесора. У нього копіюються дані, витягнуті з оперативної пам'яті. Збереження основних команд дозволяє підвищити продуктивність процесора за рахунок більш високої швидкості обробки даних (обробка з кеша швидше, ніж з оперативної пам'яті). Ємність кеш-пам'яті першого рівня невелика і обчислюється кілобайтами. Для багатоядерних моделей вказується об'єм кеш-пам'яті першого рівня для одного ядра.

* Об'єм кешу L2

Кеш-пам'ять другого рівня - це блок високошвидкісної пам'яті, що виконує ті ж функції, що і кеш L1, однак має більш низьку швидкість і більший обсяг. Якщо ви вибираєте процесор для ресурсоємних завдань, то модель з великим об'ємом кеша L2 буде кращою. Для багатоядерних процесорів вказується сумарний об'єм кеш-пам'яті другого рівня.

* Об'єм кешу L3

Інтегрована кеш-пам'ять L3 в поєднанні з швидкою системною шиною формує високошвидкісний канал обміну даними з системною пам'яттю. Як правило, кеш-пам'яті третього рівня комплектуються тільки CPU для серверних рішень або спеціальні редакції «настільних» процесорів.

* Підтримка SSE2

Технологія SSE2 включає в себе набір команд, розроблених компанією Intel на додаток до своїх попередніх технологіям SSE і MMX. Ці команди дозволяють добитися істотного приросту продуктивності в додатках, оптимізованих під SSE2. Дану технологію підтримують практично всі сучасні моделі.

* Підтримка SSE3

SSE3 - технологія, що представляє собою набір з 13 нових команд, покликаних поліпшити продуктивність процесора в ряді операцій потокової обробки даних.

* Підтримка SSE4

SSE4 - технологія, що представляє собою набір з 54 нових команд. Вони покликані збільшити продуктивність процесора в роботі з медіаконтентом, в ігрових програмах, задачах тривимірного моделювання.

* Підтримка NX Bit

NX Bit являє собою технологію, яка може запобігати виконання шкідливого коду деяких видів вірусів. Вона підтримується в операційній системі Windows XP при обов'язковій установці SP2 і у всіх 64-бітових операційних системах.

* Підтримка Virtualization Technology

Virtualization Technology дозволяє запускати на одному комп'ютері декілька операційних систем одночасно. Таким чином, за допомогою віртуалізації одна комп'ютерна система може функціонувати як кілька віртуальних систем.

* Тепловиділення

Величина тепловиділення процесора. Тепловиділення - це потужність, яку повинна відводити система охолодження, щоб забезпечити нормальну роботу процесора. Чим більше значення цього параметра, тим сильніше гріється процесор при роботі. Цей показник важливий для оверклокерів: процесор з низьким тепловиділенням легше охолоджувати, і, відповідно, його можна сильніше розігнати.

* Максимальна робоча температура

Допустима максимальна температура поверхні процесора, при якій можлива нормальна робота. Температура процесора залежить від його завантаженості і від якості тепловідводу. У холостому режимі і при нормальному охолодженні температура процесора знаходиться в межах 25-40°C, при високій завантаженості вона може досягати 60-70°C. Для процесорів з високою робочою температурою рекомендуються потужні системи охолодження.

## Загальна характеристика чотириядерних процесорів Intel Core і5

Сучасні процесори продовжують набувати різні функції, стаючи все більш складними, але і функціональними. Колись частиною процесора став раніше зовнішній співпроцесор, потім до них додали кеш-пам'ять різних рівнів, модулі керування живленням і потоками даних. За цей час у декілька етапів розширився набір виконуваних команд, що обумовило збільшення розрядності виконавчих регістрів вдвічі та навіть більше, а потім, коли стало зрозуміло, що не тільки за допомогою частоти можна збільшувати продуктивність, почалося нарощування обчислювальних ядер. З збільшенням складності і функціональності виникла необхідність у динамічному керуванні кількістю і робочими характеристиками вузлів, тому що стало неможливо забезпечувати їх постійне повне завантаження, а також це призвело до збільшення енергоспоживання, що стало невигідно. Отже, процесор стає все більш автономною і самодостатньою високоінтелектуальною системою. Черговим кроком еволюції таких систем став анонс процесорів Intel Sandy Bridge, які стали ще більш ефективними і інтелектуальними.

В процесорах Intel Core i5 використовується мікроархітектура Intel Core – динамічно масштабована 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’яті, 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). [10, 11].

Існують 3 генерації чотириядерних процесорів Intel сімейства Core i5. Кожна генерація має певні технічні особливості.

### Огляд процесорів Intel Core i5 Previous Gen[12]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Lynnfield, виконаній згідно 45-нм техпроцесу. Пропускна спроможність шини DMI 2.5ГТ/с. Процесори цієї генерації не мають вбудованого графічного контроллеру. Максимальний обсяг пам’яті, який підтримується 16 ГБ, а максимальна пропускна спроможність пам’яті становить 21 ГБ/с. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 8 МБ. До цієї генерації належать процесори: Intel Core i5-750, Intel Core i5-750S, Intel Core i5-760.

### Огляд процесорів Intel Core i5 2nd Gen[13]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Sandy Bridge, виконаній згідно 32-нм техпроцесу. Пропускна спроможність шини DMI 5ГТ/с. Майже всі процесори цієї генерації мають вбудований графічний контролер. Максимальний обсяг пам’яті, який підтримується 32 ГБ, а максимальна пропускна спроможність пам’яті становить 21 ГБ/с. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 6 МБ. До цієї генерації належать процесори: Intel Core i5-2300, Intel Core i5-2400, Intel Core i5-2400S, Intel Core i5-2500, Intel Core i5-2500K, Intel Core i5-2500S, Intel Core i5-2500T, Intel Core i5-2310, Intel Core i5-2320, Intel Core i5-2405S, Intel Core i5-2450P, Intel Core i5-2380P, Intel Core i5-2550K.

### Огляд процесорів Intel Core i5 3rd Gen[14]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Ivy Bridge, виконаній згідно 22-нм техпроцесу. Пропускна спроможність шини DMI 5ГТ/с. Майже всі процесори цієї генерації мають вбудований графічний контролер. Максимальний обсяг пам’яті, який підтримується 32 ГБ, а максимальна пропускна спроможність пам’яті становить 25,6 ГБ/с. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 6 МБ. До цієї генерації належать процесори: Intel Core i5-3330, Intel Core i5-3330S, Intel Core i5-3450, Intel Core i5-3450S, Intel Core i5-3475S, Intel Core i5-3550, Intel Core i5-3550S, Intel Core i5-3570K, Intel Core i5-3570S, Intel Core i5-3570, Intel Core i5-3570T, Intel Core i5-3470, Intel Core i5-3470S, Intel Core i5-3350P .

**Ошибка! Источник ссылки не найден.** демонструє відмінності чотириядерних процесорів сімейства Intel Core i5

Таблиця 1.1 – Порівняльна характеристика чотириядерних процесорів виробництва компанії Intel сімейства Core i5[15]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назва продукту | Intel Core i5-750 | Intel Core i5-750S | Intel Core i5-760 | Intel Core i5-2300 | Intel Core i5-2310 | Intel Core i5-2320 | Intel Core  i5-2380P | Intel Core i5-2400 | Intel Core  i5-2400S | Intel Core  i5-2405S | Intel Core  i5-2450P | Intel Core i5-2500 | Intel Core  i5-2500К | Intel Core  i5-2500S | Intel Core  i5-2500T |
| Дата запуску на ринок | III кв ’09 | I кв  ’10 | IIІ кв ‘10 | I кв  ‘11 | ІI кв  ‘11 | IIІ кв ‘11 | І кв  ‘12 | І кв  ‘11 | I кв  ‘11 | II кв  ‘11 | I кв  ‘12 | I кв  ‘11 | I кв  ‘11 | I кв  ‘11 | I кв  ‘11 |
| Тактова частота/  Частота в режимі «турбо» | 2.6/3.2  ГГц | 2.4/3.2   ГГц | 2.8/3.3 ГГц | 2.8/3.1  ГГц | 2.9/3.2  ГГц | 3/3.33 ГГц | 3.1/3.4  ГГц | 3.1/3.4 ГГц | 2.5/3.3   ГГц | 2.5/3.3   ГГц | 3.2/3.5   ГГц | 3.3/3.7 ГГц | 3.3/3.7   ГГц | 2.7/3.7   ГГц | 2.3/3.3   ГГц |
| Графічний процесор  Тактова/максимальна  частота | - | - | - | 850  МГц/  1.1ГГц | 850  МГц/  1.1ГГц | 850  МГц/  1.1ГГц | - | 850  МГц/  1.1ГГц | 850  МГц/  1.1ГГц | 850  МГц/  1.1ГГц | - | 850  МГц/  1.1ГГц | 850  МГц/  1.1ГГц | 850  МГц/  1.1ГГц | 650  МГц/  1.25ГГц |
| Кеш-пам'ять | 8 МБ | 8 МБ | 8 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ |
| Системна шина | 2.5  ГТ/с | 2.5   ГТ/с | 2.5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с |
| Літографія | 45 нм | 45 нм | 45 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм |
| Максимальне споживання енергії | 95 Вт | 82 Вт | 95 Вт | 95 Вт | 95 Вт | 65 Вт | 95 Вт | 95 Вт | 65 Вт | 65 Вт | 95 Вт | 95 Вт | 95 Вт | 65 Вт | 45 Вт |
| Максимальний розмір пам’яті | 16 ГБ | 16 ГБ | 16 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ |
| Тип пам’яті | DDR3-1066/  1333 | DDR3-1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 |
| Максимальна температура корпусу | 72.7°C | 76.7°C | 72.7°C | 72.6°C | 72.6°C | 72.6°C | 72.6°C | 72.6°C | 69.1°C | 69.1°C | 72.6°C | 72.6°C | 72.6°C | 69.1°C | 69.8°C |
| Ціна | 204у.о. | 202у.о. | 205у.о. | 177у.о. | 177у.о. | 177у.о. | 177у.о. | 184у.о. | 184у.о. | 201у.о. | 195у.о. | 205у.о. | 216у.о. | 205у.о. | 205у.о. |

