Міністерство освіти і науки України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Кантелюк -Марчак Юлія Михайлівна

Апаратні засоби для паралельного сортування даних методом злиття / Hardware for parallel sorting data by merge sort

Спеціальність 8.091501 – Комп'ютерні системи та мережі

Дипломна робота за освітньо-кваліфікаційним рівнем «магістр»

Науковий керівник
д.т.н., професор Цмоць І.Г.
_____

Дипломну роботу допущено до захисту
«___»_____ 20 __ р.
Зав. кафедри КІ
Березький О.М. _____

Тернопіль – 2017

Міністерство освіти і науки України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Освітньо-кваліфікаційний рівень магістр
Спеціальність 8.05010201 комп'ютерні системи та мережі

«Затверджую»
завідувач кафедри
д.т.н., проф. Березький О.М.
_____
"22" 10 2015 р.

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА
Кантелюк Юлії Михайлівни

1. Тема дипломної роботи "Апаратні засоби для паралельного сортування даних методом злиття"
затверджена наказом №435 від "19" жовтня 2016 р.

2. Термін здачі закінченої дипломної роботи "15" лютого 2017 р.

3. Об'єкт дослідження: Сортування даних методом злиття.

4. Апаратна реалізація сортування даних на графічному процесорі.

5. Перелік задач, які мають бути вирішені:
- дослідити сучасні методи та алгоритми паралельного сортування даних;
- дослідити апаратні засоби, за допомогою яких відсортовуються дані;
- розробити конкретизований потоковий граф алгоритму сортування методом злиття;
- розробити програму сортування даних на центральному процесорі CPU;
- розробити програму сортування даних на графічному процесорі GPU;
- виконати тестування розробленого програмного забезпечення.

6. Перелік ілюстративного матеріалу:
- тема, мета, завдання, методи досліджень, наукова новизна, практичне значення;
- актуальність;
- структура пошукової системи;
- об'єкт дослідження;
- архітектура центрального процесора для сортування даних;
- архітектура графічного процесора GPU;

– будова CPU та GPU процесорів;

– структура ПЕ з паралельним надходженням всіх розрядів даних;

– приклад алгоритму паралельного сортування Quick Sort;

– приклад алгоритму паралельне сортування даних методом злиття;

– конкретизований граф сортування даних на горизонтальну вісь X;

– конкретизований граф сортування даних на вертикальну вісь Y на о одноканального двошляхового злиття;

– основні вікна проекту на основі розширення CUDA;

– приклад сортування масиву даних методом злиття на централь процесорі;

– приклад сортування масиву даних методом злиття на графіч процесорі;

7. Консультанти по роботі

Розділ	Консультант	Підпис
Нормо-контроль	Мельник Г.М.	

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз систем пошуку даних, алгоритмів та засобів паралельного сортування даних	20. 10. 2015 – 1. 01. 2016	
2	Алгоритми сортування масивів даних методом злиття	2. 01. 2016 – 31. 05. 2016	
3	Розробка програмного забезпечення та синтез апаратного пристрою сортування методом злиття	1. 06. 2016 – 19 12. 2016	
4	Нормоконтроль, попердній захист	20.12.2016 - 21.12.2016	
5	Захист	16.02.2017	

Завдання прийняв до виконання _____ Кантелюк Ю.М.
(підпис) (прізвище та ініціали)

Керівник дипломної роботи _____ Цмоць І.Г.
(підпис) (прізвище та ініціали)

РЕЗЮМЕ

Дипломна робота на тему “Апаратні засоби для паралельного сортування даних методом злиття” на здобуття освітньо-кваліфікаційного рівня “Магістр” зі спеціальності “Комп’ютерні системи та мережі” написана обсягом 99 сторінок і містить 29 ілюстрацій, 2 таблиці, 4 додатки та 73 джерела за переліком посилань.

Метою роботи є розробка з використанням комплексного підходу програмних засобів для швидкого сортування великих масивів даних на базі графічного процесора GPU та програмної моделі CUDA.

Методи досліджень. Для розв’язання поставлених задач у дипломній роботі використано: дослідження методів і алгоритмів паралельного сортування великих масивів даних; графові моделі алгоритмів паралельного сортування масивів даних; архітектуру графічного процесора GPU та програмну модель CUDA.

Результати дослідження: розроблено конкретизований потоковий граф алгоритму сортування методом злиття, який забезпечує виявлення паралелізму та можливість управляти ним; визначено складність паралельного алгоритму сортування злиттям та його швидкодію.

Результати роботи можуть бути використані в науковій практиці, для розв’язання широкого кола задач із сортуванням даних.

Орієнтовні напрямки розвитку досліджень: паралельне сортування елементів використовується при розробці та моделюванні великого діапазону завдань, а саме: при розв’язку систем лінійних рівнянь певної розмірності, при розбитті графів, що описують точні розрахункові сітки, при стисненні сіткових функцій – результатів виконаних багатоопераційних обчислювальних експериментів. Сортування даних застосовується при обробці великих масивів інформації, в додатках, при пошуку у великих масивах експериментальних даних, тобто, для систем пошуку даних.

КЛЮЧОВІ СЛОВА: ПАРАЛЕЛЬНЕ СОРТУВАННЯ, ГРАФІЧНИЙ ПРОЦЕСОР, АПАРАТНІ ЗАСОБИ, ПОТОКОВИЙ ГРАФ, МЕТОД ЗЛИТТЯ, РЕАЛЬНИЙ ЧАС.

RESUME

Diploma work: "Hardware for parallel sorting data fusion method" for obtaining educational qualification" to education and qualification of "Master" specialty "Computer systems and networks" written 99 page volume and contains 29 illustrations, 2 tables, 4 applications and 73 sources for references.

The aim is to develop an integrated approach with the use of software to quickly sort large data sets based on GPU programming model and CUDA.

Research Methods. To solve the tasks in applied research paper, research methods and algorithms for parallel sorting of large data sets; Count model parallel sorting algorithms, data sets; GPU architecture and GPU programming model CUDA

The results: developed fleshed flow graph algorithm merge sort method that ensures parallelism detection and the ability to manage them; determined by the complexity of the parallel algorithm merge sort and its performance.

The results can be used in scientific practice, to solve a wide range of tasks from sorting data.

The estimated direction of research: parallel sorting elements used in the design and simulation of a large range of tasks, namely when solving systems of linear equations of a certain dimension, the partition graph describing the exact calculation of the grid, the grid compression functions - the results of the computational experiments multioperational. Sort the data used in the processing of large volumes of information in applications when searching in large arrays of experimental data, that is, systems for data retrieval.

KEY WORDS: DATA SORTING, GPU, HARDWARE, FLOW GRAPH, MERGE SORT, REAL TIME.

ЗМІСТ

1 АНАЛІЗ СИСТЕМ ПОШУКУ ДАНИХ, АЛГОРИТМІВ ТА ЗАСОБІВ ПАРАЛЕЛЬНОГО СОРТУВАННЯ ДАНИХ

1.1 Аналіз області застосування паралельного сортування даних

Сортування даних є одною з основних проблем обробки інформації та характеризується задачею розташування елементів невпорядкованого набору значень, в зростаючому чи спадаючому порядку. Сортування елементів масиву використовується у великій кількості додатків, при обробці певних масивів даних та ефективність сортування інформації дає змогу визначити працездатність системи в цілому. Сортування масивів являє собою операцію опрацювання інформації, за допомогою якої, елементи масиву розташовуються у певній послідовності, у відповідності до певних ознак елементів даної інформації.

Збільшення впливу сфери комп'ютерних систем у світі характеризується розвитком галузей, в певній мірі яких використовується сортування даних в паралельному режимі у реальному часі. Щоб сортування масивів даних функціонувало, можна забезпечити за допомогою спеціалізованих засобів, архітектурна будова яких апаратно відображає структуру алгоритму сортування даних та реалізована на надвелику інтегральну схему. Розробка та реалізація високопродуктивних та спеціальних засобів сортування потребують впровадження сучасних елементних ресурсів, розробку нових методів та алгоритмів сортування даних та надвеликих інтегральних структур. Режим реального часу та НВІС-реалізація алгоритмів сортування здійснюється за допомогою конвеєризації процесів сортування, апаратним відображенням структури алгоритмів у архітектуру, яка адаптована до великої кількості надходження інформації. Орієнтація структури засобів сортування на НВІС-реалізацію вимагає зменшення кількості виводів інтерфейсу та реалізацію алгоритмів на базі однотипних процесорних елементах (ПЕ) з регулярними та локальними зв'язками. Достатньої актуальності набирає питання вибору

ефективних методів і алгоритмів сортування масивів даних, орієнтованих на НВІС- реалізацію [1].

Паралельне сортування елементів використовується при розробці та моделюванні великого діапазону завдань, а саме: при розв'язку систем лінійних рівнянь певної розмірності, при розбитті графів, що описують точні розрахункові сітки, при стисненні сіткових функцій - результатів виконаних багатоопераційних обчислювальних експериментів. Певну роль сортування даних відіграє при розробці методів кешування геометричних даних, направлених на ефективну обробку двовимірних і об'ємних тіл складної конфігурації масиву. Ще більш широке застосування методи сортування даних знаходять при обробці великих масивів інформації, в додатках баз даних, при пошуку у великих масивах експериментальних даних, тобто, для систем пошуку даних.

Пошукова система має на своїй меті впорядкувати та класифікувати певну інформацію за критеріями. Основною метою певної інформаційно-пошукової системи є знаходження інформації релевантної інформації потребам людини, що здійснює пошук [2].

Пошукова система володіє роботами-пошуковими, які весь час шукають певну інформацію в мережі Інтернеті. Число роботів є значним, кожен з виконує певну функцію із знаходження та збору інформації у мережі. Зібрана інформація реєструється індексатором та представляє собою базу даних пошукової системи, що є відносно непостійною величиною. Пошукова система характеризує собою механізм, що складається із програмною та апаратної частини, яка за своїм функціоналом пристосована до пошуку потрібної інформації в мережі Інтернет, та видачу користувачу списку посилань, відповідно до його запиту. Пошукова система функціонує так само, як діє користувач при пошуку інформації та надає за його запитом матеріал швидко та достовірно. У зв'язку із зростаючими потребами користувачів, розробники пошукових машин намагаються вдосконалювати та оновлювати алгоритми і критерії пошуку, додаючи нові функції та розширення, щоб пришвидшити роботу системи.

Головними пунктами, які характеризують пошукові системи є :

- швидкість пошуку інформації характеризується тим, що до пошукових систем може надходити в секунду десятки тисяч запитів. Така завантаженість вимагає зменшення часу обробки окремого запиту, тому, пошукова машина повинна обробити запит максимально швидко, щоб не затягувати обчислення наступних запитів, коли відвідувач бажає одержати результат як омога швидше;
- повнота характеризується співвідношенням знайденої кількості за запитом інформації до повної кількості документів в мережі Інтернет, що відповідають потрібному запиту;
- точність обчислюється відношенням відповідності відшуканої інформації до запиту користувача;
- актуальність визначається часом, відколи документ опублікований у мережі, до залучення його у пошукову базу;
- наочність результатів є одним з найважливіших компонентів пошуку. Перші сторінки видачі не завжди містять лише потрібну інформацію, внаслідок нечіткості складання запитів або неточності пошуку. Різні елементи сторінки видачі пошукової системи дають змогу орієнтуватися в результатах пошуку. Користувачеві часто доводиться здійснювати додатковий пошук всередині знайденого списку [3].

Усі методи організації пошуку розділяють на дві групи. Атрибутивний пошук відноситься до першої групи пошуку. Його суть полягає у тому, що кожен документ характеризується певним набором атрибутів (полів). Дані поля змінюються для різних величин та заповнені конкретною інформацією. При пошуку із значеннями у відповідних полях, перевіряється подібність величин, що містяться в запиті. Повнотекстовий пошук і виділення видань відносяться до другої групи пошуку даних. У звичайному випадку він є списком всіх основних слів в словесній базі даних з вказівкою, в яких виданнях містяться ці слова. Є множинні індекси, в яких на найвищому рівні знаходиться словник або пошуковий індекс слова. В ньому кожному вартісному слову відповідає покажчик розташований на наступному рівні, список наявності або індекс

посилання, в якому містяться розміщення видання і, іноді, позиція слова в документів. Для реалізації повнотекстового пошуку першочергово потрібно провести індексацію, скласти індекси пошуку. [4].

Пошукові системи мають особливу структуру, відмінну від інших. Виділяють загальні основні компоненти для всіх пошукових машин. Відмінності в структурі можуть бути лише у вигляді реалізації механізмів взаємодії цих компонентів.

Всі компоненти тісно пов'язані один з одним і працюють у взаємодії, утворюючи чіткий, достатньо складний механізм роботи пошукової системи, що вимагає величезних витрат ресурсів [5].

Пошукова система (рис.1.1) для свого функціонування має пошукові роботи, що безперервно шукають оновлену інформацію в Інтернеті. Число пошукових роботів є значним, кожен з яких має різне завдання з пошуку та збору інформації. Відшукана інформація реєструється індексатором та представляє собою базу даних пошукової системи, з якої видаляються старі веб-сторінки, проте, додаються нові та актуальні з потрібною інформацією. Звернення до веб-сервера пошуковика дасть користувачу доступ до необхідних даних. Веб-сервер володіє системою видачі, тобто можливістю миттєво сортувати наявні дані і видавати релевантні відповіді.

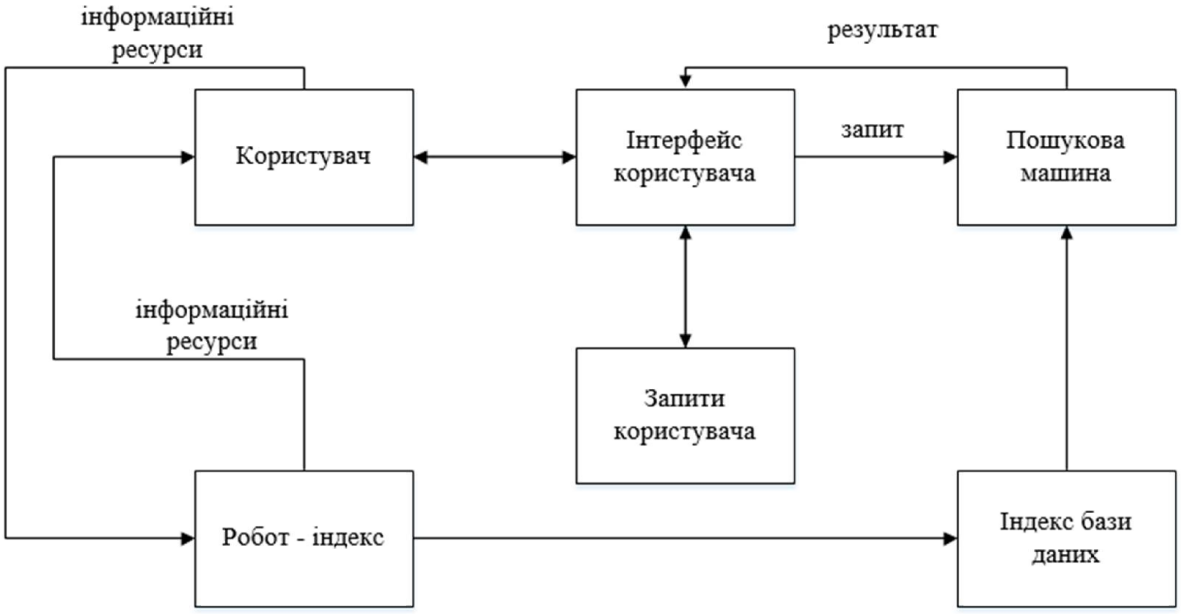

Рисунок 1.1 – Структура пошукової системи

Структура пошукової системи включає в себе наступні компоненти :

- клієнт - це програмна або апаратна складова обчислювальної системи, яка відправляє запити серверу. За допомогою протоколу, програма-клієнт має зв’язок із сервером, що дає змогу отримувати із сервера дані та інформацію, керувати даними на сервері, запускати на сервері нові процеси. В залежності від функціоналу програми, отримані від сервера дані, клієнтська програма може надавати користувачеві [6]. Програма-клієнт і програма-сервер мають змогу працювати як на одному комп'ютері, так і на двох. У останньому випадку, для обміну інформацією між програмою та клієнтом чи програмою та сервером, використовується мережеве з'єднання. Різновидом клієнтів є термінали, що являють собою робочі місця на багатокористувацьких комп’ютерах та обладнані монітором й клавіатурою, і не здатні працювати без сервера. Мережеві комп'ютери багато в чому залежать від сервера та мають спрощену структуру;
- інтерфейс користувача - це комплекс апаратних і програмних засобів, що дозволяє користувачу взаємодіяти із комп’ютером або іншим складним інструментарієм. Саме за допомогою інтерфейсу користувач сприймає програму в цілому та використовує її функціональність. Він забезпечує підтримку прийняття рішень у визначеній предметній галузі та визначає порядок використання програмних засобів і документації до нього. В дійсності, інтерфейс користувача об'єднує усі елементи і компоненти ПЗ, які здатні впливати на взаємодію користувача з програмним забезпеченням. До вищеназваних елементів належать: набір задач, які користувач розв’язує за допомогою ПЗ; елементи управління ПЗ; навігація між блоками ПЗ; візуальний дизайн вікон та екранних форм програми та інші складові Отже, інтерфейс користувача дає змогу користувачу взаємодіяти з пошуковою системою, що формує запити та перегляд результатів пошуку;
- пошукова машина –система пошуку даних, що містить дані про інформаційні складові. Визначальною ознакою пошукових машин є те, що база даних, яка включає інформацію по Web-сторінках, статтях Usenet і т.д.,

формується за допомогою робота-програми [7]. Пошук інформації в даній системі відбувається за запитом, який подається користувачем. Запит, у свою чергу, формується із набору ключових фраз або слів укладеної в лапки. Індекс формується і підтримується в актуальному стані роботами - індексами. Велика кількість пошукових систем дають змогу здійснювати пошук інформації в відомих документах, де є можливість уточнити запит, введенням додаткових термінів. Коли інтелектуальність системи на високому рівні, є функція пошуку подібних матеріалів. Для цього потрібно вказати документ в системі у якості зразка для наслідування. Адреси найбільш популярних пошукових машин :

Altavista - www.altavista.com; Excite - www.excite.com; HotBot - www.hotbot.com; Google - www.google.com; Nothern Light - www.northernlight.com Go (Infoseek) - www.go.com (infoseek.com) Fast - www.alltheweb.com;

- індекс бази даних використовується для підвищення ефективності виконання запитів користувача у системі пошуку та являє собою об'єкт бази даних. Таблиці в базі даних мають велику кількість рядків, які зберігаються у довільному порядку, і їх пошук за заданим значенням шляхом послідовного перегляду таблиці рядок за рядком може займати багато часу. Індекс формується зі значень одного чи кількох стовпчиків таблиці і вказівників на відповідні рядки таблиці і, таким чином, дозволяє знаходити потрібний рядок за заданим значенням. Унікальний індекс реалізує обмеження цілісності на таблиці, виключаючи можливість вставки значень, що повторюються. Індексів розділені на два види: кластерні й некластерні. У таблиці можливе лише розташування одного кластерного індексу та декількох некластерних. Архітектура індексу функціонує таким методом, щоб пошук відбувався з максимальною швидкістю та була змога оцінити кожен знайдений інформаційних ресурс мережі;
- запити користувача – словосполучення, слово набір символів, які вносить користувач пошукової системи для того, щоб знайти потрібну йому інформацію. Пошукові запити в переважно складаються з декількох слів. Використовуючи спеціальні алгоритми, є можливість підвищити якість пошуку

у системах що розпізнають запити користувачів, їх обробляють і показують користувачу, як результат, найбільш релевантні сторінки.

- робот-індекс-використовується для сканування Internet та обслуговування бази даних індексу в актуальному стані. Ця програма є основним джерелом інформації про стан інформаційних ресурсів мережі [9].

1.2 Аналіз алгоритмів паралельного сортування даних

Сортування є типовою проблемою обробки даних і зазвичай розуміється, як задача розміщення елементів невпорядкованого набору значень, в порядку зростання або спадання. В алгоритмах, що можуть бути в явній чи неявній формі, містяться наступні вимоги:

- детермінованість або визначеність. Полягає у тому, що стан системи визначає наступний крок роботи у кожен момент часу. Це означає, що для одних і тих же вихідних даних алгоритм видає одну і ту ж відповідь. Проте сучасне трактування алгоритмів схиляється до того, що у різних його реалізаціях має бути ізоморфний граф. З іншого боку, існують імовірнісні алгоритми. В них поточний стан системи впливає на кожен етап роботи та генерує випадкові числа. Однак при включенні методу генерації випадкових чисел в список «вихідних даних», імовірнісний алгоритм стає підвидом звичайного;
- зрозумілість – усі команди в алгоритмі повинні входити в систему команд виконавця;
- скінченність – алгоритм завершує роботу та видає результат лише коли вихідні дані задані правильно. Завершитись без результату може лише імовірнісний алгоритм, проте така ймовірність складає нуль;
- масовість – структура алгоритму повинна сприймати різноманітні набори вихідних даних;
- результативність – алгоритм повинен завершуватись результатом.

Початкові значення необхідні у роботу багатьох програм. Відповідають за перенесення цих значень у алгоритми аргументи. Аргументи - це величини, які

задаються для виконання алгоритму. Алгоритм повинен давити певний результат, не існує безрезультатних алгоритмів. Результатами вважаються величини, що отримані після виконання алгоритму. Проте аргументи та результати не єдині величини, що використовуються при складанні алгоритмів. Залучення додаткових величин залежить лише від того хто складає алгоритм. [10]. Такі величини називаються проміжними. Є декілька способів запису алгоритму. Першій із них - опис алгоритму за допомогою словесного набору. Другий спосіб - вигляд алгоритму у формі схем, формул, таблиць, малюнків тощо. Третій спосіб – подача алгоритму у вигляді блок-схеми. Даний спосіб використовується у інформатиці за допомогою набору спеціальних блоків для наочності представлення алгоритму. Четвертий метод - алгоритмічні мови (псевдокоди). Дані мови мають чітко сформований синтаксис та сильно наближені до машинної мови (мови програмування). П’ятий спосіб представлення алгоритму наближений до комп’ютера - це мови програмування. Часто на практиці виконавцем розробленого людиною алгоритму є машина, саме тому, алгоритм повинен писатись тією мовою, що буде зрозумілою для комп’ютера, тобто мовою програмування.

Для оцінки ефективності алгоритму сортування використовують оцінку по часу, який витрачається на сортування даних та пам’яті, а саме, з якою ефективністю вона використовується.

Пам'ять – середовище або фізичний пристрій для зберігання інформації протягом деякого відрізку часу. Велика кількість алгоритмів потребує виділення додаткової пам'яті для проміжного збереження даних. Спільна пам'ять є основним типом пам'яті у паралельному комп'ютері. Спільна пам'ять може представлятись як розподілена між всіма обчислювальними елементами в одному адресному просторі, чи розподілена пам'ять в якій кожен обчислювальний елемент має свій власний локальний адресний простір. У комп'ютерній науці, розподіленої спільно використовуваної пам'яті (DSM) є однією з форм архітектури пам'яті, де фізично розділені спогади можуть бути вирішені в якості одного з логічно загального адресного простору. Тут термін "загальний" не означає, що існує єдина централізована пам'ять, але що адресний

простір є "загальним" (той же фізичну адресу на двох процесорах відноситься до того ж місце в пам'яті) [1] :. 201 Розподілена глобальна адресний простір (DGAS), є аналогічний термін для широкого класу програмних і апаратних реалізацій, в яких кожен вузол кластера має доступ до загальної пам'яті на додаток до без загального доступу приватної пам'яті кожного вузла.

Систему із розподіленою пам'яттю часто називають мульти комп'ютерною, яка складається з декількох незалежних вузлів обробки з локальними модулями пам'яті, що пов'язана із загальною мережею. Програмне забезпечення системи може реалізовуватись в операційній системі в якості бібліотеки програмування та розглядатися як розширення базової архітектури віртуальної пам'яті. При реалізації в операційній системі, такі системи є прозорими для розробника. На противагу цьому, програмне забезпечення DSM системи, реалізовані в бібліотеці або на рівні мови не є прозорими. Проте, ці системи пропонують більш портативний підхід до реалізації системи DSM. Розподілена загальна пам'ять системи реалізує модель з пам'яттю для фізично розподіленої системи пам'яті [12].

Час - основний параметр, що характеризує швидкодію алгоритму. Час роботи алгоритму для тих або інших вхідних даних вимірюється в кількості елементарних операцій, або "кроків", які необхідно виконати. Проте, як правило, при цьому не потрібна велика точність. Для досить великих вхідних даних сталі множники і доданки нижчого порядку, що фігурують у виразі для точного часу роботи алгоритму, пригнічуються ефектами, що викликані збільшенням розміру вхідних даних. Розглядаючи вхідні дані досить великих розмірів для оцінки лише такої величини, як порядок зростання часу роботи алгоритму, ми тим самим вивчаємо асимптотичну ефективність алгоритмів.

Внутрішнє сортування - відсортовує масив даних, сортування якого відбувається повністю в оперативній пам'яті комп'ютера. Це можливо, коли дані, які повинні бути відсортовані, досить малі. Для сортування великих обсягів даних, необхідно провести тільки частину даних в пам'яті. Інша частина даних зазвичай сортується на великих, але більш повільних типах пам'яті, як жорсткий диск. Будь-яке читання або запис даних у повільних середовищах

може збільшити час процесу сортувальні . Ця проблема має наслідки для різних алгоритмів сортування.

Зовнішнє сортування - використовується для класу алгоритмів сортування, які можуть обробляти великі обсяги даних. Зовнішнє сортування застосовується, коли впорядковані дані не поміщаються в основну пам'ять обчислювального пристрою (зазвичай RAM) і замість цього вони повинні знаходитися на більш повільній зовнішній пам'яті (як правило, жорсткий диск). Зовнішнє сортування, як правило, використовує гібридну стратегію сортування злиттям. У фазі злиття, відсортованої субфайли об'єднані в один великий файл.

Алгоритми, подібні за обчислювальною складністю для виконання яких потрібна множина задач розпізнання називають класом складності. Задачі можна розділити на класи відповідно до складності з якою вони розв’язуються. Усі класи складності перебувають у залежності: одні містять у собі інші. Однак не можна стверджувати, що більшість включень є суворими. Усі задачі, що розв'язуються за поліноміальний час містить найнижчий клас P. Клас NP містить задачі, які розв'язуються за поліноміальний час за допомогою недетермінованої машини Тюрінга. Така машина обчислює припущення щодо розв'язку задачі паралельно перебираючи усі припущення та перевіряє дане припущення за поліноміальний час [13].

Клас P (від англ. polynomial) — підмножина завдань, для яких функціонують «швидкі» алгоритми вирішення (час виконання яких поліноміально залежить від розміру вхідних даних). Клас P включений у ширші класи складності алгоритмів. Для різної мови програмування визначається клас P наступним способом, змінивши машину Тюрінга на розробку мови програмування. Коли компілятор мови, за допомогою якого функціонує алгоритм, зменшує швидкість виконання алгоритму. Оскільки, час роботи алгоритму на машині Тюрінга менша певного многочлена від часу виконання його на мові програмування, тому класи P для даної мови програмування та для машини Тюрінга збігаються. Код на мові програмування асемблер дає можливість перетворення в машину Тюрінга з не великим поліноміальним

сповільненням, тому означення класу P для машин Тюрінга і для будь-якої існуючої мови програмування збігаються.

Клас NP (від англ. non-deterministic polynomial) – дає змогу розв’язувати множину задач розпізнавання, вирішення яких можливе за наявності певної додаткової інформації (тобто сертифіката рішень), тому є змога «швидко» (за час, що не перевершує полінома від розміру даних) перевірити розв'язання на машині Тюрінга. Еквівалентно клас NP визначається як сукупність завдань, які «швидко» вирішуються за допомогою недетермінованої машини Тюрінга.

Сортування даних дає змогу з великою швидкістю візуалізувати та удосконалити сприйняття інформації, організувати та віднайти необхідні дані й використовувати їх з великою ефективністю [14].

Під час обробки даних, яка відбувається паралельно, більша задача розділяється на декілька менших, котрі паралельно виконуються на декількох вузлах КС. Тому, вихідна задача розв’язується з більшою швидкістю. Щоб паралельна обробка даних виконувалась ефективно, все залежить від способу реалізації розподілу задачі на підзадачі, котрі виконуватимуться одночасно (паралельно).

Основними параметрами паралельної обробки даних є :

- Масштабованість та прискорення обчислень - два головні аспекти ефективної обробки даних, що виконується паралельно. Задачі, котрі виконуються в паралельний спосіб повинні обраховуватись з більшою швидкістю, аніж у послідовний спосіб. За допомогою функції «масштабованість», є можливість підвищити потужність та працездатність системи так, щоб вона спромоглась обробити зростаючі кількості даних. З використанням коефіцієнта масштабованості визначається, у скільки разів прискорення обчислень перевищує збільшення апаратних витрат;
- синхронізація - це координація обробки задач, які здійснюються паралельно. Вона необхідна для отримання точного результату обчислень. Запорукою успішної паралельної обробки є такий поділ на під задачі , коли потрібна незначна синхронізація. Внаслідок зменшення синхронізації зростає швидкодія та збільшується коефіцієнт масштабування. Потреба в синхронізації

залежить від можливості спільного використання ресурсів, обсягів і кількості задач .Головною проблемою тут є те, що незалежно зроблені зміни можуть бути несумісні одна з одною (так званий «конфлікт правок»), і навіть теоретично не існує загального способу вирішення подібних ситуацій.;

- блокування - це механізм, за допомогою якого здійснюється керування ресурсним доступом, що дозволяє досягти синхронізації виконання задач.
- передавання повідомлень - це один із ключових аспектів паралельної обробки даних, оскільки воно виконується ефективно лише в тому випадку, коли вузли системи з'єднані високопродуктивними каналами зв'язку.