Продовження Таблиці 1.1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назва продукту | Intel  Core  i5-2550К | Intel Core  i5-3330 | Intel  Core  i5-3330S | Intel Core  i5-3350P | Intel Core i5-3450 | Intel Core  i5-3450S | Intel Core i5-3470 | Intel Core  i5-3470S | Intel Core  i5-3475S | Intel Core  i5-3550 | Intel Core  i5-3570 | Intel Core  i5-3570K | Intel Core  i5-3570S | Intel Core  i5-3570T |
| Дата запуску на ринок | I кв  ’12 | IІІ кв  ’12 | IІІ кв  ’12 | IІІ кв  ’12 | IІІ кв  ’12 | IІ кв  ’12 | IІ кв  ’12 | IІ кв  ’12 | ІІ кв  ’12 | IІ кв  ’12 | IІ кв  ’12 | IІ кв  ’12 | IІ кв  ’12 | IІ кв  ’12 |
| Тактова частота/  Частота в режимі «турбо» | 3.4/3.8  ГГц | 3/3.2   ГГц | 2.7/3.2   ГГц | 3.1/3.3   ГГц | 2.7/3.2   ГГц | 2.8/3.5   ГГц | 3.2/3.6   ГГц | 2.9/3.6   ГГц | 2.9/3.6   ГГц | 3.3/3.7   ГГц | 3.4/3.8   ГГц | 3.4/3.8   ГГц | 3.1/3.8   ГГц | 2.3/3.3   ГГц |
| Графічний процесор  Тактова/максимальна  частота | - | 650  МГц/  1 ГГц | 650  МГц/  1 ГГц | - | 650  МГц/  1 ГГц | 650  МГц/  1.1 ГГц | 650  МГц/  1.1 ГГц | 650  МГц/  1.1 ГГц | 650  МГц/  1 ГГц | 650  МГц/  1.15 ГГц | 650  МГц/  1.1 ГГц | 650  МГц/  1.15 ГГц | 650  МГц/  1.15 ГГц | 650  МГц/  1.15 ГГц |
| Кеш-пам'ять | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ |
| Системна шина | 5  ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с |
| Літографія | 32 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм |
| Максимальне споживання енергії | 95 Вт | 77 Вт | 65 Вт | 69 Вт | 65 Вт | 65 Вт | 77 Вт | 65 Вт | 65 Вт | 77 Вт | 77 Вт | 77 Вт | 65 Вт | 45 Вт |
| Максимальний розмір пам’яті | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ |
| Тип пам’яті | DDR3-1066/  1333 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 | DDR3-1333/  1600 |
| Максимальна температура корпусу | 72.6°C | 67.4°C | 67.4°C | 67.4°C | 67.4°C | 69.1°C | 67.4°C | 69.1°C | 69.1°C | 67.4°C | 67.4°C | 67.4°C | 69.1°C | 69.8°C |
| Ціна | 225у.о. | 182у.о. | 177у.о. | 177у.о. | 177у.о. | 184у.о. | 184у.о. | 184у.о. | 201у.о. | 205у.о. | 205у.о. | 225у.о. | 205у.о. | 205у.о. |

## Загальна характеристика чотириядерних процесорів 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 контролер 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 відео підсистемою 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 чотириядерних процесорах Intel сімейства Core i7 були введені нові технології:

* Технологія Intel Turbo Boost максимально підвищує продуктивність ресурсномістких програм, динамічно збільшуючи продуктивність відповідно до навантаження - продуктивність вище там, де вимагається. 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 [16].
* 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.
* Intel Virtualization Technology, Intel 64 – забезпечується прямий доступ до віртуальної машини, у тому числі і для 64-розрядних операційних систем.
* Intel Trusted Execute Technology – захист системи від вірусів на апаратному рівні.

Існують 6 генерацій чотириядерних процесорів Intel сімейства Core i7. Кожна генерація має певні технічні особливості.

### Огляд процесорів Intel Core і7 Previous Gen[17]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Bloomfield i Lynnfield, виконаній згідно 45-нм техпроцесу. Процесори 9 серії використовують архітектуру Bloomfield. А також використовують інтерфейс QPI, який з’єднує процесор і інтегрований модуль пам’яті. Пропускна спроможність шини 4.8 ГТ/с. Підтримають максимальний об’єм пам’яті до 24 ГБ, з 3 каналами пам’яті. Тип пам’яті DDR3-800/1066, максимально пропускна спроможність 25.6 ГБ/с. Максимальна потужність складає 130 Вт. В свою чергу процесори 8 серії використовують архітектуру Lynnfield. А також використовують інтерфейс DMI. Пропускна спроможність шини 2.5 ГТ/с. Підтримають максимальний об’єм пам’яті до 16 ГБ, з 2 каналами пам’яті. Тип пам’яті DDR3-1066/1333, максимально пропускна спроможність 21 ГБ/с. Максимальна потужність складає 95 Вт. Графічний процесор у процесорах цієї генерації відсутній. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 8 МБ. До цієї генерації належать процесори: Intel Core i7-920, Intel Core i7-930, Intel Core i7-940, Intel Core i7-950, Intel Core i7-960, Intel Core i7-860, Intel Core i7-870, Intel Core i7-875

### Огляд процесорів Intel Core і7 Previous Gen Extreme Edition[18]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Bloomfield i Clarksfield, виконаній згідно 45-нм техпроцесу. Процесори 965 та 975 моделей використовують архітектуру Bloomfield. А також використовують інтерфейс QPI, який з’єднує процесор і інтегрований модуль пам’яті. Пропускна спроможність шини 6.4 ГТ/с. Підтримають максимальний об’єм пам’яті до 24 ГБ, з 3 каналами пам’яті. Тип пам’яті DDR3-800/1066, максимально пропускна спроможність 25.6 ГБ/с. Максимальна потужність складає 130 Вт. В свою чергу процесори 920ХМ та 940ХМ моделей використовують архітектуру Clarksfield. А також використовують інтерфейс DMI. Пропускна спроможність шини 2.5 ГТ/с. Підтримають максимальний об’єм пам’яті до 8 ГБ, з 2 каналами пам’яті. Тип пам’яті DDR3-1066/1333, максимально пропускна спроможність 21 ГБ/с. Максимальна потужність складає 55 Вт. Графічний процесор у процесорах цієї генерації відсутній. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 8 МБ. До цієї генерації належать процесори: Intel Core i7-965 Extreme Edition, Intel Core i7-975 Extreme Edition, Intel Core i7-920XM Extreme Edition, Intel Core i7-940XM Extreme Edition.

### Огляд процесорів Intel Core і7 2nd Gen[19]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Sandy Bridge, виконаній згідно 32-нм техпроцесу. А також використовують інтерфейс DMI. Пропускна спроможність шини 5 ГТ/с. До цієї генерації належать процесори: Intel Core i7-2720QM, Intel Core i7-2600, Intel Core i7-2600K, Intel Core i7-2600S, Intel Core i7-2630QM, Intel Core i7-2820QM, Intel Core i7-2635QM, Intel Core i7-2670QM, Intel Core i7-2675QM, Intel Core i7-2760QM, Intel Core i7-2860QM, Intel Core i7-2700K, Intel Core i7-3820.

### Огляд процесорів Intel Core і7 2nd Gen Extreme Edition[20]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Sandy Bridge, виконаній згідно 32-нм техпроцесу. А також використовують інтерфейс DMI. Пропускна спроможність шини 5 ГТ/с. Підтримають максимальний об’єм пам’яті до 32 ГБ, з 2 каналами пам’яті. Тип пам’яті DDR3-1066/1333/1600, максимально пропускна спроможність 25.6 ГБ/с. Максимальна потужність складає 55 Вт. Графічний процесор Intel HD 3000 Graphics, яка використовує всі 12 конвеєрів і працює на номінальній частоті 650 МГц, прискорюючись при високому навантаженні до 1,3 ГГц. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 8 МБ. До цієї генерації належать процесори: Intel Core i7-2920ХМ Extreme Edition, Intel Core i7-2960XM Extreme Edition.

### Огляд процесорів Intel Core і7 3rd Gen[21]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Ivy Bridge, виконаній згідно 22-нм техпроцесу. А також використовують інтерфейс DMI. Пропускна спроможність шини 5 ГТ/с. Графічний процесор Intel HD 4000 Graphics, яка використовує всі 16 конвеєрів і працює на номінальній частоті 650 МГц. До цієї генерації належать процесори: Intel Core i7-3820QМ, Intel Core i7-3720QM, Intel Core i7-3610QM, Intel Core i7-3615QM, Intel Core i7-3612QM, Intel Core i7-3770K, Intel Core i7-3770S, Intel Core i7-3770T, Intel Core i7-3612QE, Intel Core i7-3610QE, Intel Core i7-3770, Intel Core i7-3612QM, Intel Core i7-3840QM, Intel Core i7-3740QM, Intel Core i7-3632QM, Intel Core i7-3630QM.

### Огляд процесорів Intel Core і7 3rd Gen Extreme Edition[22]

Чотириядерні процесори, які представлені цією генерацією засновані на архітектурі Ivy Bridge, виконаній згідно 22-нм техпроцесу. А також використовують інтерфейс DMI. Пропускна спроможність шини 5 ГТ/с. Підтримають максимальний об’єм пам’яті до 32 ГБ, з 2 каналами пам’яті. Тип пам’яті DDR3-1333/1600, максимально пропускна спроможність 25.6 ГБ/с. Максимальна потужність складає 55 Вт. Графічний процесор у процесорах цієї генерації відсутній. Розподіл кеш-пам'яті у процесорах цієї архітектури розроблена за таким принципом : кеш-пам'ять першого рівня становить по 64 КБ на ядро, з яких 32 КБ призначає для кешування даних і стільки ж для інструкцій, причому в другій частині відведене місце і для кешу декодованих мікрооперацій; кеш-пам'ять другого рівня має об'єм по 256 КБ на кожне ядро; кеш третього рівня є загальною для всього процесора і його об'єм дорівнює 8 МБ. До цієї генерації належать процесори: Intel Core i7-3920XM Extreme Edition, Intel Core i7-3940XM Extreme Edition.