Використання паралельних баз даних дає можливість для прикладних систем, що їх застосовують мати:

- високу продуктивність – із збільшенням кількості вузлів, зростає швидкість виконання задачі;
- підвищену працездатність – коли один із вузлів відмовляє в роботі, його функціональні можливості виконує інший, в цей час, система є діючою. Тому, у порівнянні із системою, що складається з одного процесора, дані є доступнішими;
- підвищену гнучкість – дає можливість уточнювати конфігурацію технічних засобів у відповідності до потреб прикладних задач, збільшувати обчислювальні потужності системи чи зменшувати їх;
- збільшення кількості користувачів - технологія паралельних баз даних дає змогу здолати наслідки обмеженості ресурсів пам'яті, що позволяє істотно збільшити кількість користувачів, які обслуговуються одночасно [16].

Алгоритми сортування інформації розділяють на класи з глобальними та локальними зв’язками між програмними елементами. Щоб оцінити зв’язки між програмними елементами, часової та апаратної складності реалізації алгоритму, розроблять решіткову модель, що у свій склад включає множину ПЕ, які у декартовій системі координат створюють точкові системи (решітки). В решітковій моделі кожному ПЕ ставиться у відповідність j часовий та просторовий і індекси, які вказують коли та де виконується кожна із операцій

алгоритму. В алгоритмах з локальними пересилками даних різниця між просторовими індексами j на кроці рекурсії обмежена деякою константою, так як в таких алгоритмах обміни здійснюються тільки між найближчими сусідніми ПЕ. Алгоритми, що при рекурсії мають рознесені просторові індекси відносяться до класу алгоритмів з глобальними зв'язками [17]. Складність реалізації обміну між ПЕ залежить як від розрядності каналів передачі даних, так і від віддалі між ними, яка визначається як різниця двох просторових індексів.

Вартість НВІС для паралельного сортування масивів даних в основному залежить від площі кристала, яка визначається як витратами обладнання (кількість транзисторів), так і кількістю зовнішніх виводів, число яких обмежене рівнем технології та розміром кристалу. Орієнтація структур сортування даних на НВІС - реалізацію вимагає зменшення числа виводів інтерфейсу та кількості з'єднань між ПЕ [18].

1.3 Апаратні засоби для паралельного сортування даних

Одним із основних шляхів збільшення швидкодії сортування великих масивів даних є розпаралелювання процесу сортування та використання масово-паралельних обчислюваних засобів із великим обсягом пам'яті. До таких засобів відноситься графічні процесори (GPU – Graphics Processing Unit), які є процесорами класу SIMD (Single Instruction Multiple Data). Методи сортування, що ґрунтуються на базових елементарних операціях порівняння двох чисел та їх перестановки відрізняється один від одного способом вибору пар даних для виконання базових елементарних операцій [19]. Особливістю графічного процесора GPU є те, що він орієнтований на виконання однієї команди над 2p даними, які одночасно надходять на p обчислювальних ядер. Для ефективного сортування масивів даних на базі графічного процесора GPU необхідно вибрати методи сортування, структура яких орієнтована на паралельно-конвеєрну реалізацію.

Центральний процесор CPU та графічний процесора GPU наділені рядом відмінностей як у будові, так у принципі функціонування. CPU (Central Processing Unit) є головним компонентом комп'ютера, що зчитує з пам'яті і виконує команди, обробляючи при цьому дані і керуючи роботою всього комп'ютера (рис.1.2).З іншими пристроями комп'ютера процесор пов'язаний шинами. Є типи шин: шина даних, адресна шина і командна шина. Для функціонування процесора необхідні регістри даних, керуючі регістри, операційний (обробляє) блок, керуючий блок і система команд, яку процесор розпізнає і виконує [20].

Роботу процесора синхронізує зовнішній генератор тактів. Відповідно до цих сигналами відбувається зчитування та виконання команд. Частота процесора визначає швидкодію процесора. Графічний процесор досягає високого прискорення в той же час є більш енергоефективними та економічно ефективним, ніж CPU. GPU-прискорювач встановлюється основний підтримці нових операційних систем від Apple (з OpenCL) і Microsoft (за допомогою DirectCompute). Причиною широкого і основного прийняття є те, що GPU є обчислювальним гігантом, і його можливості зростають швидше, ніж у x86 CPU. У сучасному ПК, графічний процесор може приймати безліч мультимедійних завдань, таких як прискорення Adobe Flash відео, транскодування відео між різними форматами, розпізнавання образів, вірусу зіставлення зі зразком і інші.

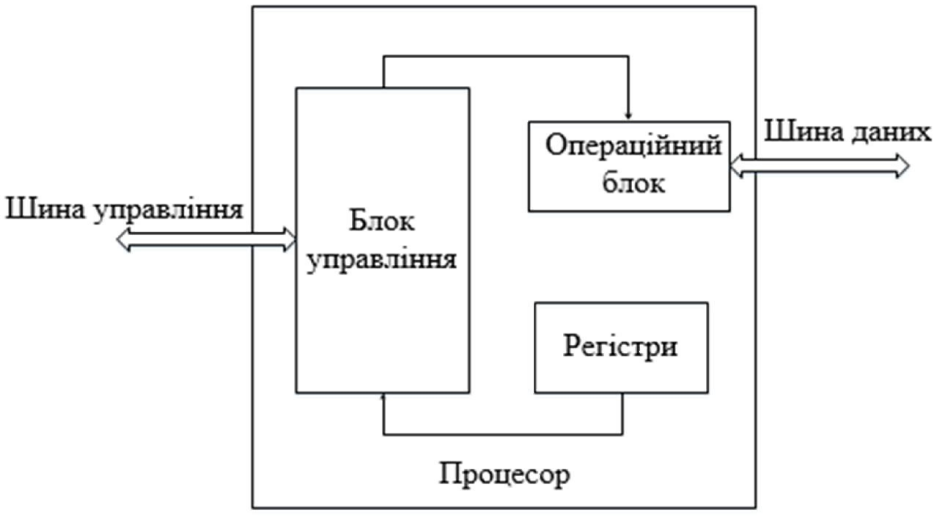

Рис.1.2 – Архітектура центрального процесора

Архітектура центрального процесора включає наступні компоненти : блок управління, операційний блок та регістри.

Блок управління (Control Unit) декодує команди і дає іншим частинам процесора відповідні вказівки для виконання команди і відповідає потім за передачу результатів в пам'ять. При цьому він використовує спеціальні регістри: лічильник команд (Program Counter) і регістр команд (Instruction Register).

Операційний блок (Processing Unit) містить арифметично-логічний пристрій (ALU - Arithmetic Logic Unit), який здатний виконувати обчислювальні дії з зазначеними даними або виконувати логічні операції. Він може комбінувати ці дії і виконувати такі складні операції як множення з плаваючою точкою в відповідному пристрої (FPU - Floating Point Unit), які неможливо виконати в арифметико-логічному пристрої. Операційний блок використовує спеціальні регістри: регістр стану (Status Register) і акумуляторний регістр (Accumulator Register) [21].

Регістри є внутрішньою пам'яттю процесора і поділяються наступним чином:

- регістри загального користування, які призначені для запам'ятовування даних і / або операндів при виконанні команд.
- спеціальні регістри, на які призначені для виконання спеціальних функцій при роботі процесора.
- акумулятор (A - Accumulator Register) запам'ятовує проміжні результати обчислень.
- лічильник команд (PC - Program Counter) містить адресу наступної команди. Він збільшується автоматично з кожним новим циклом. Підпрограми і переривання змінюють цей порядок, записуючи в лічильник команд нове значення.
- регістр команд (IR - Instruction Register) містить зчитану з пам'яті команду

- регістр стану (SR - Status Register) містить даний стан, що відбиває хід виконання команди.
- покажчик стека (SP - Stack Pointer) містить адресу наступної вільної комірки стековой пам'яті [22].

GPU обчислення, як правило, здійснюються в наукових і інженерних областях. Графічні процесори впроваджується у віртуальне середовище на робочому столі. GPU прискорює додатки, що працюють на процесорі шляхом виключення завантаженням деяких інтенсивних і трудомістких ділянок коду програми. Інша частина програми як і раніше працює на процесорі.

Обчислення на GPU, або GPGPU (англ. General-Purpose computing on Graphics Processing Units), заключаються у використані GPU (графічного процесора) для універсальних обчислень в області науки та проектування.

GPU обчислення представлені сумісним використанням CPU і GPU в гетерогенній моделі обчислень. Стандартна частина програми виконується на CPU, а більш вимоглива до обчислень частина обробляється з GPU прискоренням. З точки зору користувача програма працює швидше, оскільки вона використовує високу продуктивність GPU для підвищення загальної швидкодії.

GPGPU є останньою тенденцією в галузі комп'ютерних наукових досліджень. Графічні процесори є спів-процесорами, які добре оптимізовані для обробки комп'ютерної графіки. Обробка комп'ютерної графіки - це поле діяльності, де переважають операції паралельної обробки данних - зокрема, матричні операції лінійної алгебри.

Технологічний консорціум Khronos Group випустив специфікацію OpenCL, яка є основою для написання програм, які виконуються на різних платформах, що складаються з CPU і GPU. Apple, Intel, Nvidia та інші підтримують OpenCL [23].

Сучасні графічні процесори (рис.1.3) містять набір однакових обчислювальних пристроїв (потокових процесорів,), що працюють з загальною пам'яттю графічного процесора (відео ОЗП).

Використання GPU-обчислень дозволяють обробляти великі дані в багатьох типах додатків, в тому числі:

- аналітики;
- системи автоматизованого проектування ;
- прогнозування погоди;
- науки про життя;

засоби масової інформації, індустрії розваг та анімації

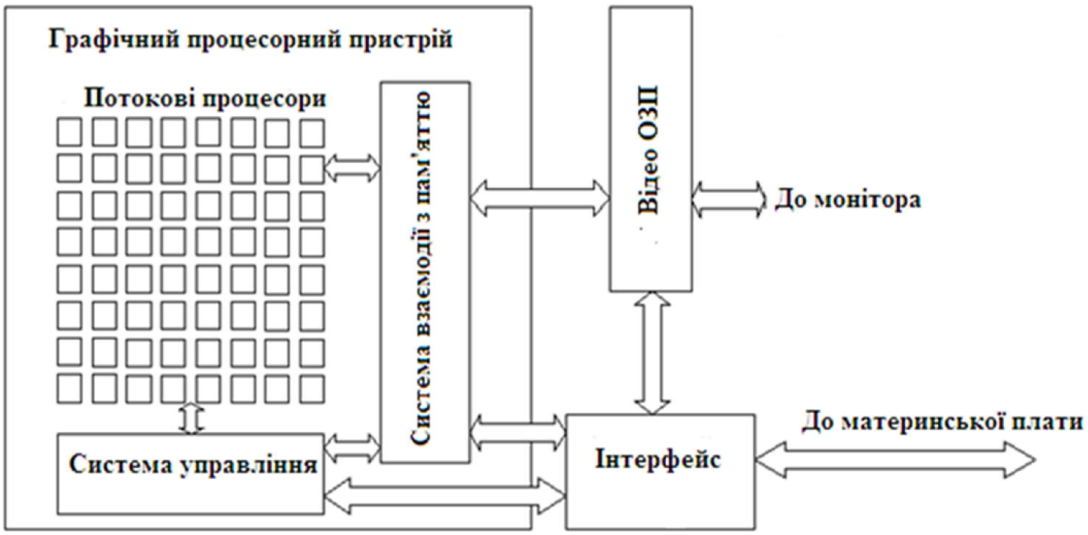

Рис.1.3- Архітектура графічного процесора

Число потокових процесорів, а також розмір відео ОЗП може бути різним, залежно від моделі GPU. Всі Потокові процесори синхронно виконують одну і ту ж команду. Система команд потокових процесорів включає арифметичні команди обчислень з 32-розрядної точністю, команди управління (розгалуження і цикли), а також команди звернення до пам'яті. якщо поточний потік блокується з доступу до пам'яті, на виконання ставитися другий. Оскільки контекст потоку повністю зберігається на регістрах графічного процесора, перемикання здійснюється за один такт. За перемикання потоків відповідає диспетчер потоків, який не є програмованим. Завдяки великій кількості потокових процесорів продуктивність GPU дуже значна, а якщо ще врахувати, що існує можливість установки на одну машину двох графічних карт, то це дозволяє отримати пікову продуктивність до декількох ТФлопс. Крім того, на

деяких практичних завданнях може досягатися значний відсоток пікової продуктивності (до 70%). На Рис. 1.4 представлена будова CPU та GPU процесорів.

Можлива установка на один ПСК відразу декількох обчислювачів, що забезпечить загальну продуктивність до 4х терафлопс. 1 терафлопс = 1 трильйон операцій в секунду = 1000 мільярдів операцій в секунду [24].

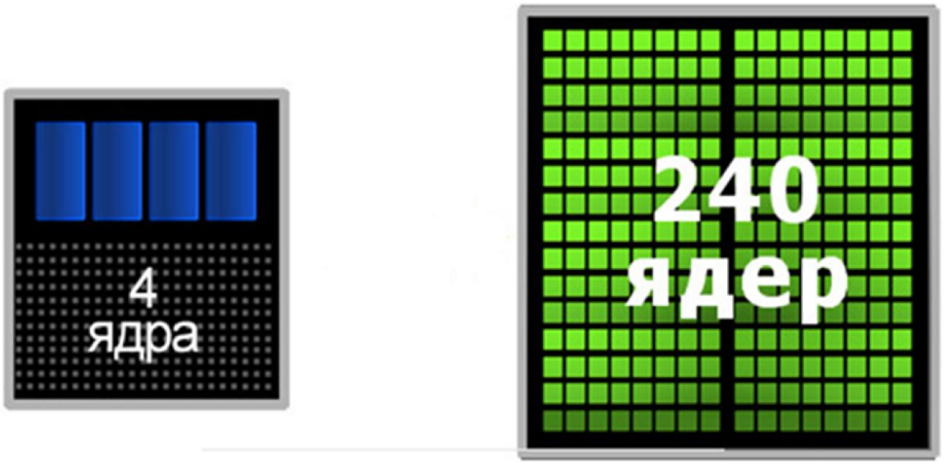

Рис.1.4- Будова CPU та GPU процесорів

CPU складається з декількох ядер, оптимізованих для послідовної обробки даних, в той час як GPU складається з тисяч дрібніших і енергоефективних ядер, створених для обробки декількох завдань одночасно. Модель програмування CUDA використовується як CPU і GPU.

CUDA (англ. Compute Unified Device Architecture) – це програмно-апаратна архітектура паралельних обчислень від NVIDIA, що дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню GPU (графічних процесорів).

CUDA має кілька переваг у порівнянні з традиційними обчисленнями загального призначення на графічних процесорах за допомогою графічних інтерфейсів API:

- код може зчитувати інформацію з довільних адрес в пам'яті;
- єдина віртуальна пам'ять (CUDA 4.0 і вище);

- спільна пам'ять : CUDA має великий об’єм пам'яті, які можуть бути розділена серед потоків;
- швидше завантаження і readbacks ;
- повна підтримка цілочисельних і побітових операцій.

Типова послідовність операцій для програми CUDA C є:

- оголошення та виділення хосту і пам'ять пристрою;
- ініціалізація даних хоста;
- передача даних від хоста до пристрою;
- передача результатів від пристрою до хосту.