**Ошибка! Источник ссылки не найден.**2 демонструє відмінності чотириядерних процесорів сімейства Intel Core i7

Таблиця 1.2 – Порівняльна характеристика чотириядерних процесорів виробництва компанії Intel сімейства Core i7[23]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назва продукту | Intel Core i7-870 | Intel Core  i7-875K | Intel Core i7-930 | Intel Core i7-960 | Intel Core i7-975 | Intel Core i7-940ХМ | Intel Core  i7-2600K | Intel Core i7-2670QM | Intel Core i7-2675QM | Intel Core i7-2700K | Intel Core i7-2760QM | Intel Core i7-2720QM | Intel Core i7-2820QM | Intel Core i7-2860QM |
| Дата запуску на ринок | III кв  ’09 | IІ кв  ‘10 | I кв  ’10 | IV кв ’09 | ІI кв  ’09 | IIІ кв  ’10 | І кв  ‘11 | ІV кв  ‘11 | IV кв  ‘11 | IV кв  ‘11 | IV кв  ‘11 | I кв  ‘11 | I кв  ‘11 | IV кв  ‘11 |
| Тактова частота/  Частота в режимі «турбо» | 2.9/3.6   ГГц | 2.9/3.6 ГГц | 2.8/3.0  ГГц | 3.2/3.4  ГГц | 3.3/3.6  ГГц | 2.13/3.33 ГГц | 3.4/3.8  ГГц | 2.2/3.1   ГГц | 2.2/3.1   ГГц | 3.5/3.9   ГГц | 2.4/3.5   ГГц | 2.2/3.3   ГГц | 2.3/3.4   ГГц | 2.5/3.6   ГГц |
| Графічний процесор  Тактова/максимальна  частота | - | - | - | - | - | - | - | 650  МГц/  1.1ГГц | 650  МГц/  1.2 ГГц | 850  МГц/  1.35 ГГц | 650  МГц/  1.3 ГГц | 650  МГц/  1.3ГГц | 650  МГц/  1.3ГГц | 650  МГц/  1.3ГГц |
| Кеш-пам'ять | 8 МБ | 8 МБ | 8 МБ | 8 МБ | 8 МБ | 8 МБ | 8 МБ | 6 МБ | 6 МБ | 8 МБ | 6 МБ | 6 МБ | 8 МБ | 6 МБ |
| Системна шина | 2.5   ГТ/с | 2.5   ГТ/с | 4.8   ГТ/с | 4.8  ГТ/с | 6.4   ГТ/с | 2.5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с |
| Літографія | 45 нм | 45 нм | 45 нм | 45 нм | 45 нм | 45 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм | 32 нм |
| Максимальне споживання енергії | 95 Вт | 95 Вт | 130 Вт | 130 Вт | 130 Вт | 55 Вт | 95 Вт | 45 Вт | 45 Вт | 95 Вт | 45 Вт | 45 Вт | 45 Вт | 45 Вт |
| Максимальний розмір пам’яті | 16 ГБ | 16 ГБ | 24 ГБ | 24 ГБ | 24 ГБ | 8 ГБ | 32 ГБ | 32 ГБ | 16 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ |
| Тип пам’яті | DDR3-1066/  1333 | DDR-3 1066/  1333 | DDR-3 800/  1066 | DDR3-800/  1066 | DDR-3 800/  1066 | DDR-3 1066/  1333 | DDR-3 1066/  1333/  1600 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333 | DDR-3 1066/  1333/  1600 | DDR-3 1066/  1333/  1600 | DDR-3 1066/  1333/  1600 | DDR-3 1066/  1333/  1600 |
| Максимальна температура корпусу | 72.7°C | 67.9°C | 67.9°C | 67.9°C | 67.9°C | 72.9°C | 72.6°C | 72.6°C | 72.9°C | 72.6°C | 72.9°C | 72.9°C | 72.9°C | 72.9°C |
| Ціна | 294у.о. | 353у.о. | 294у.о. | 294у.о. | 990у.о. | 970у.о. | 317у.о. | 378у.о. | 378у.о. | 332у.о. | 378у.о. | 378у.о. | 568у.о. | 378у.о. |

Продовження Таблиці 1.2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назва продукту | Intel Core i7-3820 | Intel Core i7-2960ХМ | Intel  Core i7-3610QM | Intel  Core i7-3615QM | Intel  Core i7-3630QM | Intel Core i7-3632QM | Intel  Core i7-3720QM | Intel  Core i7-3740QM | Intel  Core i7-3770K | Intel  Core i7-3820QM | Intel  Core i5-3840QM | Intel  Core i5-3920XM | Intel  Core i5-3940XM |
| Дата запуску на ринок | I кв  ’12 | IV кв  ’11 | II кв  ‘12 | ІI кв  ‘12 | ІII кв  ‘12 | IIІ кв  ‘12 | II кв  ‘12 | III кв  ‘12 | II кв  ‘12 | II кв  ‘12 | III кв  ‘12 | II кв  ‘12 | III кв  ‘12 |
| Тактова частота/  Частота в режимі «турбо» | 3.6/3.8  ГГц | 2.7/3.7   ГГц | 2.3/3.3  ГГц | 2.3/3.3  ГГц | 2.4/3.4  ГГц | 2.2/3.2 ГГц | 2.6/3.6   ГГц | 2.7/3.7   ГГц | 3.5/3.9   ГГц | 2.7/3.7 ГГц | 2.8/3.8   ГГц | 2.9/3.8   ГГц | 3.0/3.9   ГГц |
| Графічний процесор  Тактова/максимальна  частота | - | 650  МГц/  1.3ГГц | 650  МГц/  1.1ГГц | 650  МГц/  1.2ГГц | 650  МГц/  1.15ГГц | 650  МГц/  1.15ГГц | 650  МГц/  1.25ГГц | 650  МГц/  1.3ГГц | 650  МГц/  1.15ГГц | 650  МГц/  1.25ГГц | 650  МГц/  1.3ГГц | 650  МГц/  1.3ГГц | 650  МГц/  1.35ГГц |
| Кеш-пам'ять | 10 МБ | 8 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 6 МБ | 8 МБ | 8 МБ | 8 МБ | 8 МБ | 8 МБ |
| Системна шина | 5  ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с | 5   ГТ/с |
| Літографія | 32 нм | 32 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм | 22 нм |
| Максимальне споживання енергії | 130 Вт | 55 Вт | 45 Вт | 45 Вт | 45 Вт | 35 Вт | 45 Вт | 45 Вт | 77 Вт | 45 Вт | 45 Вт | 55 Вт | 55 Вт |
| Максимальний розмір пам’яті | 64 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ | 32 ГБ |
| Тип пам’яті | DDR3-1066/  1333/  1600 | DDR3-1066/  1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 | DDR-3 1333/  1600 |
| Максимальна температура корпусу | 66.8°C | 76.9°C | 76.9°C | 76.9°C | 76.9°C | 76.9°C | 76.9°C | 76.9°C | 67.4°C | 76.9°C | 76.9°C | 76.9°C | 76.9°C |
| Ціна | 294у.о. | 1096у.о. | 378у.о. | 378у.о. | 378у.о. | 378у.о. | 378у.о. | 378у.о. | 332у.о. | 568у.о. | 568у.о. | 1096у.о. | 1096у.о. |

## Висновки до розділу 1

* Сучасним способом підвищення продуктивності мікропроцесора є збільшення кількості ядер в одному корпусі.
* Компанія Intel випускає чотириядерні процесори для серверів та для настільних комп’ютерів.
* Чотириядерні процесори модельних рядів Intel Core i5 та Intel Core i7 орієнтовані на використання в системах, що забезпечують швидку і точну обробку великої кількості інформації: аналітичні обчислення для бізнес моделей, робота з великими базами даних, розрахунок масиву астрономічних даних, а також де необхідно обробляти дані в декілька потоків одночасно тощо
* Мікропроцесори серії Intel Core  i7 хоч і мають достатній обчислювальний потенціал, але дуже дорогі в порівнянні з процесорами модельного ряду Intel Core  i5.
* Процесор Intel Core i7-3820[24] забезпечує серед усіх розглянутих модельних рядів найкраще поєднання продуктивності, вбудованих функцій та рентабельності, роблячи обчислювальні центри більш гнучкими та ефективними.

# РОЗРОБКА ПРОГРАМИ ПРГ1 ДЛЯ ПКС СП

## Розробка паралельного математичного алгоритму

Паралельний математичний алгоритм відповідно до рекомендованої методики [**Ошибка! Источник ссылки не найден.**] можна подати у вигляді наступних двох етапів:

1. 
2. ,

де:

* ;
*  –  елементів вектора ;
*  –  елементів вектора ;
*  –  елементів вектора ;
*  –  елементів вектора ;
*  –  рядків матриці ;

Спільні ресурси: .

Проаналізуємо розроблений паралельний математичний алгоритм із застосуванням концепції необмеженого паралелізму.

Для оцінки необхідного часу обчислень використаємо теорему Мунро-Петерсона, яка для компютерної системи з необмеженим числом процесорів формулюється так: якщо виконується обчислення скалярної величини, яке потребує ** бінарних операцій, то необхідний час обчислень :

.

Для обчислення  необхідно виконати  додавання на кожен елемент, що обчислюється. Тому час виконання буде:

.

Для операції  необхідно виконати  множення та  додавань на кожен елемент, що обчислюється. Тому час виконання буде:

.

Сумарний час виконання всіх двох етапів обчислень буде виражатись наступною формулою:

.

## Розробка алгоритмів процесів

Оскільки розроблюване програмне забезпечення є масштабованим і працює на системі із кількістю процесорів , то зручним варіантом реалізації є написання єдиного алгоритму для всіх задач.

|  |  |
| --- | --- |
| Задача T1 | КД і ТС |
| Введення |  |
| **Сигнал** про завершення вводу | S2,3,4-1 |
| **Чекати** на завершення вводу в інших процесах | W2,3,4-1 |
| Обчислення |  |
| **Сигнал** про завершення обчислення | S2,3,4-2 |
| **Чекати** на завершення обчислення  в інших процесах  Копіювати | W2,3,4-2  КД |
| Копіювати  9. Обчислення | КД |
| 10. **Чекати** про завершення обчислення  в інших процесах  11. Виведення результату | W2,3,4-3 |
| Задача T2 | КД і ТС |
| Введення |  |
| **Сигнал** про завершення вводу | S1,3,4-1 |
| **Чекати** на завершення вводу в інших процесах | W1,3,4-1 |
| Обчислення |  |
| **Сигнал** про завершення обчислення | S1,3,4-2 |
| **Чекати** на завершення обчислення  в інших процесах  Копіювати | W1,3,4-2  КД |
| Копіювати  9. Обчислення | КД |
| 10. **Сигнал** про завершення обчислення  задачі Т1 | S1-3 |
| Задача T3 | КД і ТС |
| Введення |  |
| **Сигнал** про завершення вводу | S1,2,4-1 |
| **Чекати** на завершення вводу в інших процесах | W1,2,4-1 |
| Обчислення |  |
| **Сигнал** про завершення обчислення | S1,2,4-2 |
| **Чекати** на завершення обчислення  в інших процесах  Копіювати | W1,2,4-2  КД |
| Копіювати  9. Обчислення | КД |
| 10. **Сигнал** про завершення обчислення  задачі Т1 | S1-3 |
| Задача T4 | КД і ТС |
| Введення |  |
| **Сигнал** про завершення вводу | S1,2,3-1 |
| **Чекати** на завершення вводу в інших процесах | W1,2,3-1 |
| Обчислення |  |
| **Сигнал** про завершення обчислення | S1,2,3-2 |
| **Чекати** на завершення обчислення  в інших процесах  Копіювати | W1,2,3-2  КД |
| Копіювати  9. Обчислення | КД |
| 10. **Сигнал** про завершення обчислення  задачі Т1 | S1-3 |

## Розробка схеми взаємодії процесів

На основі алгоритму для всіх задач, наведеному в попередньому розділі, було розроблена структурна схема взаємодії задач (рис.

Рисунок 2.1‑

Рисунок 2.2). Схема розділена на два рисунки для зручного сприйняття. Вона дозволяє наочно контролювати зв’язки належних точок синхронізації (сигналів S та очікувань W). Графічне зображення взаємодії задач дозволяє виявити тупикові ситуації в програмі у випадку, коли точка синхронізації W не буде повязана з належною точкою синхронізації S. Крім того, на структурній схемі уводяться також семафори і події, що будуть використовуватись в програмі.

Для демонстрації взаємодій між задачами вибрано чотири задачі: T1, T2, T3 та T4 . Задачі T1, T2, T3, T4 вводять дані, тому з ними взаємодіють всі інші задачі (синхронізація по вводу). Всі задачі виконують обчислення і синхронізуються зі всіма задачами по обчисленню.

На структурній схемі взаємодії задач уведено такі засоби для вирішення задачі взаємного виключення та синхронізації:

* множинний семафор Sem\_B для синхронізації із завершенням вводу в T1
* множинний семафор Sem\_С\_D для синхронізації із завершенням вводу в T2
* множинний семафор Sem\_MO\_alpha для синхронізації із завершенням вводу в T3
* множинний семафор Sem\_ME для синхронізації із завершенням вводу в T4
* множинний семафор Sem\_end\_T1 для синхронізації із завершенням обчислень  в T1
* множинний семафор Sem\_end\_T2 для синхронізації із завершенням обчислень  в T2
* множинний семафор Sem\_end\_T3 для синхронізації із завершенням обчислень  в T3
* множинний семафор Sem\_end\_T4 для синхронізації із завершенням обчислень  в T4
* масив подій Еvent\_end[3] для синхронізації із завершенням обчислення  в Т2, Т3, Т4
* критична секція CS\_MO\_E для керування доступом до спільних ресурсів ;
* мютекс Mut\_alpha для керування доступом до спільного ресурсу 



Рисунок 2.1 – Структурна схема взаємодії задач для ПРГ1



Рисунок 2.2 – Структурна схема взаємодії задач для ПРГ1

## Розробка програми ПРГ1

Програма написана на мові C++ з використанням бібліотеки Win32 та складається з основного модуля: основного prg1.cpp.

Основний модуль prg1.cpp містить дві функції:

* main – точка входу в програму, формує ідентифікатори task задач, запускає задачі;
* inputVector – процедура для заповнення вектора цілим числом value;
* inputMatrix – процедура для заповнення матриці цілим числом value;
* output – процедура для виведення матриці і вектора;
* calculation – процедура обчислення вектора ;

В основному модулі об’явлені такі змінні: N – установка розмірності матриць, P – налаштування кількості процесорів. Ініціалізація цих змінних виконується при запуску користувачем програми через параметри командного рядка.

Лістинг програми ПРГ1 наведено у додатку Додаток Г. .

## Тестування програми ПРГ1

Для тестування використовувалась паралельна обчислювальна система з наступними апаратними характеристиками:

* процесор: Intel Core i5 - 2400 (4 ядра по 3.1 Ггц, 6 МБ кешу третього рівня);
* оперативна пам'ять: DDR3 1333 МГц, 3327 МБ.

В якості програмного забезпечення виступали:

* операційна система: Microsoft Windows 7 x86 SP1 (Version 6.7.7601);
* середовище розробки і компіляції C++ програми: Microsoft Visual Studio 2010 (Version 10.0.30319.1 RTMRel).

Для вимірювання часу виконання програми використовувався високоточний таймер time вбудований в Python 2.7.4.

Тестування проводилось в автоматичному режимі за допомогою скрипта, який запускав програму із заданою розмірністю матриці на заданій кількості процесорів за допомогою вбудованого модуля subprocess, який вмикав необхідну кількість ядер. Результати тестування виводились на екран. Код скрипта тестування приведений у додатку Додаток Д. .

Для оцінки ефективності програми використовуються коефіцієнти прискорення та ефективності.

Коефіцієнт прискорення  показує скорочення часу виконання паралельної програми в паралельній системі з ** процесорами ** в порівнянні з часом виконання послідовної програми в однопроцесорній системі :



Коефіцієнт ефективності  застосування комп’ютерної системи показує ступінь використання ** процесорів системи:



Результати тестування і проведених досліджень ефективності розробленої програми наведено в таблицях Таблиця 2.1‑Таблиця 2.3.

Таблиця 2.1 – Час виконання програми ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N |  |  |  |  |
| 800 | 3982 | 2001 | 1434 | 1148 |
| 1600 | 39232 | 19625 | 13974 | 10952 |
| 2400 | 141126 | 71138 | 49711 | 39176 |

На основі даних із таблиці Таблиця 2.1 виконано розрахунок значень коефіцієнтів прискорення, які наведені в таблиці Таблиця 2.2.

Таблиця 2.2 – Коефіцієнти прискорення для програми ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | |
| N | 1 | 2 | 3 | 4 |
| 800 | 1,00 | 1,990 | 2,776 | 3,468 |
| 1600 | 1,00 | 1,999 | 2,807 | 3,582 |
| 2400 | 1,00 | 1,983 | 2,838 | 3,602 |

Коефіцієнти ефективності (таблиця Таблиця 2.3) обчислено за даними таблиці Таблиця 2.2.

Таблиця 2.3 – Коефіцієнти ефективності для програми ПРГ1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | |
| N | 1 | 2 | 3 | 4 |
| 800 | 100,00% | 99,5% | 92,561% | 86,716% |
| 1600 | 100,00% | 99,954% | 93,583% | 89,57% |
| 2400 | 100,00% | 99,191% | 94,63% | 90,058% |

Використовуючи таблиці Таблиця 2.2‑Таблиця 2.3 побудовано графіки зміни коефіцієнтів прискорення і ефективності в залежності від *N* і .



Рисунок 2.3 – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер при N=800



Рисунок 2.4 – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер при N=1600



Рисунок 2.5 – Графік зміни коефіцієнту прискорення програми ПРГ1 в залежності від кількості ядер при N=2400



Рисунок 2.6 – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер при N=800



Рисунок 2.7 – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер при N=1600



Рисунок 2.8 – Графік зміни коефіцієнту ефективності програми ПРГ1 в залежності від кількості ядер при N=2400

## Висновки до розділу 2

Виконано розробку програми ПРГ1 для ПКС СП з використанням мови С++ і засобів синхронізації з бібліотеки Win32. Тестування програми показало наступне:

* Використання багатоядерної ПКС та програми ПРГ1 забезпечує скорочення часу обчислення заданої математичної задачі. Значення  лежать в межах від 1,98 до 3,586.
* Максимальне значення  забезпечує ПКС з та 
* Мінімальне значення  виявлено у ПКС з  та .
* З ростом N для та   збільшується.
* Значення  змінюються від 86,716% до 99,954%.
* Найефективніше програма ПРГ1 використовує ПКС з . При цьому .
* Зі зростанням  від 1 до 4  лінійно спадає від 100,0% до 86,716% при .

# РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП

## Розробка паралельного математичного алгоритму

Паралельний математичний алгоритм відповідно до рекомендованої методики [25] можна подати у вигляді наступних двох етапів:

1. 
2. ,

де:

* ;
*  –  елементів вектора ;
*  –  елементів вектора ;
*  –  елементів вектора ;
*  –  елементів вектора ;
*  –  рядків матриці ;

Проаналізуємо розроблений паралельний математичний алгоритм із застосуванням концепції необмеженого паралелізму.

Для оцінки необхідного часу обчислень використаємо теорему

Мунро-Петерсона, яка для компютерної системи з необмеженим числом процесорів формулюється так: якщо виконується обчислення скалярної величини, яке потребує ** бінарних операцій, то необхідний час обчислень :

.

Для обчислення  необхідно виконати  додавання на кожен елемент, що обчислюється. Тому час виконання буде:

.

Для операції  необхідно виконати  множення та  додавань на кожен елемент, що обчислюється. Тому час виконання буде:

.

Сумарний час виконання всіх двох етапів обчислень буде виражатись наступною формулою:

.

## Розробка алгоритмів процесів

Задача T1

1. **Прийняти**  від задачі T3.
2. **Прийняти**  від задачі T4.
3. Обчислення .
4. **Прийняти**  від задачі T2.
5. **Прийняти**  від задачі T3.
6. **Прийняти**  від задачі T4.
7. **Передати**  задачі T2.
8. **Передати**  задачі T3.
9. **Передати**  задачі T4.
10. Обчислення .
11. **Передати** задачі T4 результат .

Задача T2

1. **Прийняти**  від задачі T4.
2. **Прийняти**  від задачі T3.
3. Обчислення .
4. **Передати**  задачі T1.
5. **Прийняти**  від задачі T1.
6. Обчислення .
7. **Передати** задачі T4 результат .

Задача T3

1. **Введення** 
2. **Передати**  задачі T1.
3. **Передати**  від задачі T2.
4. **Прийняти** від задачі Т4.
5. **Передати**  **задачі Т4.**
6. Обчислення .
7. **Передати**  задачі T1.
8. **Прийняти**  від задачі T1.
9. Обчислення .
10. **Передати** задачі T4 результат .