1.4 Постановка завдання на дипломну роботу

Оскільки задача сортування чисел є однією з типових проблем обробки даних,то важливим є пошук розробки ефективних пристроїв сортування методом злиття на основі просторово – часових графів та знаходження оптимального співвідношення між затратами обладнання та швидкодією. Основним шляхом підвищення швидкодії сортування великих масивів даних є розпаралелювання процесу сортування та використання масово-паралельних обчислюваних засобів із великим обсягом пам’яті [25].

Метою роботи є формування вимог і вибір принципів побудови апаратних засобів реального часу, розроблення узгоджених потокових графів алгоритму сортування масивів даних методом злиття та синтез на їх основі апаратних засобів з високою ефективністю використання обладнання

Реалізація високоефективних спеціалізованих засобів сортування потребує широкого використання сучасної елементної бази, розроблення нових методів, алгоритмів і НВІС-структур. Режим реального часу та НВІС-реалізація алгоритмів сортування з високою ефективністю використання обладнання забезпечується розпаралелюванням і конвеєризацією процесів сортування, апаратним відображенням структури алгоритмів у архітектуру, яка адаптована до інтенсивності надходження потоків даних

Отже, розроблення паралельних алгоритмів і програмних засобів для сортування великих масивів даних на графічному процесорі GPU є актуальною задачею.

2 АЛГОРИТМИ СОРТУВАННЯ МАСИВІВ ДАНИХ МЕТОДОМ ЗЛИТТЯ

2.1 Формування вимог до апаратних засобів сортування даних

Для повноцінної розробки додатків CUDA необхідно мати відеокарту з підтримкою даної технології. Якщо відеокарта підтримує CUDA, то завантажується і встановлюється остання версія драйвера.Далі необхідно встановити CUDA Toolkit, GPU Computing SDK і Паралельний Nsight для відповідної версії операційної системи.

NVIDIA CUDA Toolkit - набір інструментальних засобів для розробки GPU-додатків на C / C ++ [23]. Ці інструменти включають: CUDA-компілятор, бібліотеку математичних функцій, а також набір утиліт для налагодження і профілювання додатків. Крім цього в установку входить докладний опис програмно-апаратної моделі, керівництво користувача і документація по CUDA API.NVIDIA GPU Computing SDK містить безліч прикладів використання CUDA, які супроводжуються докладним описом.

NVIDIA® Parallel Nsight - потужне розширення для Visual Studio, що дозволяє здійснювати налагодження, профілювання і аналіз CUDA-додатків і не тільки [24]. При використанні GPU розробнику доступно декілька видів пам'яті: регістрова, локальна, глобальна, що розділяється, константна і текстурна пам'ять.

Однією з найширше розповсюджених вимог, що ставиться до апаратних засобів паралельного сортування масивів даних є забезпечення високої швидкодії роботи даного пристрою та режиму роботи в реальному часі. Подібна проблема виникає, як правило, при використанні таких засобів для сортування інтенсивних потоків масивів даних в реальному часі [26]. Для забезпечення реального часу необхідно, щоб час сортування T_c не перевищував час надходження даних $T_{нд}$, тобто:

$$T_c \leq T_{нд} \tag{2.1}$$

Крім того, апаратні засоби повинні мати високу ефективність використання обладнання, яка враховує кількість виводів інтерфейсу, зв'язує продуктивність з витратами обладнання та дає оцінку елементам за

продуктивністю. Кількісна величина ефективності використання обладнання при НВІС - реалізації алгоритмів сортування даних визначається :

$$E = \frac{R}{t_c(\sum_{i=1}^{H} W_{ПЕ_i} + k_1 Y + k_2 P)} \quad (2.2)$$

де, R – складність алгоритмів сортування даних, яка визначається кількістю операцій попарного порівняння;

$W_{ПЕ}$ – витрати обладнання на реалізацію ПЕ;

H – кількість ПЕ;

t_c – час сортування масиву даних;

Y – кількість виводів інтерфейсу;

k_1– коефіцієнт врахування кількості виводів інтерфейсу, де k_1= f (Y);

P – кількість зв'язків між ПЕ;

k_2 – коефіцієнт врахування зв'язків між ПЕ, де k_1= f(P).

Однією з умов досягнення високої ефективності використання обладнання при сортуванні масивів даних у реальному часі є виконання умови:

$$P_d \leq D_c \quad (2.3)$$

де, $P_d = kn_k F_d$ - інтенсивність надходження вхідних даних;

$D_c = \frac{sn_s}{T_k}$ - інтенсивністю сортування;

k – кількість каналів надходження вхідних даних;

n_k– розрядність каналів надходження даних;

F_d – частота надходження даних;

s – кількість каналів сортування даних;

n_s– розрядність каналів сортування даних;

T_k– конвеєрний такт роботи пристрою сортування.

Для НВІС - реалізацій алгоритми сортування повинні бути добре структурованими, орієнтованими на реалізацію на множині взаємозв'язаних ПЕ та забезпечувати детерміноване переміщення даних. Структура та операції, які виконують ПЕ залежить від вимог, що висуваються до часу сортування. При

розробці алгоритмів сортування для НВІС - реалізацій потрібно одночасно враховувати багато взаємопов'язаних факторів. Передусім необхідно, щоб алгоритми сортування були рекурсивними та локально залежними. В рекурсивному алгоритмі всі ПЕ повинні виконувати однакові операції [27].

Алгоритми сортування даних можна розділити на два класи з локальними та глобальними зв'язками між ПЕ. Для оцінювання зв'язків між ПЕ, часової та апаратної складності реалізації алгоритму розроблять решіткову модель, яка складається з множини ПЕ, що в декартові системі координат утворюють точкові системи. В решіткові моделі кожному ПЕ ставиться у відповідність j часовий та просторовий і індекси, які вказують коли та де виконується кожна із операцій алгоритму. В алгоритмах з локальними пересилками даних різниця між просторовими індексами j на кроці рекурсії обмежена деякою константою, так як в таких алгоритмах обміни здійснюються тільки між найближчими сусідніми ПЕ. Алгоритми, що при рекурсії мають рознесені просторові індекси відносяться до класу алгоритмів з глобальними зв'язками. Складність реалізації обміну між ПЕ залежить як від розрядності каналів передачі даних, так і від віддалі між ними, яка визначається як різниця двох просторових індексів.

Для забезпечення високої швидкодії та ефективності використання обладнання функціональні оператори, які реалізуються у сходинках конвеєра, мають бути простими та мати приблизно однаковий час реалізації [28]. Однотактні алгоритмічні пристрої можна розглядати як одноступінчатий конвеєр. У зв'язку з цим актуальним є розгляд питань пов'язаних із синтезом конвеєрних алгоритмічних структур реального часу з високою ефективністю використання обладнання .

Вихідною інформацією для синтезу пристроїв сортування масивів даних у реального часу є:

- кількість вхідних даних N;
- розрядність вхідних даних;
- вимоги до інтерфейсу;
- інтенсивність надходження вхідних даних , $P_d = kn_k F_d$;
- техніко-економічні вимоги і обмеження.

Основними шляхами мінімізації апаратних затрат і узгодження інтенсивності надходження даних із інтенсивністю сортування є:

- вибір ефективних методів і алгоритмів сортування даних;
- вибір складності функціональних операторів;
- зміна розрядності каналів надходження даних і розрядності ПЕ;
- зміна кількості каналів надходження даних.

Для сортування даних методом злиття можна використовувати багатоядерний графічний процесор Nvidia GT 610 з підтримкою архітектури і програмної моделі CUDA.

Графічний процесор Nvidia GT 610 має такі характеристики:

- кількість ядер - 48;
- тактова частота графічного процесора - 1620;
- тактова частота процесора – 810 МГц;
- швидкодія пам’яті - 1.8 Гбіт/с;
- об’єм пам’яті - 1024 Мб;
- інтерфейс пам’яті - 64-біт DDR3;
- максимальна смуга пропускання пам'яті - 14.4;
- програмне середовище - CUDA
- шина - PCI-E 2.0.

Для НВІС - реалізацій алгоритми сортування повинні бути добре структурованими, орієнтованими на реалізацію на множині взаємозв’язаних ПЕ та забезпечувати детерміноване переміщення даних. Структура та операції, які виконують ПЕ залежить від вимог, що висуваються до часу сортування. При розробці алгоритмів сортування для НВІС - реалізацій потрібно одночасно враховувати багато взаємопов’язаних факторів. Передусім необхідно, щоб алгоритми сортування були рекурсивними та локально залежними. В рекурсивному алгоритмі всі ПЕ повинні виконувати однакові операції [29].

Для забезпечення високої ефективності використання обладнання при розроблені НВІС - структур для сортування масивів даних у реальному часі використовуються такі принципи :

- розпаралелювання процесу сортування даних;

- спеціалізації та адаптації апаратних засобів до структури алгоритмів сортування та інтенсивності надходження даних;
- однорідності ПЕ та регулярності зв’язків між ними;
- узгодженості інтенсивності сортування з інтенсивністю надходження даних.

Для синтезу апаратних засобів сортування масивів даних у реального часу є відомий метод адекватного апаратного відображення структури графів алгоритмів. При використанні даного методу кожному функціональному оператору ставиться у відповідність ПЕ або блок сортування, макрооператору затримки та перестановки – пам'ять і комутатори, макрооператор управління - блок управління, а дугам між функціональними операторами - каналами передачі даних. Синтезовані таким чином апаратні засоби є алгоритмічними. У таких структурах алгоритм реалізується при проходженні та обробці даних від входів до виходів через всі операційні блоки. За режимами роботи алгоритмічні структури діляться на синхронні та асинхронні [30].

Конвеєризація алгоритмічних структур передбачає розділення їх на сходинки шляхом введення буферної пам’яті. При цьому, кожна сходинка конвеєра складається з двох компонентів: буферної пам’яті; ПЕ (блоку сортування) блоку управління, які реалізують оператори ярусу [32]. Для забезпечення високої швидкодії та ефективності використання обладнання функціональні оператори, які реалізуються у сходинках конвеєра, мають бути простими та мати приблизно однаковий час реалізації. Однотактні алгоритмічні пристрої можна розглядати як одноступінчатий конвеєр [33].У зв’язку з цим актуальним є розгляд питань пов’язаних із синтезом конвеєрних алгоритмічних структур реального часу з високою ефективністю використання обладнання.

При синтезі матричного пристрою паралельного сортування кожному функціональному оператору ставиться у відповідність ПЕ, які з’єднані між собою каналами передачі даних у відповідності з потоковим графом алгоритму паралельного сортування (рис.2.1).

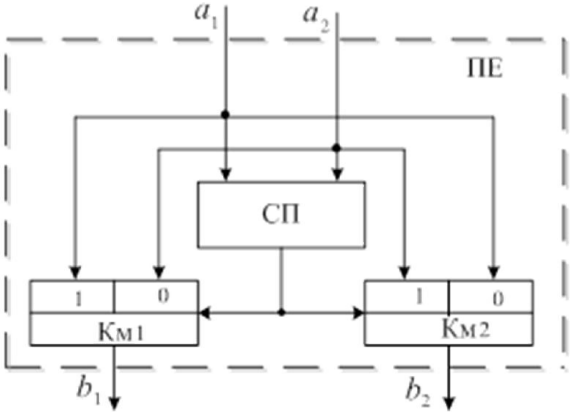

Рис.2.1 - Структура ПЕ з паралельним надходженням всіх розрядів даних

де , a_1, a_2 – вхідні дані;

Км – комутатор;

СП – схема порівняння;

b_1, b_2- вихідні дані.

У матричному пристрою паралельного сортування методом злиття одним із варіантів узгодження інтенсивності надходження даних із інтенсивністю сортування є зміна розрядності каналів надходження даних і розрядності ПЕ[34]. Дані типи ПЕ є крайніми варіантами використання вертикально-групового надходження і порівнянням даних. У ПЕ (рис.2.1) числа надходять паралельним кодом і їх порівняння здійснюється за один такт.

2.1.1 Переваги та недоліки мови C++

В процесі розробки мови C + +, одним із важливих критеріїв вибору, була простота даної мови. У синтаксисі мови C++ не існує типів даних та операцій,що базуються на високому рівні. У C++ немає випадків, що могли б привести до накладних втрат пам'яті , в тому числі, якщо вони безпосередньо не використовуються у програмі. Якщо користувач описав структуру,яка містить дві величини, що займають по 32 розряди, то гарантується, що вона поміститься в 64-х розрядний регістр.

Однією із основних переваг мови програмування С++ є її швидкодія. Швидкість програмування та роботи фактично нічим не відстає від програм на С,хоча ,розробники мають у своєму арсеналі нові технології та засоби.

Другою перевагою мови С++ є масштабованість ,а саме : можливість розробляти нові програми для широкого кола платформ та систем.

Важливою особливістю є можливість створення узагальнених алгоритмів для будь-яких даних, їхня спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів.

С ++ надає можливість підтримки різних технік програмування, враховуючи стандартне директивне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, метапрограмування.

Незважаючи на ряд переваг,мова С++ має й недоліки. Наприклад,недостатня підтримка модульності. Під'єднання зовнішнього модуля через заголовний файл (#include) разюче сповільнює його компіляцію. Для виправлення даної проблеми,велика кількість компіляторів впроваджують перекомпіляцію заголовних файлів. Також,істотним недоліком мови програмування С++ є примітивність процесора пам'яті. Результатом цього,- неспроможність вирішувати більшості завдань метапрограмування .З іншого боку,в силу своєї примітивності,є причиною видачі помилок та затрати достатньо великого проміжку часу збоку розробника,для вирішення проблеми.

Мова С++ представлена як мультипарадигменна мова. Проте, незважаючи на це,в ній відсутня підтримка функціонального програмування. В якійсь мірі, ця недовершеність вирішується шляхом підключення бібліотек,що застосовують засоби мета програмування для розширення мови функціональними конструкціями. Якість такого рішення суттєво поступається якості вбудованих у функціональні мови рішень.

Також,однією з проблем С++ є синтаксис мови. Операція порівняння позначається « == »,в свою чергу, операція присвоєння має символ « = ». Їх просто переплутати,хоча й дана конструкція матиме правильний синтаксис,проте завдасть важковловимої помилки.

Одним з недоліком мови С++ вважають відсутність системи збірки сміття. Проте, С++ має достатній запас засобів (класи з конструкторами і деструкторами, стандартні шаблони, передача параметрів за посиланням), що дають змогу практично скоротити використання небезпечних вказівників. Однак, відсутність вбудованої збірки сміття дає можливість користувачеві вибрати стратегію управління ресурсами на власний розсуд. Також, збірка сміття суттєво збільшує час роботи програми, і це перешкода там, де продуктивність є дуже важливою. У С++ за допомогою трьокрапки (...) можуть бути описані функції з невизначеним набором параметрів. Контроль за типами параметрів таких функцій не ведеться, що підвищує гнучкість їх використовування. У С++ можна задавати параметрам функцій значення за замовчуванням. У такому випадку при виклику функції можуть бути вказані значення тільки деяких параметрів, тоді як іншим вони будуть призначені автоматично. Транслятор мови С++ перевірить відповідність фактичних типів значень, переданих у функцію, формальним типам аргументів функції. Також буде перевірена відповідність типу значення типу змінної, якій привласнюється це значення, що повертається.