Задача T4

1. Введення .
2. **Передати**  задачі T2.
3. **Передати**  задачі T1.
4. **Передати**  задачі T3.
5. **Прийняти**  від задачі T3.
6. Обчислення .
7. **Передати**  задачі T1.
8. **Прийняти**  від задачі T1.
9. Обчислення .
10. **Прийняти** результат від задачі Т1.
11. **Прийняти** результат від задачі Т2.
12. **Прийняти** результат від задачі Т3.
13. Виведення результату 

## Розробка схеми взаємодії процесів

Схема взаємодії задач приведена на рисунку Рисунок 3.1.



Рисунок 3.1 – Структурна схема взаємодії задач для ПРГ2

## Розробка програми ПРГ2

Програма написана на мові Ada з використанням механізму організації взаємодії Rendezvous та складається з одного основного модуля: prg2.adb

Основний модуль prg2.adb містить тільки функцію main, в якій і зосереджена основна логіка програми: пересилання і отримання повідомлень між задачами. За допомогою оператора входу entry і прийняття виклику входу accept. В основному модулі prg2.adb у специфікаціях задач були описані наступні входи:

* Send\_Vectors\_Const – вхід для отримання векторів  і константи , які були введені в задачі Т4;
* Send\_Matrixes – вхід для отримання матриць , які були введені в задачі Т3;
* Send\_CalcE2 – вхід для отримання проміжного результату вектору  від задачі Т2;
* Send\_CalcE3 – вхід для отримання проміжного результату вектору  від задачі Т3;
* Send\_CalcE4 – вхід для отримання проміжного результату вектору  від задачі Т4;
* Send\_E – вхід для отримання вектору Е від задачі Т1;
* Send\_Result1 – вхід для отримання вектору  від задачі Т1;
* Send\_Result2 – вхід для отримання вектору  від задачі Т2;
* Send\_Result3 – вхід для отримання вектору  від задачі Т3;

Лістинг програми ПРГ2 наведено у додатку Додаток К. .

## Тестування програми ПРГ2

Для тестування ПРГ2 використовувалось те ж саме апаратне і програмне забезпечення, що і для програми ПРГ1.

Код скрипта для автоматичного тестування програми ПРГ2 поданий у додатку Додаток Л. .

Результати тестування і проведених досліджень ефективності розробленої програми наведено в таблицях Таблиця 3.1‑Таблиця 3.3.

Таблиця 3.1 – Час виконання програми ПРГ2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N |  |  |  |  |
| 800 | 4921 | 2465 | 1763 | 1492 |
| 1600 | 54402 | 27311 | 19386 | 15858 |
| 2400 | 198464 | 100084 | 69843 | 57624 |

На основі даних із таблиці Таблиця 3.1 виконано розрахунок значень коефіцієнтів прискорення, які наведені в таблиці Таблиця 3.2.

Таблиця 3.2 – Коефіцієнти прискорення для програми ПРГ2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | |
| N | 1 | 2 | 3 | 4 |
| 800 | 1,00 | 1,996 | 2,791 | 3,298 |
| 1600 | 1,00 | 1,991 | 2,806 | 3,43 |
| 2400 | 1,00 | 1,982 | 2,841 | 3,444 |

Коефіцієнти ефективності (таблиця Таблиця 3.3) обчислено за даними таблиці Таблиця 3.2.

Таблиця 3.3 – Коефіцієнти ефективності для програми ПРГ2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Кількість процесорів (P) | | | |
| N | 1 | 2 | 3 | 4 |
| 800 | 100,00% | 99,817% | 93,042% | 82,456% |
| 1600 | 100,00% | 99,597% | 93,541% | 85,764% |
| 2400 | 100,00% | 99,148% | 94,719% | 86,103% |

Використовуючи таблиці Таблиця 3.2‑Таблиця 3.3 побудовано графіки зміни коефіцієнтів прискорення і ефективності в залежності від *N* і .



Рисунок 3.2 – Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер при N=800



Рисунок 3.3 – Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер при N=1600



Рисунок 3.4 – Графік зміни коефіцієнту прискорення програми ПРГ2 в залежності від кількості ядер при N=2400



Рисунок 3.5 – Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер при N=800



Рисунок 3.6 – Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер при N=1600



Рисунок 3.7 – Графік зміни коефіцієнту ефективності програми ПРГ2 в залежності від кількості ядер при N=2400

## Висновки до розділу 3

Виконано розробку програми ПРГ2 для ПКС ЛП з використанням мови Ada і механізму організації взаємодії Rendezvous. Тестування програми показало наступне:

* Використання багатоядерної ПКС та програми ПРГ2 забезпечує скорочення часу обчислення заданої математичної задачі. Значення  лежать в межах від 1,982 до 3,444.
* Максимальне значення  забезпечує ПКС з та 
* Мінімальне значення  виявлено у ПКС з .
* З ростом N  лінійно збільшується.
* Значення  змінюються від 82,45% до 99,81%.
* Найефективніше програма ПРГ2 використовує ПКС з . При цьому .
* Зі зростанням  від 1 до 4  лінійно спадає від 99,597% до 85,764% при .

ОСНОВНІ РЕЗУЛЬТАТИ І ВИСНОВКИ ПО РОБОТІ

* Подальше підвищення тактової частоти мікропроцесорів стало неможливим із‑за досягнення деяких фізичних обмежень. Випуск багатоядерних процесорів є рішенням цієї проблеми.
* Чотириядерні процесори компанії Intel присутні тільки у лінійках серверних процесорів Core i5 та Core i7.
* Процесори модельного ряду Intel Core i5 в порівнянні з Intel Core i7 більш дешевші, але менш продуктивніші.
* Падіння коефіцієнту прискорення у програмах ПРГ1 та ПРГ2 зі збільшенням кількості ядер пов’язано в основному з недостатньою пропускною здатністю кеш-пам’яті.
* Час виконання програм ПРГ1 та ПРГ2 майже не відрізняється.
* При малій розмірності  програма для роботи з ПКС ЛП програє ПРГ1. Це пояснюється тим, що складна топологія системи збільшує час на передачу даних між процесорами і як наслідок впливає на загальний час виконання ПРГ2.
* Алгоритм роботи для ПРГ1 значно легший і при проектуванні, і при реалізації, ніж у ПРГ2. У випадку системи з локальною пам‘яттю, велика кількість повідомлень різного розміру ускладнюють систему з пересилкою повідомлень, а також підвищують вірогідність помилки при реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moore G. E. Cramming more components onto integrated circuits [Текст] // Electronics.— 1965.— Т. 38, № 8.
2. The nanoelectronic road ahead: Despite challenges, silicon offers 20 more   
   years of semiconductor progress [Електронний ресурс].— Режим доступу:   
   http://gtresearchnews.gatech.edu/newsrelease/FUTURECHIP.html.— дата звернення: 23.03.2013.— Назва з екрану.
3. Multi-core processors—the next evolution in computing [Електронний ресурс].— Режим доступу: http://www.amd.com/us-en/assets/content\_type/white\_papers\_and\_tech\_docs/Multi-Core\_Processing\_33211A.pdf. — дата звернення: 23.03.2013.— Назва з екрану.
4. From a few cores to many: a tera-scale computing research overview [Електронний ресурс].— Режим доступу:   
   http://download.intel.com/research/platform/terascale/terascale\_overview\_paper.pdf. — дата звернення: 23.03.2013.— Назва з екрану.
5. Двухъядерные процессоры Intel и AMD: теория, часть 1 [Електронний ресурс].— Режим доступу: http://www.ferra.ru/online/processors/s25920/.— дата звернення: 23.03.2013.— Назва з екрану.
6. Auto-vectorization in GCC [Electronic resource].— Access mode:   
   http://gcc.gnu.org/projects/tree-ssa/vectorization.html. — Date access: 24.03.2013.— Title from the screen.
7. Naishlos Dorit. Autovectorization in GCC [Text] // GCC developers’ summit 2004.— 2004.
8. Power4 system microarchitecture [Електронний ресурс].— Режим доступу: http://www.03.ibm.com/systems/resources/systems\_p\_hardware\_whitepapers\_power4.pdf.— дата звернення: 24.03.2013.— Назва з екрану.
9. Parallax Inc.— Propeller P8X32A Datasheet. <http://www.parallax.com/Portals/0/Downloads/docs/prod/prop/PropellerDatasheet-v1.2.pdf>.
10. Core i5 [Електронний ресурс] – Режим доступу: <http://ru.wikipedia.org/wiki/Intel_Core_i5> (дата звернення: 25.03.13). – Назва з екрану
11. Список микропроцессоров Core i5 [Електронний ресурс] – Режим доступу: <http://ru.wikipedia.org/wiki/Список_микропроцессоров_Core_i5> (дата звернення:25.03.13). – Назва з екрану.
12. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=Previous%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i5%20Processor&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
13. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=2nd%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i5%20Processors&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
14. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=3rd%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i5%20Processors&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
15. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/compare/65509,65510,69114,65515,65701,65702,68315,68316,65511,65512,65516,65518,65520,65521,64843,64844,65647,53446,53445,55446,52206,52207,52208,52209,52210,52211,52212,48496,42917,42915>. – Last access: 27.03.2013. – Title from the screen.
16. Turbo boost [Електронний ресурс] – Режим доступу: <http://ru.wikipedia.org/wiki/Intel_Turbo_Boost> (дата звернення: 25.03.13). – Назва з екрану
17. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=Previous%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i7%20Processor&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
18. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=Previous%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i7%20Extreme%20Processor&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
19. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=2nd%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i7%20Processors&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
20. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=2nd%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i7%20Extreme%20Processor&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
21. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=3rd%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i7%20Processors&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
22. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/search/advanced/?s=t&FamilyText=3rd%20Generation%20Intel%C2%AE%20Core%E2%84%A2%20i7%20Extreme%20Processor&CoreCountMin=4&CoreCountMax=4>. – Last access: 27.03.2013. – Title from the screen.
23. Compare Intel products [Electronic resource]. – Access mode: <http://ark.intel.com/ru/compare/?ids=48499,48500,41318,37151,43122,43124,41315,41316,37150,37147,37148,41447,43125,48498,49024,43126,37153,37149,43127,63698,61275,53469,53470,53474,53476,50067,52213,52214,52215,52219,52227,53463,53472,54644,53478,52237,70846,70847,71458,71459,71460,71670,64889,64891,64899,64900,64901,65523,65524,65525,65709,65710,65711,65719,67356,71096,64887>. – Last access: 27.03.2013. – Title from the screen.
24. Обзор и тестирование процессора Intel Core i7-3820 [Електронний ресурс] – Режим доступу: <http://www.overclockers.ru/lab/46505/Obzor_i_testirovanie_processora_Intel_Core_i7-3820.html> (дата звернення:28.03.13). – Назва з екрану.
25. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення: Навч. посібник [Текст]. – К.: Корнійчук, 2005. – 226 с. – ISBN 996-7599-36-1.