С++ був розвинутий з мови програмування С і за дуже малими виключеннями зберігає С як підмножину. Базова мова, С підмножина С++, спроектована так, що існує дуже близька відповідність між його типами, операціями й операторами і комп'ютерними об'єктами, з якими безпосередньо приходиться мати справу: числами, символами й адресами.

Microsoft Visual C++ містить безліч інтегрованих засобів візуального програмування. Компілятор Visual C++ містить багато нових інструментальних засобів і поліпшених можливостей, надає величезні можливості в плані оптимізації додатків, внаслідок чого можна отримати виграш як відносно розміру програми, так і відносно швидкості її виконання, незалежно від того, що являє собою ваш додаток.

Система Microsoft Visual C++ дозволяє створювати як маленькі програми і утиліти для персонального використання, так і корпоративні системи, що працюють з базами даних на різних платформах.

2.1.2 Програмне середовище для сортування даних

Visual Studio 2015 є повнофункціональною і розширювальною програмою для індивідуальних розробників, проектів з відкритим вихідним кодом, наукових досліджень, освіти і невеликих професійних команд. Visual Studio 2015 призначений для створення додатків для Windows, Android а також веб-додатків і хмарних сервісів та додатків для будь-якої платформи. Visual Studio Test Professional 2015 ідеально підходить для тестувальників, бізнес-аналітиків, менеджерів по продуктах, а також інших зацікавлених осіб, які потребують інструментів для спільної роботи. Visual Studio Test Professional 2015 підтримує безперервний зворотний зв'язок з клієнтами, а також забезпечує простежуваність через ці процеси.

У Visual Studio, щоб включити інструмент або скрипт, які часто використовуєте при кодуванні можна створювати власні пункти меню і вікна інструментів, щоб об'єднати власні інструменти в середовищі розробки Visual Studio. Можлива функція роширення редактора Visual Studio для аналізу і виправлення коду, або додати новий тип проекту, щоб включати в себе тільки те, що вам потрібно.

Visual Studio легко інтегрується з візуальними Studio Team Services або Team Foundation Server для управління версіями з Git або TFVC (Control Team Foundation Version). За допомогою Visual Studio Team Services можливе зберігання коду в хмарі, без підтримки локального сервера. Кожен раз, коли створюється проект коду, є можливість додати його в систему управління версіями.

Visual C ++ надає інструменти для редагування, створення, розгортання і налагодження крос-платформенного коду. Крім шаблонів для додатків Windows, можна створювати проекти з шаблонів для Android Native активність додатків, ОС IOS додатків або спільних проектів бібліотеки коду для декількох платформ, які включають Xamarin гібридні програми. Visual Studio дає можливість встановити точки зупину, спостереження за змінними, переглядати стек через код C ++ в Visual Studio відладчик.

2.2 Вибір алгоритму для паралельного сортування даних

Під час паралельної обробки даних велика задача поділяється на кілька менших [35]. Ефективність реалізації паралельної обробки даних залежить від способу поділу задачі на підзадачі , які можна виконувати одночасно (паралельно).

Найбільш орієнтованими на апаратну реалізацію є методи Quick Sort, метод злиттям .

Час роботи алгоритму швидкого сортування залежить від степеня збалансування, яким характеризується розбиття. Збалансування, в свою чергу, залежить від того, який елемент був обраний в якості опорного [40]. Якщо розбиття збалансоване, асимптотично алгоритм працює так само швидко, як сортування злиттям. В протилежному випадку асимптотична поведінка цього алгоритму така сама повільна, як й в алгоритму сортування включенням.

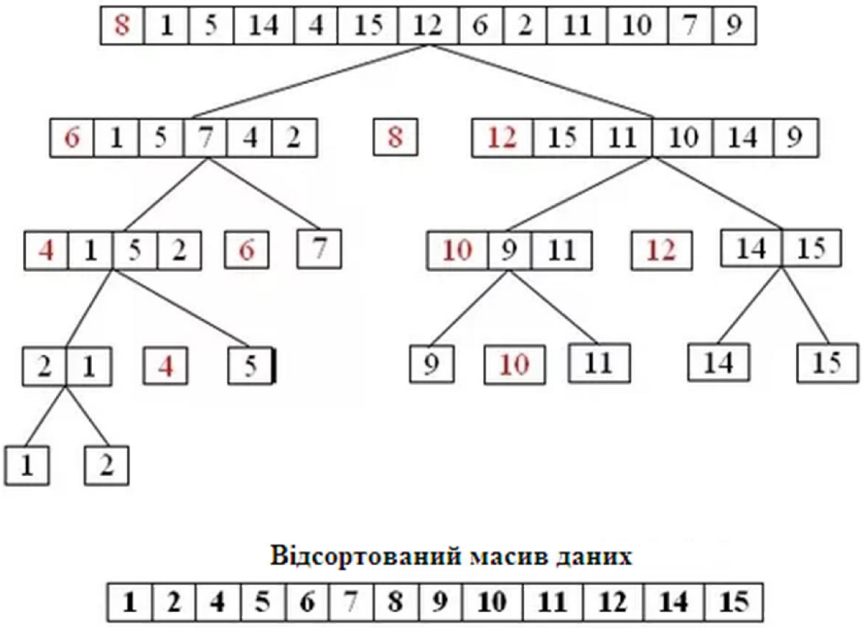

Рис.2.2 – Алгоритм паралельного сортування Quick Sort

Сортування методом злиття як і quick sort керується принципом «розділяй та владарюй», тобто він ділить множину елементів, які потрібно відсортувати, на підмножини які сортуються цим же алгоритмом [41].

Сортування методом злиття: роботу алгоритму можна розбити на два кроки:

- поділити масив на дві рівні частини і застосувати сортування злиттям до кожної з них.
- злити два відсортовані масиви з попереднього кроку у один таким чином, щоб він був відсортований.

Самого ж сортування “злиттям” розрізняють декілька методів :

- злиття зверху донизу полягає у тому, що спочатку впорядковуються дві частини масиву (L..(L+R)/2 та (L+R)/2+1..R), а потім ці частини зливаються [42].

```
Procedure MergeSort (var A:IntArray;L,R:integer);
Begin
  If L>=R then Exit;
  MergeSort (A,L,(L+R) div 2);
  MergeSort (A,(L+R) div 2+1,R);
  Merge (A,L,(L+R) div 2+1,R);
End;
```

- злиття знизу догори не використовує рекурсію. Спочатку весь масив розбивається (умовно) на частини довжиною в один елемент, які після цього попарно зливаються. Коли маленькі частинки злито, процес повторюється для отриманих вдвічі більших частин [43].

```
Procedure MergeSort (var A:IntArray;N:integer);
Var m,i:integer;
Begin
  m:=1;
  While m<N do
       Begin
         i:=0;
```

```
    While i+m<=N do
     Begin
      Merge (A,i+1,i+m,Min (i+2*m,N));
      i:=i+2*m;
     End;
    m:=2*m;
   End;
 Merge (A,1,N div 2,N);
End;
```

Сортування методом злиття характеризується тим,що масив ділиться елементарним чином, а саме діленням націло пополам (рис.2.3) [45].

В основі алгоритмів сортування методом злиття лежить базова операція об’єднання двох упорядкованих масивів у один упорядкований масив . На початку сортування вхідний масив чисел розбивається на N/2 упорядкованих масивів довжиною одиниця. У результаті виконання першої базової операції формуються N/4 впорядкованих масивів довжиною два.

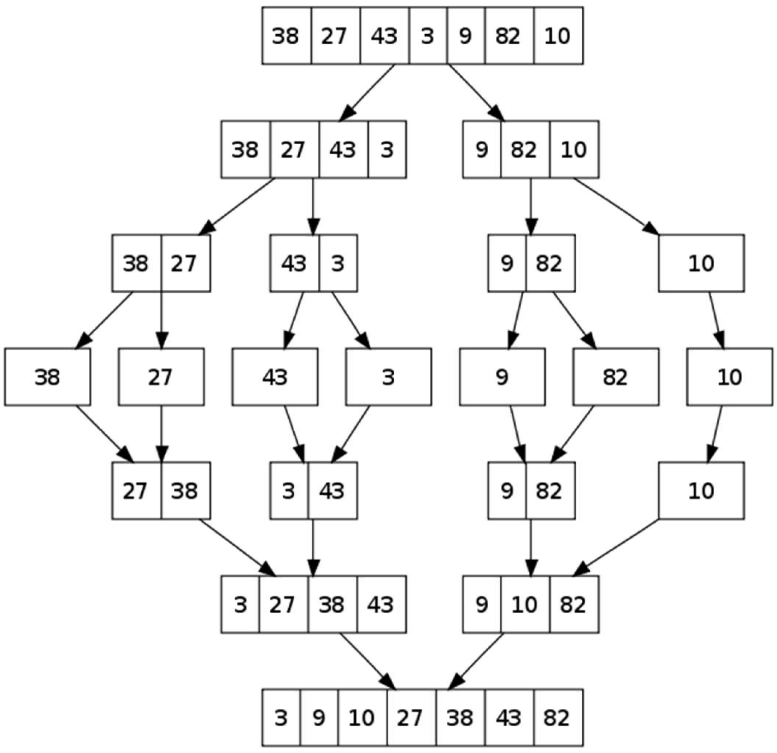

Рис.2.3 - Паралельне сортування даних методом злиття

Кількість типів базових операцій, необхідних для сортування масиву із N чисел визначається за формулою:

$$k = \log_2 N \qquad (2.4)$$

У таблиці 2.1 наведено основні переваги та недоліки алгоритмів паралельного сортування даних,а саме: Quick Sort та сортування “злиттям”.

Таблиця 2.1 – Порівняльна характеристика алгоритмів паралельного сортування даних

Назва алгоритму	Переваги	Недоліки	Обчисл. складність
Quick Sort	1.Швидкодіючий алгоритм внутрішнього сортування загального призначення. 2.Вимагає лише $O(\log n)$ додаткової пам'яті . 3.Добре поєднується з механізмами кешування і віртуальної пам'яті. 4.Працює на пов'язаних списках та інших структурах з послідовним доступом.	1.Реалізація алгоритму може привести до помилки переповнення стека 2.Нестійкий	$O(n \log n)$
Сортування “злиттям”	1.Поєднується з механізмом кешування пам'яті. 2.Працює в паралельному варіанті: легко розбити завдання між процесорами . 3.Якщо один процесор затримається, решта беруть його роботу на себе.	1.На «майже відсортованих» масивах працює з такою швидкістю, як на хаотичних.	$O(n \log n)$

Отже, основною перевагою сортування методом злиття є те , що він працює в паралельному варіанті: легко розбити завдання між процесорами порівну, якщо один процесор затримається, решта візьмуть його роботу на себе [50].

2.3 Форми відображення алгоритмів паралельного сортування даних

Для оцінки обчислювальних і структурних характеристик алгоритмів сортування даних використовується їх подання у вигляді функціонального графу:

$$F = (Ф, Г) \tag{2.5}$$

де, $Ф = \{Ф_1, Ф_2, Ф_3, \dots, Ф_n\}$ – множина функціональних операторів,

Г- закон відображення зв'язків між операторами .

Функціональний граф алгоритму відображає послідовність і взаємну залежність функціональних операторів [51]. Графічно функціональний граф алгоритму відображається у вигляді вершин, що відповідають операторам алгоритму $Ф_i$ та дуг, які відображають зв'язки між операторами. Складність функціональних операторів $Ф_i$ визначається як структурними одиницями інформації, так і складністю виконуваних операцій. Таке подання алгоритму не в повній мірі відображає просторово-часові залежності між функціональними операторами.

Виявити паралелізм алгоритму сортування даних, управляти ним для знаходження оптимальних просторово-часових рішень забезпечує подання функціонального графу в ярусно-паралельній формі (ЯПФ) [52]. При такій формі подання алгоритму здійснюється розподіл всіх його функціональних операторів $Ф_i$ за ярусами таким чином, що в j-у ярусі розміщені функціональні оператори, які залежать хоча б від одного функціонального оператора (j-1) ярусу і не залежать від операторів наступних ярусів. В середині ярусу функціональні оператори між собою не мають з'єднань [53].

$$\vec{V}_0 = \vec{V}_{Ф1} + \vec{V}_{Ф2} + \cdots + \vec{V}_{Фn} \quad (2.6)$$

В векторі $\vec{V}_0$ визначаємо номера елементів, які дорівнюють нулю, наприклад, другий та п’ятий [55]. За визначеними номерами знаходимо функціональні оператори, що не мають нащадків і відповідно утворюють нульовий ярус, в даному випадку $Ф_2$ та $Ф_5$. Далі за формулою обчислюємо:

$$\vec{V}_1 = \vec{V}_0 - \vec{V}_{Ф2} - \vec{V}_{Ф5} \quad (2.7)$$

У векторі $\vec{V}_1$ знаходимо номери нульових елементів, за якими визначаємо функціональні оператори, що утворюють перший ярус. Аналогічно обчислюємо наступні вектори та визначаємо функціональні оператори, що утворюють наступні яруси. Для переходу від алгоритмів сортування до НВІС - архітектур з високою ефективністю використання обладнання пропонується такі алгоритми подавати в вигляді узгодженого потокового графу [56].

2.4 Узгоджено - потоковий граф для сортування даних

Для ефективної реалізації алгоритму сортування великих масивів даних методом злиття на графічному процесорі GPU необхідно даний алгоритм подати у вигляді конкретизованого потокового графу [57]. Особливістю конкретизованого потокового графу сортування великих масивів даних методом злиття є його орієнтація на архітектуру графічного процесора GPU.

Розробка конкретизованого потокового графу сортування великих масивів даних методом злиття, можна розбити на такі чотири етапи:

- декомпозиція алгоритму розв’язання задачі;
- проектування комунікацій (обмінів даними) між функціональними операторами;
- укрупнення функціональних операторів;

- планування процесу сортування.

На етапі декомпозиції алгоритм сортування масивів даних розбивається на функціональні оператори Φ_{ji}, між якими установлюються зв'язки, що відповідають даному алгоритму сортування. Чим більша степінь деталізації алгоритму отримуємо у результаті декомпозиції, тим простіше і легше можна здійснити адаптацію його до вимог конкретного застосування. Декомпозицію можна здійснювати за методом функціональної декомпозиції [58].

Використання методу функціональної декомпозиції дозволяє отримати просторово-часове відображення структури алгоритму сортування масивів даних на основі елементарних операцій попарного порівняння та перестановки даних [59]. Час і спосіб виконання таких операцій ПЕ є одними із основних параметрів при визначені конвеєрного такту роботи і розрядності каналів надходження даних у НВІС - структурах сортування даних.

Результатом першого етапу розробки є граф схема алгоритму, де функціональні оператори мають приблизно однаковий час виконання, а їх складність визначається необхідною швидкодією.

На етапі проектування комунікацій необхідно визначити структуру та розрядність каналів обміну даними між функціональними операторами . Для чого виконується перехід від граф схеми алгоритму до потокового графу, в якому здійснюється просторово-часове розміщення і закріплення функціональних операторів за ярусами. Структура зв'язків у потоковому графі між функціональними операторами сусідніх ярусів визначається кількістю каналів надходження даних і їх розрядністю[60].

За результатами перших двох етапів розробки можна оцінити інтенсивністю сортування масивів даних D_c, яку можна отримати при апаратній реалізації потокового графу сортування. Вихідними даними для визначення інтенсивністю сортування $D_c = \frac{sn_s}{T_k}$, масивів даних є:

- кількість каналів сортування даних s і їх розрядність n_s;
- швидкодія елементної бази, яка враховується при визначені конвеєрного такту сортування T_k.

Для оцінки узгодженості інтенсивності надходження даних P_d із обчислювальної здатності D_c вводиться коефіцієнт узгодженості, який визначається так:

$$L = \frac{P_d}{D_c} \quad (2.8)$$

Коефіцієнт узгодженості L може бути L = 1, L > 1 та L < 1.

Коли L = 1, то розроблений граф сортування даних є узгодженим і його є апаратна реалізація забезпечує високу ефективністю використання обладнання.

У випадку коли L>1, то розроблений граф сортування не є узгодженим і для його узгодження необхідно збільшувати інтенсивність сортування D_c. Підвищення інтенсивності сортування D_c може бути досягнуте шляхом збільшенням кількість каналів сортування даних s і їх розрядності n_s або зменшенням складності функціональних операторів $Ф_{ji}$. У випадку коли зміною перерахованих параметрів не вдається досягнути необхідної інтенсивності сортування D_c , то тоді підвищення інтенсивності сортування D_c досягається за рахунок паралельної реалізації L графів [61].

Першим із варіантів отримання конкретизованого графу сортування даних є його лінійна проекція на горизонтальну вісь X [63].У цьому випадку укрупнення операцій здійснюється об’єднанням між ярусами функціональних операторів і каналів передачі даних. Приклад проекції потокового графу алгоритму паралельного сортування методом злиття масиву із 16-и чисел на горизонтальну вісь X, наведений на рис.2.4, де

$Ф_{ji}$ – функціональний оператор попарного порівняння та перестановки даних;

$Ф_{мзп}$ – макрооператор затримки та перестановки;

$Ф_{му}$ – макрооператор управління.

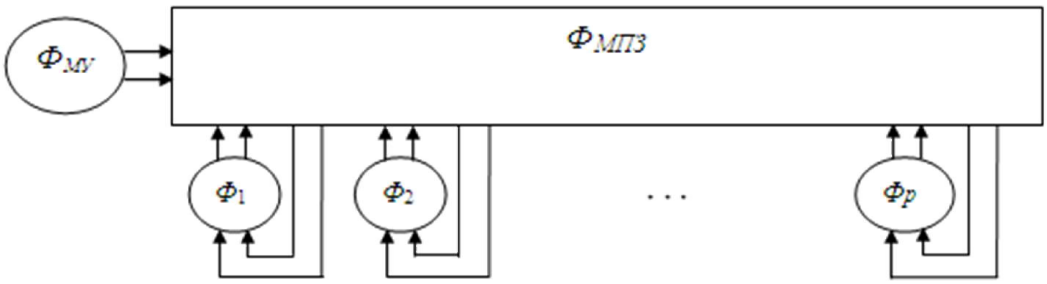

Рисунок 2.4 – Конкретизований граф сортування даних на горизонтальну вісь X

Реалізація конкретизованого потокового графу сортування масиву даних методом злиття відбувається по тактам, кількість яких визначається за формулою:,

$$T = (N - 1)\frac{N}{2p} \tag{2.9}$$

де, N – кількість даних в масиві;

p – кількість обчислювальних ядер у процесорі.

Другим варіантом отримання конкретизованого потокового графу сортування даних методом злиття є його лінійна проекція на вертикальну вісь Y. У цьому випадку укрупнення операцій здійснюється об'єднанням функціональних операторів і каналів передачі даних як у межах ярусу, так і між ярусами[64].У результаті такого укрупнення операцій отримуємо конкретизований потоковий граф з одним каналом передачі даних і чотирьох типів базових операцій злиття двох упорядкованих масивів розміром 2^{i-1} у один упорядкований масив розміром 2^i, де i=1,…,N.

Оціночна інтенсивністю сортування масивів даних для лінійної проекції на вертикальну вісь Y дорівнює:

$$D_c = \frac{n_s}{T_k} \tag{2.10}$$

Приклад проекції потокового графу алгоритму паралельного сортування методом злиття масиву із 16-и чисел на вертикальну вісь Y основі одноканального двошляхового злиття, наведений на рис. 2.5, де $Ф_{зп1}$ - $Ф_{зп4}$ –

оператори затримки та перестановки для одноканального двошляхового злиття; $\Phi_{y1} - \Phi_{y4}$ – оператори управління [65].

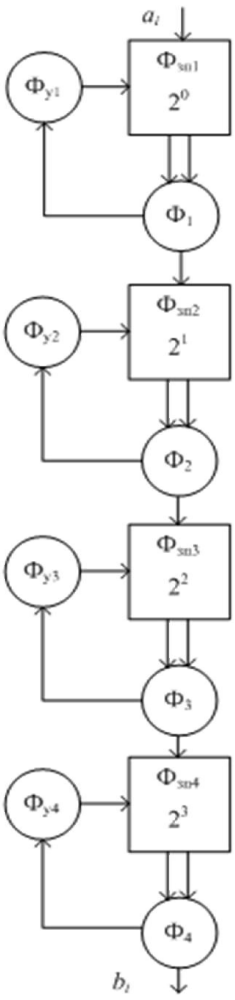

Рисунок 2.5 – Конкретизований граф сортування даних на вертикальну вісь Y на основі одноканального двошляхового злиття

Третім варіантом отримання конкретизованого графу сортування даних методом злиття є використання для його реалізації базових операцій багатоканального двошляхового злиття [66]. Необхідна кількість базових операцій двошляхового паралельного злиття для сортування масиву з N даних, та кількості каналів m визначається за формулою:

$$g = \log_2 \frac{N}{m} \qquad (2.11)$$

Оціночна інтенсивністю сортування масивів даних на основі багатоканального двошляхового злиття для лінійної проекції на вертикальну вісь *Y* дорівнює:

$$D_c = \frac{mn_s}{T_k} \quad (2.12)$$

Основним шляхом підвищення інтенсивності сортування масивів даних на основі багатоканального двошляхового злиття є збільшення кількості каналів [68].

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА СИНТЕЗ АПАПРАТНОГО ПРИСТРОЮ СОРТУВАННЯ ДАНИХ МЕТОДОМ ЗЛИТТЯ

3.1 Розробка програми сортування на CPU

Для розробки програми сортування даних методом злиття використовується програмне середовище Visual Studio 2015 (рис.3.1). Щоб реалізувати паралельні обчислення за допомогою центрального процесора CPU використовується мова програмування C++ та розширення CUDA.

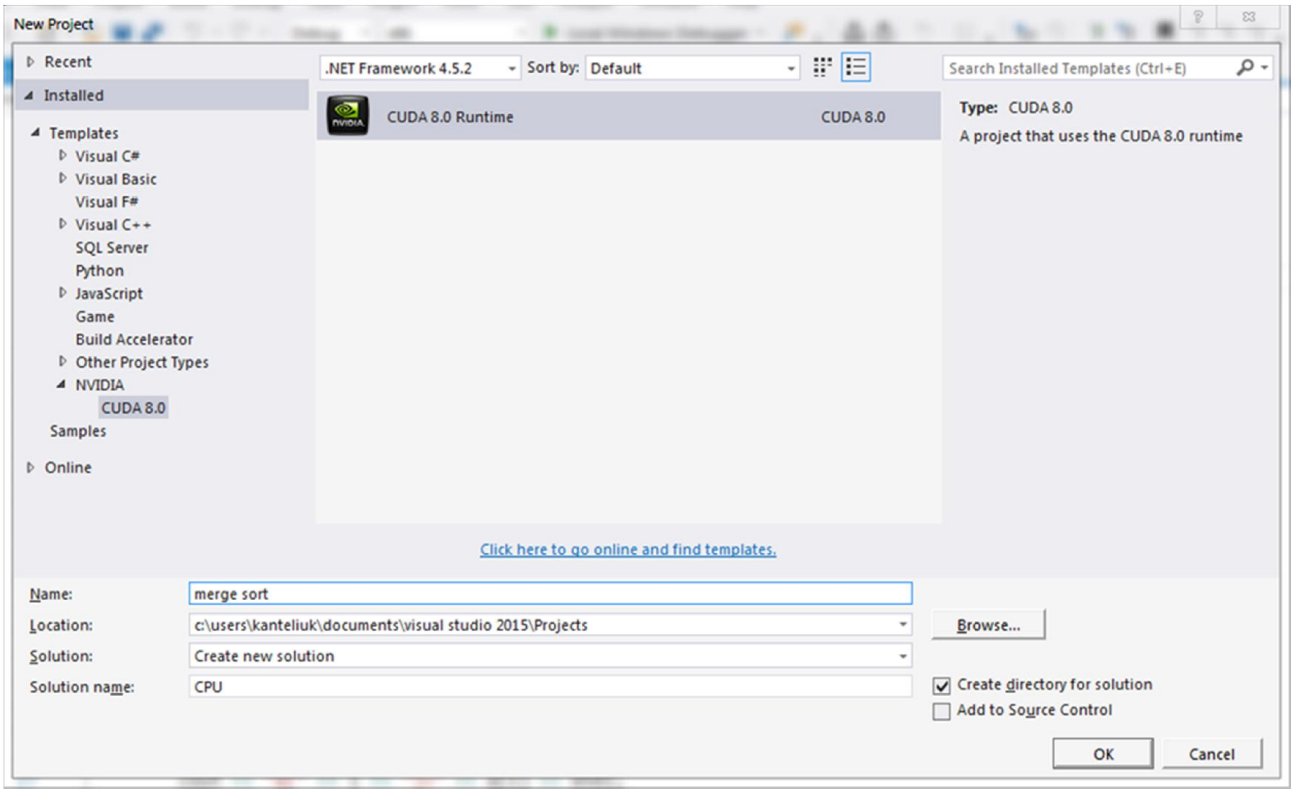

Рисунок 3.1 – Створення нового проекту на основі розширення CUDA

Назва даного проекту – merge sort. Після обчислення коду програми створюється новий за стосунок, який запускає програму паралельного сортування даних методом злиття.

Для сортування масиву даних методом злиття , була створена наступна функція:

```
void merge(int *, int, int, int);
void mergesort(int *a, int low, int high)
```

```
{
        int mid;

        if (low < high)
        {
                mid = (low + high) / 2;
                mergesort(a, low, mid);
                mergesort(a, mid + 1, high);
                merge(a, low, high, mid);
        }
        return;
}
void merge(int *a, int low, int high, int mid)
{
        int i, j, k, c[50];
        i = low;
        k = low;
        j = mid + 1;
        while (i <= mid && j <= high)
        {
                if (a[i] < a[j])
                {
                        c[k] = a[i];
                        k++;
                        i++;
                }
                else
                {
                        c[k] = a[j];
                        k++;
                        j++;
                }
        }
        while (i <= mid)
        {
                c[k] = a[i];
                k++;
                i++;
        }
        while (j <= high)
        {
                c[k] = a[j];
                k++;
                j++;
        }
        for (i = low; i < k; i++)
```

```
        {
                a[i] = c[i];
        }
}
```

Функція void використовується тоді, коли функції не треба повертати будь-яке значення. Припустимо їй треба тільки що небудь порахувати і вивести результат ніде не зберігаючи.

Функція mid = (low + high) / 2 розділяє масив із чисел на дві рівні частини для виконання сортування злиттям до кожної з них. На початку сортування вхідний масив чисел розбивається на N/2 упорядкованих масивів. У результаті виконання першої базової операції формуються впорядковані масиви. Під час сортування в дві допоміжні черги з основної поміщаються перші дві відсортовані підпослідовності, які потім зливаються в одну і результат записується в тимчасову чергу . Потім з основної черги беруться наступні дві відсортовані підпослідовності і так до тих пір доки основна черга не стане порожньою. Після цього послідовність з тимчасової черги переміщається в основну чергу. І знову продовжується сортування злиттям двох відсортованих під послідовностей. Сортування триватиме до тих пір поки довжина відсортованої підпослідовності не стане рівною довжині самої послідовності.

Функції злиття mergesort(a, low, mid) та mergesort(a, mid + 1, high) розділюють початково заданий масив даних на дві частини відповідно , де

- low – початок масиву даних;
- mid – середина масиву даних;
- high – кінець масиву даних.

На рисунку 3.2 зображено запуск розробленої програми сортування даних на центральному процесорі CPU та створення застосунку для проведення паралельного сортування методом злиття.

- 1 succeeded – проект успішно запущений в роботу;
- 0 failed – у коді програми немає помилок;
- 0 up-to-date – немає додаткових уточнень у програмному коді;
- 0 skipped – немає пропущених функцій дій.

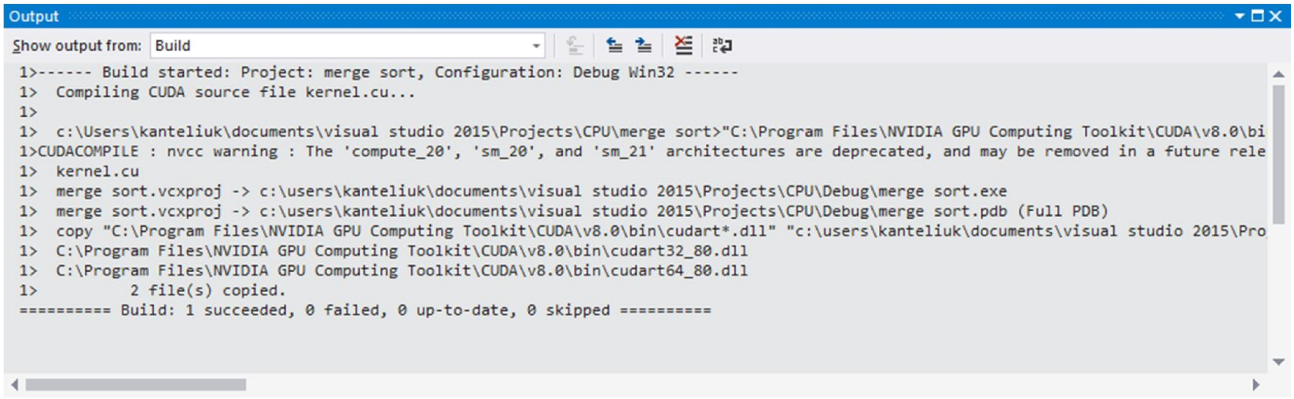

Рисунок 3.2 – Компіляція програми та створення застосунку для проведення паралельного сортування методом злиття

На рисунку 3.3 представлено запуск розробленої програми за допомогою функції Start Debugging – почати налагодження. При натисканні клавіші F5 (почати налагодження) в Visual Studio налагоджувальник – це інструмент, який дозволяє спостерігати за поведінкою під час виконання програми і знайти логічні помилки. Start Debugging працює з усіма мовами програмування Visual Studio і пов'язаних з ними бібліотек. За допомогою Start Debugging, можна перервати або призупинити виконання програми, щоб вивчити код, оцінювати і редагувати змінні в програмі, переглядати регістри, дивіться інструкції, створені з вихідного коду, а також переглядати обсяг пам'яті, який використовується додатком.