ДОДАТКИ

1. Структурна схема ПКС СП
2. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ1
3. Схема алгоритму процесів для програми ПРГ1
4. Лістинг програми ПРГ1

1. /\*\*

2. \* ------------------------------------------------------------------

3. \* Паралельні та розподілені обчислення

4. \* Курсова робота. ПРГ1. Бібліотека Win32

5. \*

6. \* Завдання: A = (B+C+D)\*(alpha\*MO\*ME)

7. \*

8. \* Файл: prg1.cpp

9. \* Виконав: Куріло Сергій Андрійович

10. \* Група: ІО-01

11. \* Дата: 09.03.13

12. \* ------------------------------------------------------------------

13. \*/

14.

15. #include "windows.h"

16. #include <iostream>

17. #include <fstream>

18. #include <algorithm>

19. #include <string>

20.

21. using namespace std;

22.

23. int N;

24. int P = 4;

25. int H;

26.

27. //========================================================

28.

29. int alpha;

30. int \*A = new int[N];

31. int \*B = new int[N];

32. int \*C = new int[N];

33. int \*D = new int[N];

34. int \*E = new int[N];

35. int \*\* MO = new int\*[N];

36. int \*\* ME = new int\*[N];

37. int\*\* MT = new int\*[N];

38.

39.

40. //Семафоры

41. HANDLE Sem\_B, Sem\_C\_D, Sem\_MO\_alpha, Sem\_ME, Sem\_end\_T1, Sem\_end\_T2, Sem\_end\_T3, Sem\_end\_T4;

42.

43. //Критическая секция

44. CRITICAL\_SECTION CS\_MO\_E;

45.

46. //Мютекс

47. HANDLE Mut\_alpha;

48.

49. //События

50. HANDLE Event\_end[3];

51.

52. void inputVector(int\* vec, int temp) {

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

54. vec[j] = temp;

55. }

56. }

57.

58. void inputMatrix(int\*\* matr, int temp) {

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

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

61. matr[i][j] = temp;

62. }

63. }

64. }

65.

66. void output(int\* v) {

67. if(N <= 10) {

68. cout << "\t result is:" << endl;

69. cout << "\t " << endl;

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

71. cout << v[i] << " ";

72. }

73. cout << endl;

74. }

75. }

76.

77. void output(int\*\* m) {

78. if(N <= 10) {

79. cout << "\t result is:" << endl;

80. cout << "\t " << endl;

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

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

83. cout << m[i][j] << " ";

84. }

85. cout << endl;

86. }

87. }

88. }

89.

90. void calculation(int\*\* MO, int\*\* ME, int alpha, int\* E, int task) {

91. int l = (task-1)\*H;

92. int r = task\*H;

93.

94. if(task==4){

95. r = N;

96. }

97.

98. for (int i = l; i < r; i ++){

99. A[i]=0;

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

101. MT[i][j]=0;

102. for (int k=0; k<N; k++){

103. MT[i][j]=MT[i][j]+MO[i][k]\*ME[k][j]\*alpha;

104. }

105. A[i]=A[i]+E[j]\*MT[i][j];

106. }

107. }

108. }

109.

110. //================Задача\_1=======================

111.

112. void T1() {

113. int alpha1;

114. int\* C1 = NULL;

115. int\* E1 = NULL;

116. int \*\* MO1 = NULL;

117. C1 = new int[N];

118. E1 = new int[N];

119. MO1 = new int\*[N];

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

121. MO1[i] = new int[N];

122. }

123.

124.

125. int tid = 1;

126. cout << "T1 started" << endl;

127. //1. Ввод B

128. inputVector(B,1);

129.

130. //2. Сигнал T2,T3,T4 о вводе B

131. ReleaseSemaphore(Sem\_B, 3, NULL); // S2,3,4-1

132.

133. //3. Ждать ввода в других процесах

134. WaitForSingleObject(Sem\_C\_D, INFINITE); //W2,3,4-1

135. WaitForSingleObject(Sem\_MO\_alpha, INFINITE);

136. WaitForSingleObject(Sem\_ME, INFINITE);

137.

138. //4. Счет Eh = Bh + Ch + Dh

139. for (int i = (tid-1)\*H; i < tid\*H; i ++){

140. E[i] = B[i]+C[i]+D[i];

141. }

142.

143. //5. Сигнал T2,T3,T4 об окончании счета Eh

144. ReleaseSemaphore(Sem\_end\_T1, 3, NULL); // S2,3,4-2

145.

146. //6. Ждать окончание счета Eh в других процесах

147. WaitForSingleObject(Sem\_end\_T2, INFINITE); //W2,3,4-2

148. WaitForSingleObject(Sem\_end\_T3, INFINITE);

149. WaitForSingleObject(Sem\_end\_T4, INFINITE);

150.

151. //7. Копирование alpha1 := alpha

152. WaitForSingleObject(Mut\_alpha, INFINITE);

153. alpha1=alpha;

154. ReleaseMutex(Mut\_alpha);

155.

156. //8. Копирование МО1 := МО, E1 := E

157. EnterCriticalSection(&CS\_MO\_E);

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

159. E1[i] = E[i];

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

161. MO1[i][j] = MO[i][j];

162. }

163. }

164. LeaveCriticalSection(&CS\_MO\_E);

165.

166. //9. Счет Ah = E1 \* (MO1 \* MEh \* alpha1)

167. calculation(MO1, ME, alpha1, E1, tid);

168.

169. //10. Ждать окончания счета Ah в задаче Т2, Т3, T4

170. WaitForMultipleObjects(3,Event\_end, TRUE, INFINITE); //W2,3,4-3

171.

172. //11. Вывод A

173. output(A);

174.

175. cout << "T1 finished" << endl;

176. }

177.

178. //================Задача\_2===========================

179.

180. void T2() {

181. int alpha2;

182. int\* C2 = NULL;

183. int\* E2 = NULL;

184. int \*\* MO2 = NULL;

185. C2 = new int[N];

186. E2 = new int[N];

187. MO2 = new int\*[N];

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

189. MO2[i] = new int[N];

190. }

191.

192.

193. int tid = 1;

194. cout << "T2 started" << endl;

195. //1. Ввод C, D

196. inputVector(C,1);

197. inputVector(D,1);

198.

199. //2. Сигнал T1,T3,T4 о завершении ввода

200. ReleaseSemaphore(Sem\_C\_D, 3, NULL); // S1,3,4-1

201.

202. //3. Ждать завершение ввода в других процесах

203. WaitForSingleObject(Sem\_B, INFINITE); //W1,3,4-1

204. WaitForSingleObject(Sem\_MO\_alpha, INFINITE);

205. WaitForSingleObject(Sem\_ME, INFINITE);

206.

207. //4. Счет Eh = Bh + Ch + Dh

208. for (int i = (tid-1)\*H; i < tid\*H; i ++){

209. E[i] = B[i]+C[i]+D[i];

210. }

211.

212. //5. Сигнал T1,T3,T4 об окончании счета Eh

213. ReleaseSemaphore(Sem\_end\_T2, 3, NULL); // S1,3,4-2

214.

215. //6. Ждать окончание счета Eh в других процесах

216. WaitForSingleObject(Sem\_end\_T1, INFINITE); //W1,3,4-2

217. WaitForSingleObject(Sem\_end\_T3, INFINITE);

218. WaitForSingleObject(Sem\_end\_T4, INFINITE);

219.

220. //7. Копирование alpha2 := alpha

221. WaitForSingleObject(Mut\_alpha, INFINITE);

222. alpha2=alpha;

223. ReleaseMutex(Mut\_alpha);

224.

225. //8. Копирование МО2 := МО, E2 := E

226. EnterCriticalSection(&CS\_MO\_E);

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

228. E2[i] = E[i];

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

230. MO2[i][j] = MO[i][j];

231. }

232. }

233. LeaveCriticalSection(&CS\_MO\_E);

234.

235. //9. Счет Ah = E2 \* (MO2 \* MEh \* alpha2)

236. calculation(MO2, ME, alpha2, E2, tid);

237.

238. //10. Сигнал T1 о завершении счета

239. SetEvent(Event\_end[0]);//S1-3

240. cout << "T2 finished" << endl;

241. }

242.

243. //================Задача\_3=======================

244.

245. void T3() {

246. int alpha3;

247. int\* C3 = NULL;

248. int\* E3 = NULL;

249. int \*\* MO3 = NULL;

250. C3 = new int[N];

251. E3 = new int[N];

252. MO3 = new int\*[N];

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

254. MO3[i] = new int[N];

255. }

256.

257. int tid = 2;

258. cout << "T3 started" << endl;

259.

260. //1. Ввод MO, alpha

261. inputMatrix(MO,1);

262. alpha=1;

263.

264. //2. Сигнал T1,T2,T4 о завершении ввода

265. ReleaseSemaphore(Sem\_MO\_alpha, 3, NULL); // S1,2,4-1

266.

267.

268. //3. Ждать завершение ввода в других процесах

269. WaitForSingleObject(Sem\_B, INFINITE); //W1,2,4-1

270. WaitForSingleObject(Sem\_C\_D, INFINITE);

271. WaitForSingleObject(Sem\_ME, INFINITE);

272.

273. //4. Счет Eh = Bh + Ch + Dh

274. for (int i = (tid-1)\*H; i < tid\*H; i ++){

275. E[i] = B[i]+C[i]+D[i];

276. }

277.

278. //5. Сигнал T1,T2,T4 об окончании счета Eh

279. ReleaseSemaphore(Sem\_end\_T3, 3, NULL); // S1,2,4-2

280.

281. //6. Ждать окончание счета в других процесах

282. WaitForSingleObject(Sem\_end\_T1, INFINITE); //W1,2,4-2

283. WaitForSingleObject(Sem\_end\_T2, INFINITE);

284. WaitForSingleObject(Sem\_end\_T4, INFINITE);

285.

286. //7. Копирование alpha3 := alpha

287. WaitForSingleObject(Mut\_alpha, INFINITE);

288. alpha3=alpha;

289. ReleaseMutex(Mut\_alpha);

290.

291. //8. Копирование МО3 := МО, E3 := E

292. EnterCriticalSection(&CS\_MO\_E);

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

294. E3[i] = E[i];

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

296. MO3[i][j] = MO[i][j];

297. }

298. }

299. LeaveCriticalSection(&CS\_MO\_E);

300.

301. //9. Счет Ah = E3 \* (MO3 \* MEh \* alpha3)

302. calculation(MO3, ME, alpha3, E3, tid);

303.

304. //10. Сигнал T1 о завершении счета

305. SetEvent(Event\_end[1]);//S1-3

306. cout << "T3 finished" << endl;

307. }

308.

309. //================Задача\_4=======================

310.

311. void T4() {

312. int alpha4;

313. int\* C4 = NULL;

314. int\* E4 = NULL;

315. int \*\* MO4 = NULL;

316. C4 = new int[N];

317. E4 = new int[N];

318. MO4 = new int\*[N];

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

320. MO4[i] = new int[N];

321. }

322.

323. int tid = 3;

324. cout << "T4 started" << endl;

325. //1. Ввод ME

326. inputMatrix(ME,1);

327.

328. //2. Сигнал T1,T2,T3 о завершении ввода

329. ReleaseSemaphore(Sem\_ME, 3, NULL); // S1,2,3-1

330.

331.

332. //3. Ждать завершение ввода в других процесах

333. WaitForSingleObject(Sem\_B, INFINITE); //W1,2,3-1

334. WaitForSingleObject(Sem\_C\_D, INFINITE);

335. WaitForSingleObject(Sem\_MO\_alpha, INFINITE);

336.

337. //4. Счет Eh = Bh + Ch + Dh

338. for (int i = (tid-1)\*H; i < N; i ++){

339. E[i] = B[i]+C[i]+D[i];

340. }

341.

342. //5. Сигнал T1,T2,T3 об окончании счета Eh

343. ReleaseSemaphore(Sem\_end\_T4, 3, NULL); // S1,2,3-2

344.

345. //6. Ждать окончание счета Eh в других процесах

346. WaitForSingleObject(Sem\_end\_T1, INFINITE); //W1-2

347. WaitForSingleObject(Sem\_end\_T2, INFINITE); //W2-2

348. WaitForSingleObject(Sem\_end\_T3, INFINITE); //W3-2

349.

350. //7. Копирование alpha4 := alpha

351. WaitForSingleObject(Mut\_alpha, INFINITE);

352. alpha4=alpha;

353. ReleaseMutex(Mut\_alpha);

354.

355. //8. Копирование МО3 := МО

356. EnterCriticalSection(&CS\_MO\_E);

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

358. E4[i] = E[i];

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

360. MO4[i][j] = MO[i][j];

361. }

362. }

363. LeaveCriticalSection(&CS\_MO\_E);

364.

365. //9. Счет Ah = E4 \* (MO4 \* MEh \* alpha4)

366. calculation(MO4, ME, alpha4, E4, tid);

367.

368. //10. Сигнал T1 о завершении счета

369. SetEvent(Event\_end[2]);//S1-3

370. cout << "T4 finished" << endl;

371. }

372.

373. //========================================================

374. int main(int argc, char \*\*argv) {

375. N = (int) strtoul(argv[1], 0, 0);

376. H = N / P;

377. DWORD Tid1, Tid2, Tid3, Tid4;

378. HANDLE Thread1, Thread2, Thread3, Thread4;

379.

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

381. MO[i] = new int[N];

382. ME[i] = new int[N];

383. MT[i] = new int[N];

384. }

385.

386. Event\_end[0] = CreateEvent(NULL, 0, 0, NULL);

387. Event\_end[1] = CreateEvent(NULL, 0, 0, NULL);

388. Event\_end[2] = CreateEvent(NULL, 0, 0, NULL);

389.

390. Sem\_B = CreateSemaphore(NULL, 0, 3, NULL);

391. Sem\_C\_D = CreateSemaphore(NULL, 0, 3, NULL);

392. Sem\_MO\_alpha = CreateSemaphore(NULL, 0, 3, NULL);

393. Sem\_ME = CreateSemaphore(NULL, 0, 3, NULL);

394. Sem\_end\_T1 = CreateSemaphore(NULL, 0, 3, NULL);

395. Sem\_end\_T2 = CreateSemaphore(NULL, 0, 3, NULL);

396. Sem\_end\_T3 = CreateSemaphore(NULL, 0, 3, NULL);

397. Sem\_end\_T4 = CreateSemaphore(NULL, 0, 3, NULL);

398.

399. Mut\_alpha = CreateMutex(NULL, FALSE, NULL);

400.

401. InitializeCriticalSection(&CS\_MO\_E);

402.

403. Thread1=CreateThread(NULL,0,(LPTHREAD\_START\_ROUTINE)T1,NULL,0,&Tid1);

404. Thread2=CreateThread(NULL,0,(LPTHREAD\_START\_ROUTINE)T2,NULL,0,&Tid2);

405. Thread3=CreateThread(NULL,0,(LPTHREAD\_START\_ROUTINE)T3,NULL,0,&Tid3);

406. Thread4=CreateThread(NULL,0,(LPTHREAD\_START\_ROUTINE)T4,NULL,0,&Tid4);

407.

408. CloseHandle(Thread1);

409. CloseHandle(Thread2);

410. CloseHandle(Thread3);

411. CloseHandle(Thread4);

412.

413. cin.get();

414. return 0;

415. }

1. Код скрипта для тестування ПРГ1

1. #

2. # --------------------------------------------------------------------

3. # Паралельні та розподілені обчислення

4. # Курсова робота. ПРГ1. Бібліотека Win32

5. #

6. # Скрипт для тестування

7. #

8. # Файл: runner.py

9. # Виконав: Куріло Сергій Андрійович

10. # Група: ІО-01

11. # Дата: 14.04.13

12. # --------------------------------------------------------------------

13. #

14. import time, subprocess, sys

15.

16. COMMAND = ' '.join(sys.argv[1:])

17.

18. def measure\_time(affinity, command):

19. start\_t = time.time()

20.

21. subp = subprocess.Popen('start /AFFINITY 0x%X %s' % (pow(2, affinity) - 1, command), shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

22. stderr, stdout = subp.communicate()

23. return\_code = subp.returncode

24. subp.wait()

25.

26. end\_t = time.time()

27.

28. return (end\_t - start\_t)

29.

30. for i in xrange(1, 5):

31. print 'Running %s with %d processors\n' % (COMMAND, i)

32. time\_elapsed = measure\_time(i, COMMAND)

33. print 'Time elapsed: %f' % time\_elapsed

1. Структурна схема ПКС ЛП
2. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ2
3. Схема алгоритму процесів для програми ПРГ2
4. Лістинг програми ПРГ2

1. -- ------------------------------------------------------------------

2. -- Паралельні та розподілені обчислення

3. -- Курсова робота. ПРГ2. Ada Rendezvous

4. --

5. -- Завдання: A = (B+C+D)\*(alpha\*MO\*ME)

6. --

7. -- Файл: prg2.adb

8. -- Виконав: Куріло Сергій Андрійович

9. -- Група: ІО-01

10. -- Дата: 13.04.13

11. -- ------------------------------------------------------------------

12.

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

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

15.

16. procedure Main is

17. N : Integer := Integer'Value(Argument(1)); --Size of vectors and matrixes

18. P: integer := 4; --Number of tasks

19. H: integer := N / P; --Size of subvector

20.

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

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

23.

24.

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

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

27.

28. --Task specification T1-------------------------------------------

29. task T1 is

30. entry Send\_Vectors\_Const(alpha: in Integer; Bh: in Vector\_General; Ch: in Vector\_General; Dh: in Vector\_General);

31. entry Send\_Matrixes(MO: in Matrix; MEh: in Matrix\_General);

32. entry Send\_CalcE2(Eh: in Vector\_General);

33. entry Send\_CalcE3(Eh: in Vector\_General);

34. entry Send\_CalcE4(Eh: in Vector\_General);

35. end T1;

36. ------------------------------------------------------------------

37.

38. --Task specification T2-------------------------------------------

39. task T2 is

40. entry Send\_Vectors\_Const(alpha: in Integer; Bh: in Vector\_General; Ch: in Vector\_General; Dh: in Vector\_General);

41. entry Send\_Matrixes(MO: in Matrix; MEh: in Matrix\_General);

42. entry Send\_E(E: in Vector);

43. end T2;

44. ------------------------------------------------------------------

45.

46. --Task specification T3-------------------------------------------

47. task T3 is

48. entry Send\_Vectors\_Const(alpha: in Integer; Bh: in Vector\_General; Ch: in Vector\_General; Dh: in Vector\_General);

49. entry Send\_E(E: in Vector);

50. end T3;

51. ------------------------------------------------------------------

52.

53. --Task specification T4-------------------------------------------

54. task T4 is

55. entry Send\_Matrixes(MO: in Matrix; MEh: in Matrix\_General);

56. entry Send\_E(E: in Vector);

57. entry Send\_Result1(Ah: in Vector\_General);

58. entry Send\_Result2(Ah: in Vector\_General);

59. entry Send\_Result3(Ah: in Vector\_General);

60. end T4;

61. ------------------------------------------------------------------

62.

63. --Task body T1----------------------------------------------------

64. task body T1 is

65. A1, B1, C1, D1, TV: Vector;

66. alpha1, ti: Integer;

67. ME1: Matrix;

68. MO1: Matrix;

69. E: Vector;

70. begin

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

72.

73. -- 1. Receive MO, MEh from T3

74. accept Send\_Matrixes(MO: in Matrix; MEh: in Matrix\_General) do

75. MO1 := MO;

76. ME1(MEh'Range) := MEh;

77. end Send\_Matrixes;

78.

79. -- 2. Receive Bh, Ch, Dh, alpha from T4

80. accept Send\_Vectors\_Const(alpha: in Integer; Bh: in Vector\_General; Ch: in Vector\_General; Dh: in Vector\_General) do

81. alpha1 := alpha;

82. B1(Bh'Range) := Bh;

83. C1(Ch'Range) := Ch;

84. D1(Dh'Range) := Dh;

85. end Send\_Vectors\_Const;

86.