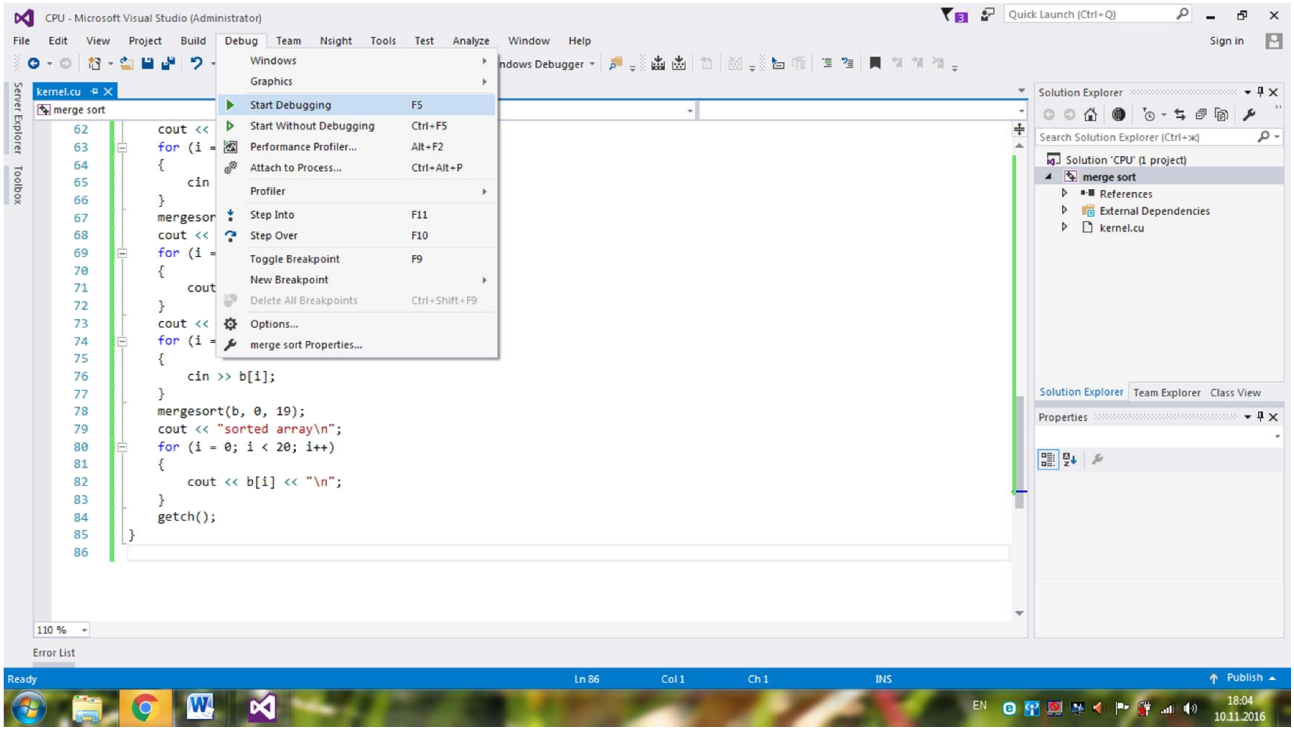

Рисунок 3.3 – Запуск програми сортування даних на центральному процесорі

Для сортування масиву даних методом злиття ,із кількістю 20 елементів, створена наступна функція:

```
cout << "Vvedit elementu\n";
        for (i = 0; i < 20; i++)
        {
                cin >> a[i];
        }
        mergesort(a, 0, 19);
        cout << "Vidsortovanuu masuv\n";
        for (i = 0; i < 20; i++)
        {
                cout << a[i] << "\n";
        }
        cout << "Vvedit elementu\n";
        for (i = 0; i < 20; i++)
        {
                cin >> b[i];
        }
        mergesort(b, 0, 19);
        cout << "Vidsortovanuu masuv\n";
        for (i = 0; i < 20; i++)
        {
```

```
            cout << b[i] << "\n";
        }
        getch();
}
```

У головному вікні програми (рис.3.4) з'явилась стрічка «Введіть елементи». Користувач даного програмного забезпечення вводить із клавіатури цілі числа, що повинні відсортуватись на центральному процесорі CPU.

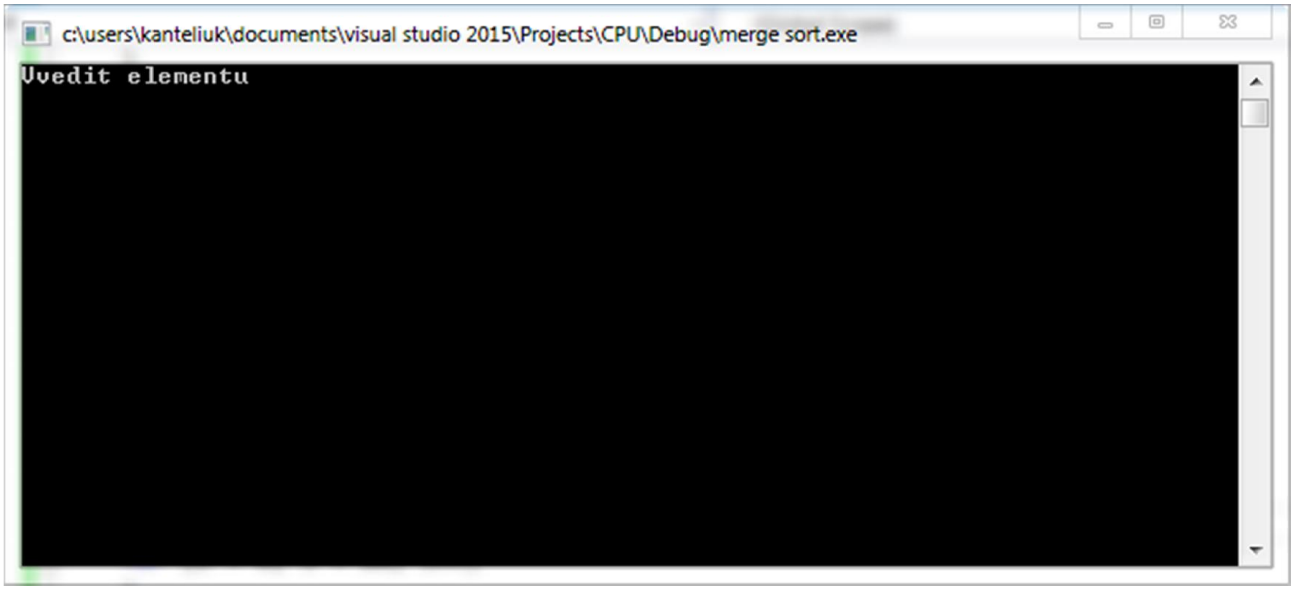

Рисунок 3.4 – Вікно програми «Введіть елементи»

Користувач на запит програми «Введіть елементи» набирає із клавіатури 20 чисел, у даному випадку це : 56,89,74,55,2154,8956,354,21,4,8,458, 785,56,32,68,42,546,895,5,8.

Програма сортування даних методом злиття відсортувала інформацію та видала користувачу результат (рис.3.5).

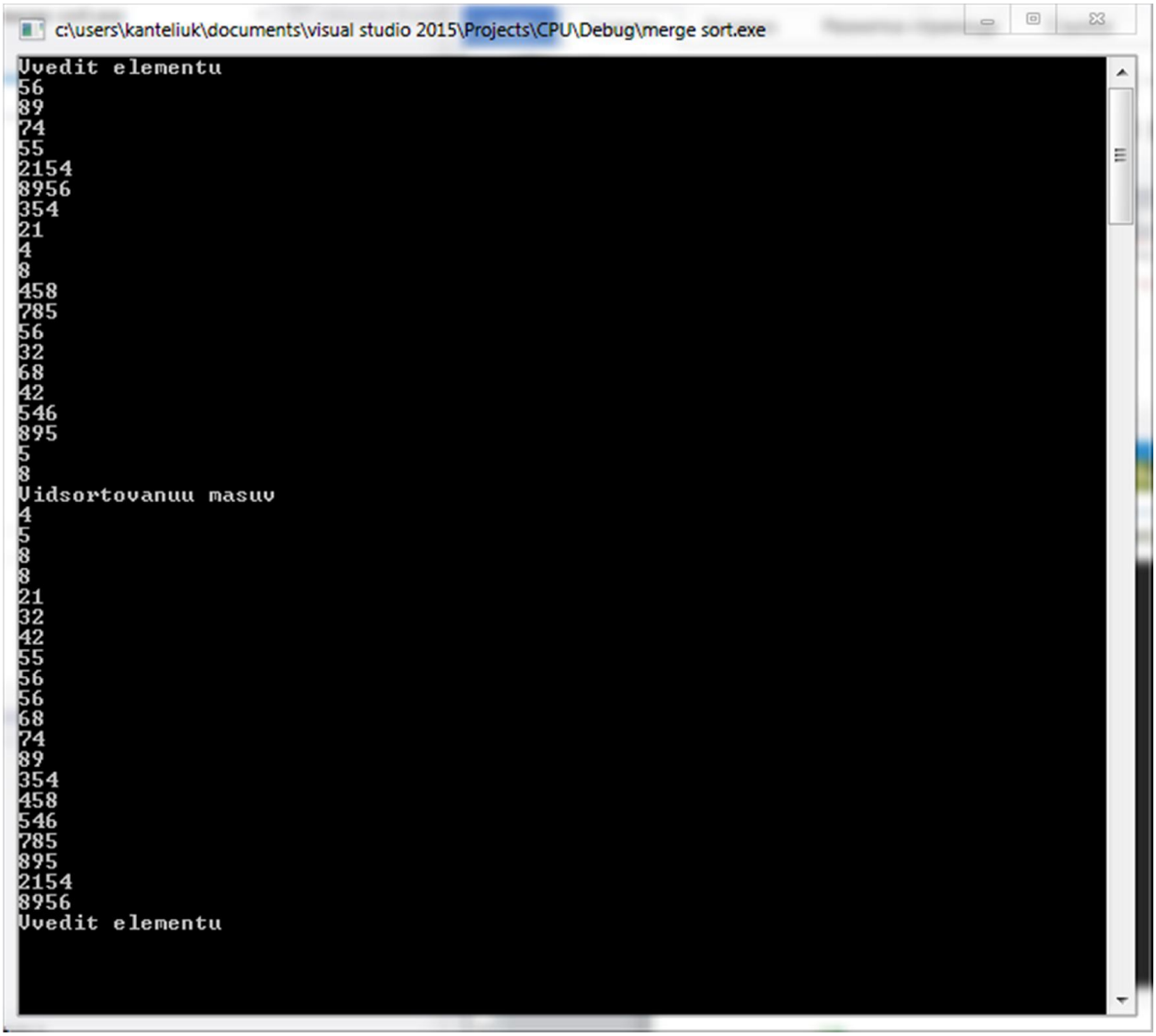

Рисунок 3.5 – Відсортований масив даних методом злиття на центральному процесорі

Для сортування масиву даних методом злиття , із кількістю 7 елементів, створена наступна функція:

```
cout << "Vvedit elementu\n";
        for (i = 0; i < 7; i++)
        {
                cin >> a[i];
        }
        mergesort(a, 0, 6);
        cout << "Vidsortovanuu masuv\n";
        for (i = 0; i < 7; i++)
{
                cout << a[i] << "\n";
```

```
}
cout << "Vvedit elementu\n";
for (i = 0; i < 6; i++)
{
        cin >> b[i];
}
mergesort(b, 0, 6);
cout << "Vidsortovanuu masuv\n";
for (i = 0; i < 7; i++)
{
        cout << b[i] << "\n";
}
getch();
```

Користувач на запит програми «Введіть елементи» набирає із клавіатури 7 чисел, у даному випадку це : 45,89,458,5,2,55,54.

Програма сортування даних методом злиття відсортувала інформацію та видала користувачу результат (рис.3.6).

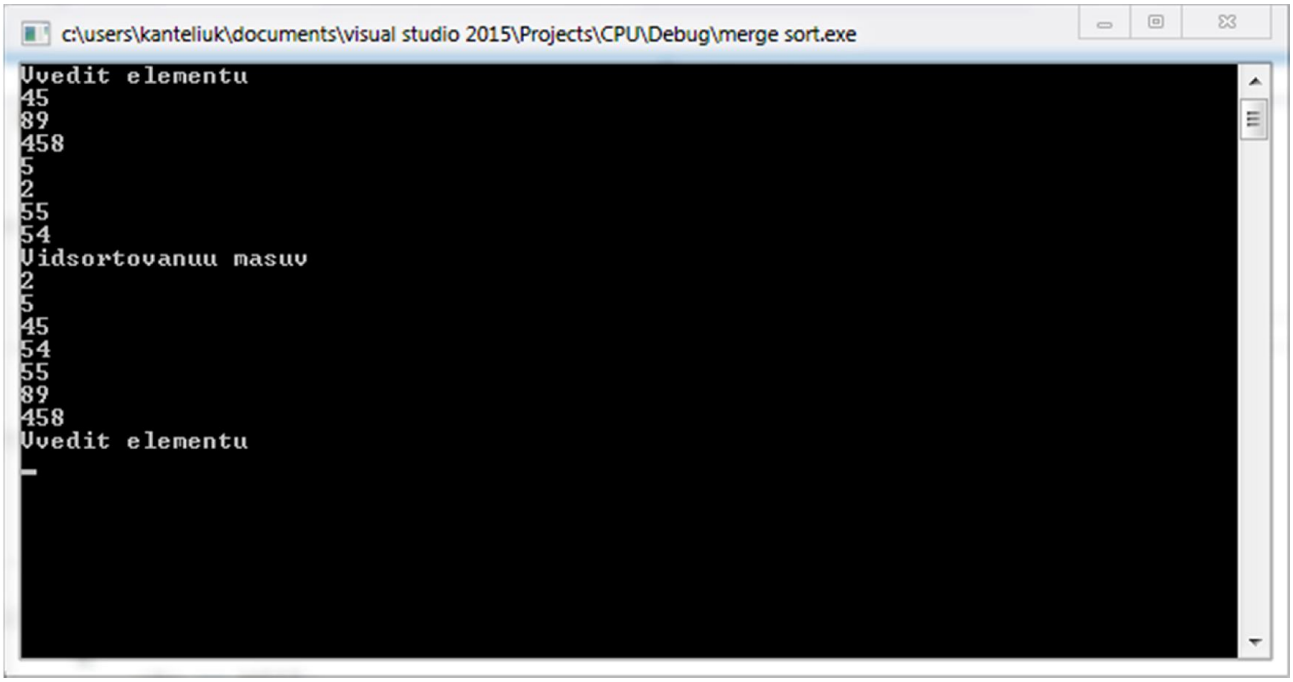

Рисунок 3.6 – Відсортований масив даних методом злиття на центральному процесорі

Відсортований масив даних методом злиття на центральному процесорі виконаний правильно,у порядку зростання введених чисел із клавіатури.