87. -- 3. Calc Eh=Bh+Ch+Dh

88. for i in 1..H loop

89. E(i) := B1(i)+C1(i)+D1(i);

90. end loop;

91.

92. -- 4. Receive Eh from T2

93. accept Send\_CalcE2 (Eh : in Vector\_General) do

94. E(Eh'Range) := Eh;

95. end Send\_CalcE2;

96.

97. -- 5. Receive Eh from T3

98. accept Send\_CalcE3 (Eh : in Vector\_General) do

99. E(Eh'Range) := Eh;

100. end Send\_CalcE3;

101.

102. -- 6. Receive Eh from T4

103. accept Send\_CalcE4 (Eh : in Vector\_General) do

104. E(Eh'Range) := Eh;

105. end Send\_CalcE4;

106.

107. -- 7. Send E to T2

108. T2.Send\_E(E);

109. -- 8. Send E to T3

110. T3.Send\_E(E);

111. -- 9. Send E to T4

112. T4.Send\_E(E);

113.

114. -- 10. Calculate

115. for i in 1..H loop

116. for j in 1..N loop

117. TV(j) := 0;

118. for k in 1..N loop

119. TV(j) := TV(j) + MO1(k)(j) \* ME1(i)(k) \* alpha1;

120. end loop;

121. end loop;

122. ti := 0;

123. for j in 1..N loop

124. ti := ti + E(j) \* TV(j);

125. end loop;

126. A1(i):= ti;

127. end loop;

128.

129. -- 11. Send Result to T4

130. T4.Send\_Result1(A1(1..H));

131.

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

133. end T1;

134. ------------------------------------------------------------------

135.

136. --Task body T2----------------------------------------------------

137. task body T2 is

138. A2, B2, C2, D2, Eh, TV: Vector;

139. alpha2, ti: Integer;

140. ME2: Matrix;

141. MO2: Matrix;

142. E2: Vector;

143.

144. begin

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

146.

147. -- 1. Receive Bh, Ch, Dh, alpha from T4

148. accept Send\_Vectors\_Const(alpha: in Integer; Bh: in Vector\_General; Ch: in Vector\_General; Dh: in Vector\_General) do

149. alpha2 := alpha;

150. B2(Bh'Range) := Bh;

151. C2(Ch'Range) := Ch;

152. D2(Dh'Range) := Dh;

153. end Send\_Vectors\_Const;

154.

155. -- 2. Receive MO, MEh from T3

156. accept Send\_Matrixes(MO: in Matrix; MEh: in Matrix\_General) do

157. MO2 := MO;

158. ME2(MEh'Range) := MEh;

159. end Send\_Matrixes;

160.

161. -- 3. Calc Eh=Bh+Ch+Dh

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

163. Eh(i) := B2(i)+C2(i)+D2(i);

164. end loop;

165.

166. -- 4. Send Eh to T1

167. T1.Send\_CalcE2(Eh(H+1..2\*H));

168.

169. -- 5. Receive E from T1

170. accept Send\_E (E : in Vector) do

171. E2:=E;

172. end Send\_E;

173.

174. -- 6. Calculate

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

176. for j in 1..N loop

177. TV(j) := 0;

178. for k in 1..N loop

179. TV(j) := TV(j) + MO2(k)(j) \* ME2(i)(k) \* alpha2;

180. end loop;

181. end loop;

182. ti := 0;

183. for j in 1..N loop

184. ti := ti + E2(j) \* TV(j);

185. end loop;

186. A2(i):= ti;

187. end loop;

188.

189. -- 7. Send Result to T4

190. T4.Send\_Result2(A2(H+1..2\*H));

191.

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

193. end T2;

194. ------------------------------------------------------------------

195.

196. --Task body T3----------------------------------------------------

197. task body T3 is

198. A3, B3, C3, D3, Eh, TV: Vector;

199. alpha3, ti: Integer;

200. MO, ME: Matrix;

201. E3: Vector;

202. begin

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

204.

205. -- 1. Input MO, ME

206. for i in 1..N loop

207. for j in 1..N loop

208. ME(i)(j) := 1;

209. MO(i)(j) := 1;

210. end loop;

211. end loop;

212.

213. -- 2. Send MO, MEh to T1

214. T1.Send\_Matrixes(MO, ME(1..H));

215.

216. -- 3. Send MO, MEh to T2

217. T2.Send\_Matrixes(MO, ME(H+1..2\*H));

218.

219. -- 4. Receive alpha, Bh, Ch, Dh, alpha from T4

220. accept Send\_Vectors\_Const(alpha: in Integer; Bh: in Vector\_General; Ch: in Vector\_General; Dh: in Vector\_General) do

221. alpha3 := alpha;

222. B3(Bh'Range) := Bh;

223. C3(Ch'Range) := Ch;

224. D3(Dh'Range) := Dh;

225. end Send\_Vectors\_Const;

226.

227. -- 5. Send MO, MEh to T4

228. T4.Send\_Matrixes(MO, ME(3\*H+1..N));

229.

230. -- 6. Calc Eh=Bh+Ch+Dh

231. for i in 2\*H+1..3\*H loop

232. Eh(i) := B3(i)+C3(i)+D3(i);

233. end loop;

234.

235. -- 7. Send Eh to T1

236. T1.Send\_CalcE3(Eh(2\*H+1..3\*H));

237.

238. -- 8. Receive E from T1

239. accept Send\_E (E : in Vector) do

240. E3:=E;

241. end Send\_E;

242.

243. -- 9. Calculate

244. for i in 2\*H+1..3\*H loop

245. for j in 1..N loop

246. TV(j) := 0;

247. for k in 1..N loop

248. TV(j) := TV(j) + MO(k)(j) \* ME(i)(k) \* alpha3;

249. end loop;

250. end loop;

251. ti := 0;

252. for j in 1..N loop

253. ti := ti + E3(j) \* TV(j);

254. end loop;

255. A3(i):= ti;

256. end loop;

257.

258. -- 10. Send Result to T4

259. T4.Send\_Result3(A3(2\*H+1..3\*H));

260.

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

262. end T3;

263. ------------------------------------------------------------------

264.

265. --Task body T4----------------------------------------------------

266. task body T4 is

267. Eh, TV: Vector;

268. alpha, ti: Integer;

269. ME4: Matrix;

270. MO4: Matrix;

271. A, B, C, D: Vector;

272. E4: Vector;

273. begin

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

275.

276. -- 1. Input B, C, D, alpha

277. for i in 1..N loop

278. B(i) := 1;

279. C(i) := 1;

280. D(i) := 1;

281. end loop;

282. alpha := 1;

283.

284. -- 2. Send Bh, Ch, Dh to T2

285. T2.Send\_Vectors\_Const(alpha, B(H+1..2\*H), C(H+1..2\*H), D(H+1..2\*H));

286.

287. -- 3. Send Bh, Ch, Dh to T1

288. T1.Send\_Vectors\_Const(alpha, B(1..H), C(1..H), D(1..H));

289.

290. -- 4. Send Bh, Ch, Dh to T3

291. T3.Send\_Vectors\_Const(alpha, B(2\*H+1..3\*H), C(2\*H+1..3\*H), D(2\*H+1..3\*H));

292.

293. -- 5. Receive MO, MEh from T3

294. accept Send\_Matrixes(MO : Matrix; MEh : Matrix\_General) do

295. MO4 := MO;

296. ME4(MEh'Range) := MEh;

297. end Send\_Matrixes;

298.

299. -- 6. Calc Eh=Bh+Ch+Dh

300. for i in 3\*H+1..N loop

301. Eh(i) := B(i)+C(i)+D(i);

302. end loop;

303.

304. -- 7. Send Eh to T1

305. T1.Send\_CalcE4(Eh(3\*H+1..N));

306.

307. -- 8. Receive E from T1

308. accept Send\_E (E : in Vector) do

309. E4:=E;

310. end Send\_E;

311.

312. -- 9. Calculate

313. for i in 3\*H+1..N loop

314. for j in 1..N loop

315. TV(j) := 0;

316. for k in 1..N loop

317. TV(j) := TV(j) + MO4(k)(j) \* ME4(i)(k) \* alpha;

318. end loop;

319. end loop;

320. ti := 0;

321. for j in 1..N loop

322. ti := ti + E4(j) \* TV(j);

323. end loop;

324. A(i):= ti;

325. end loop;

326.

327. -- 10. Receive A3h from T1

328. accept Send\_Result1 (Ah : in Vector\_General) do

329. A(Ah'Range) := Ah;

330. end Send\_Result1;

331.

332. -- 11. Receive A3h from T2

333. accept Send\_Result2 (Ah : in Vector\_General) do

334. A(Ah'Range) := Ah;

335. end Send\_Result2;

336.

337. -- 12. Receive A3h from T3

338. accept Send\_Result3 (Ah : in Vector\_General) do

339. A(Ah'Range) := Ah;

340. end Send\_Result3;

341.

342. -- 13. Output A

343. put("A = (");

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

345. put(A(i));

346. put(", ");

347. end loop;

348. put(A(N));

349. put\_line(")");

350.

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

352. end T4;

353. ------------------------------------------------------------------

354.

355. begin

356. null;

357. end Main;

1. Код скрипта для тестування ПРГ2

1. #

2. # -------------------------------------------------------------------------

3. # Паралельні та розподілені обчислення

4. # Курсова робота. ПРГ2. Ada Rendezvous

5. #

6. # Скрипт для тестування

7. #

8. # Файл: runner.py

9. # Виконав: Куріло Сергій Андрійович

10. # Група: ІО-01

11. # Дата: 14.04.13

12. # --------------------------------------------------------------------------

13. #

14. import time, subprocess, sys

15.

16. COMMAND = ' '.join(sys.argv[1:])

17.

18. def measure\_time(affinity, command):

19. start\_t = time.time()

20.

21. subp = subprocess.Popen('start /AFFINITY 0x%X %s' % (pow(2, affinity) - 1, command), shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

22. stderr, stdout = subp.communicate()

23. return\_code = subp.returncode

24. subp.wait()

25.

26. end\_t = time.time()

27.

28. return (end\_t - start\_t)

29.

30. for i in xrange(1, 5):

31. print 'Running %s with %d processors\n' % (COMMAND, i)

32. time\_elapsed = measure\_time(i, COMMAND)

33. print 'Time elapsed: %f' % time\_elapsed