3.2 Розробка програми сортування на GPU

Для сортування масиву даних методом злиття на GPU - процесорі (рис.3.7), була створена наступна функція:

```
_global__ - code = sm_20, sm_21 void merge(int *, int, int, int);
__global__ - code = sm_20, sm_21 void mergesort(int *a, int low, int high)
{
    int mid = blockIdx.x * blockDim.x
            + threadIdx.x;

    if (low < high)
    {
        mid = (low + high) / 2;
        mergesort(a, low, mid);
        mergesort(a, mid + 1, high);
        merge(a, low, high, mid);
    }
    return;
}
```

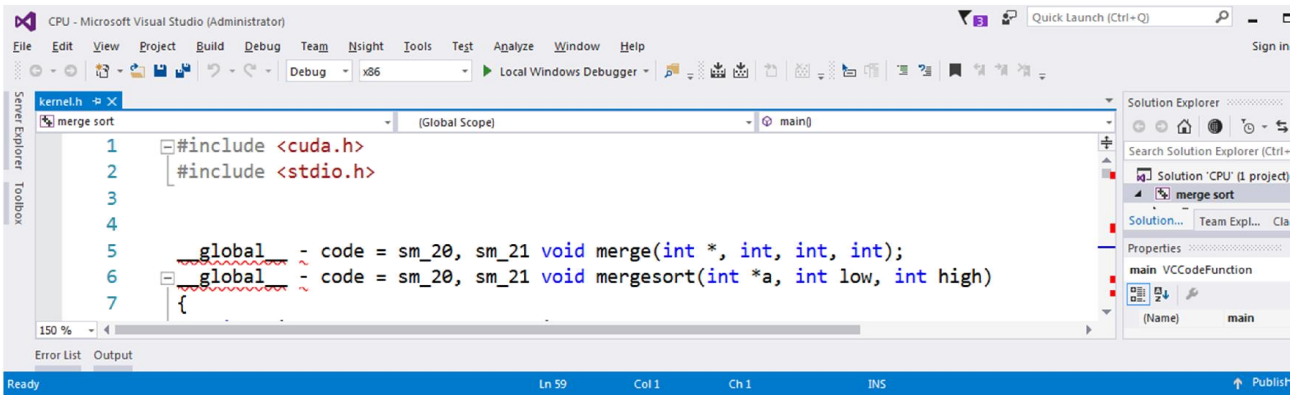

Рисунок 3.7 – Код для запуску обрахунків на GPU

Функція __global__ викликає обрахунок даних на GPU.В коді це є першою ознакою того,що після виконання даної функції , програма

обраховуватиметься паралельно. Оскільки, Visual Studio 2015 та набір інструментів CUDA 8.0 використовують архітектуру графічних процесорів з індексом 35 та вище,для обрахунку на нижчій архітектурі в програмному коді,потрібно вказати ,яка архітектура буде використовуватись.

Для цього використовується функція - code = sm_20, sm_21,де sm_20, sm_21вказуються індекс архітектури.

Оскільки, код програми запускається з використанням графічного процесора, потрібно встановити паралельні потоки даних (рис.3.8).

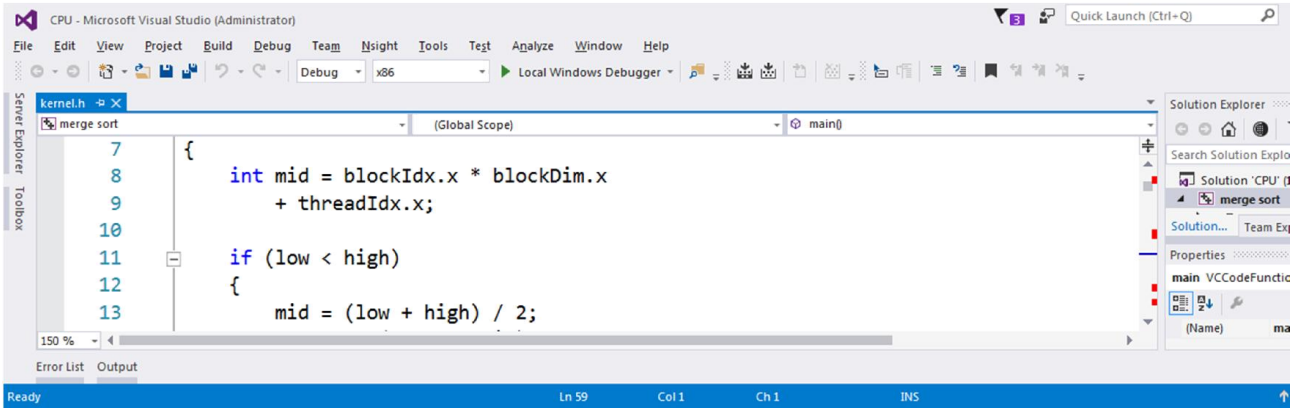

Рисунок 3.8 – Встановлення паралельних потоків

Функція threadIdx.x встановлює паралельні потоки для виконання сортування методом злиття.

Програмний код виконується на центральному на графічному процесорах, тому, потрібно вводити два види даних для двох пристроїв (рис.3.9).

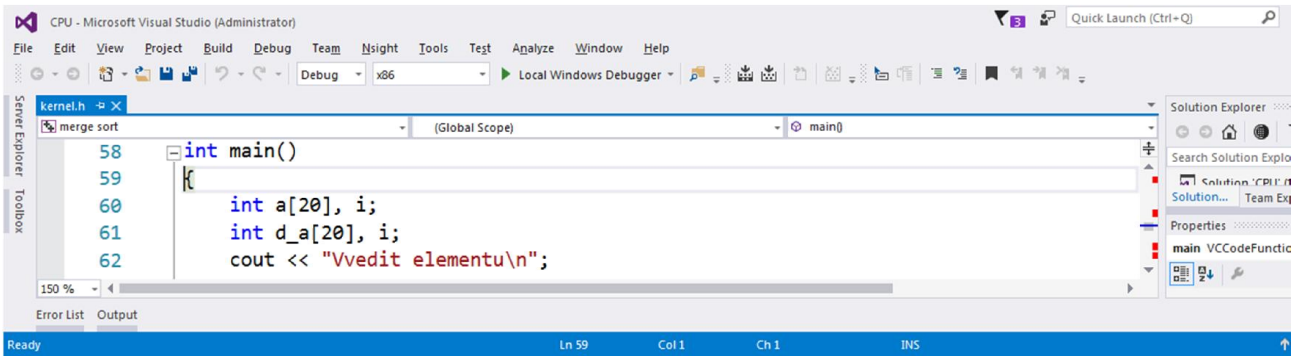

Рисунок 3.9 – Виділення окремих даних для CPU та GPU

Для виділення змінних для GPU в програмному коді застосовується індекс d «d_a», скорочено від пристрій (англ.- device).

Далі потрібно ініціалізувати виділення пам’яті для пристроїв. Першою виділяється пам’ять на центральному процесорі (рис 3.10).

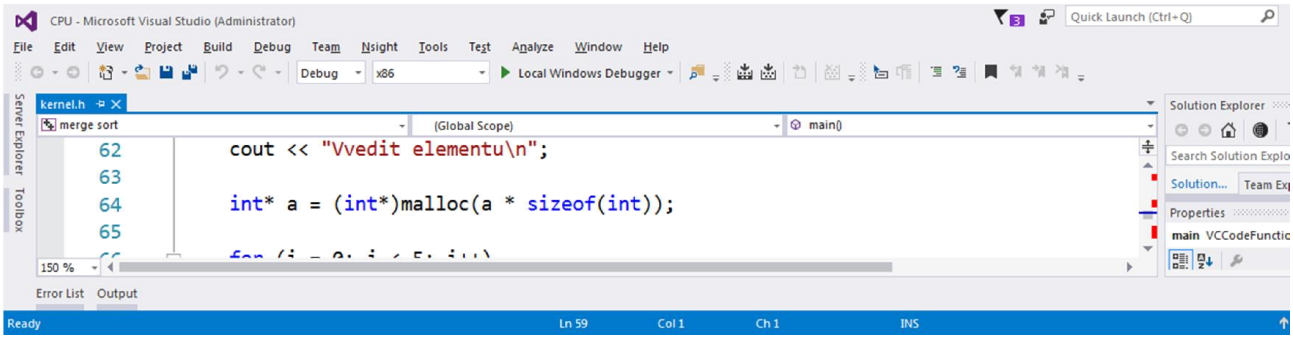

Рисунок 3.10 – Ініціалізація пам’яті на CPU

Пам’ять CPU у коді викликається функцією malloc. Далі виділяється пам’ять на GPU (рис 3.11). Для цього використовується схожа функція cudaMalloc.

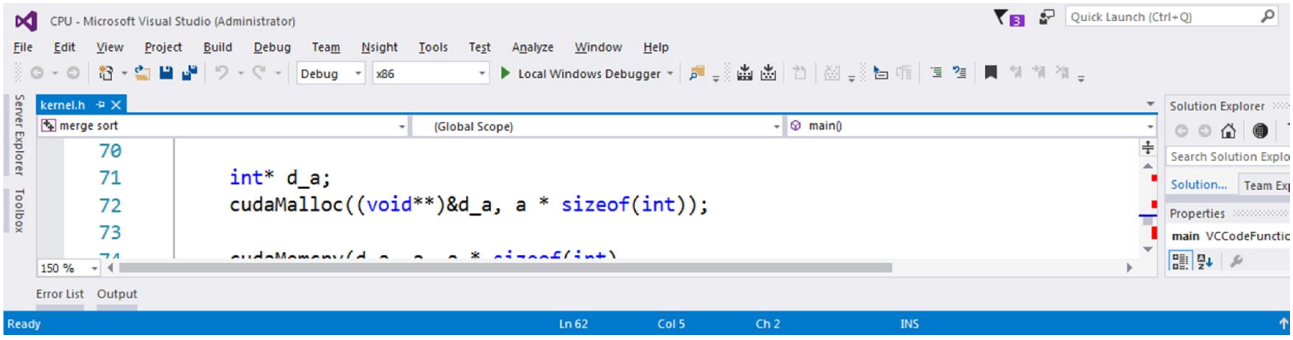

Рисунок 3.11 – Ініціалізація пам’яті на GPU

Після виділення пам’яті на центральному та графічному процесорах, потрібно скопіювати дані із CPU на GPU (рис.3.12).

Функція cudaMemcpy запускає копіювання даних на пристрій. У ній вказується: призначення(куди копіюються дані), джерело (дані з центрального процесора CPU) та змінна.

Далі прописується шлях для копіювання , в даному випадку із центрального процесора (Host) у графічний (Device). В коді має наступний вигляд :

cudaMemcpy(d_a, a, a * sizeof(int),

cudaMemcpyHostToDevice)

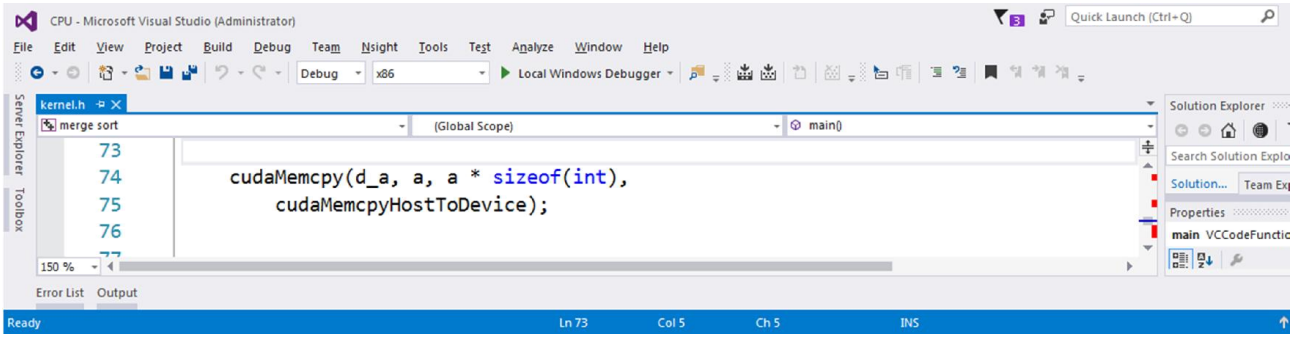

Рисунок 3.12 – Копіювання даних із CPU на GPU

Коли дані програми скопійовані на графічний процесор GPU, потрібно провести на ньому сортування (рис.3.13.)

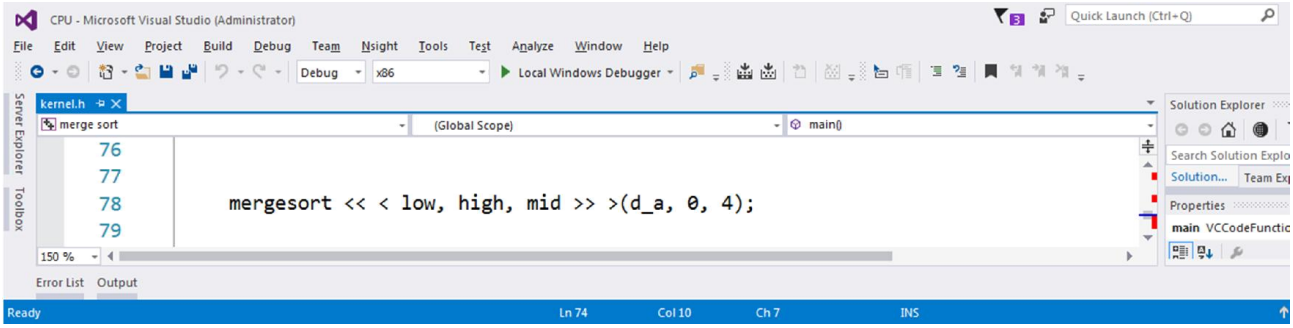

Рисунок 3.13 – Запуск сортування на GPU

Для проведення паралельного сортування злиттям на GPU процесорі, до функції mergesort додається елемент <<< >>>, в якому вказуються змінні для сортування.

Після проведення сортування на GPU, відсортовані масиви даних потрібно скопіювати на центральний процесор CPU. Для цього застосовується функція cudaMemcpy. Дана функція копіює інформацію з GPU на CPU (рис.3.14).

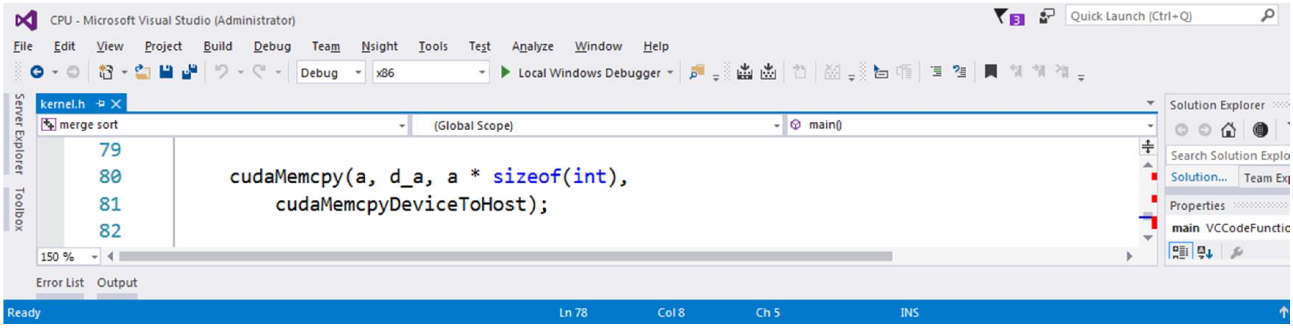

Рисунок 3.14 – Копіювання інформації з GPU на CPU

Після виведення відсортованих даних,очищається пам’ять GPU та CPU (рис.3.15).

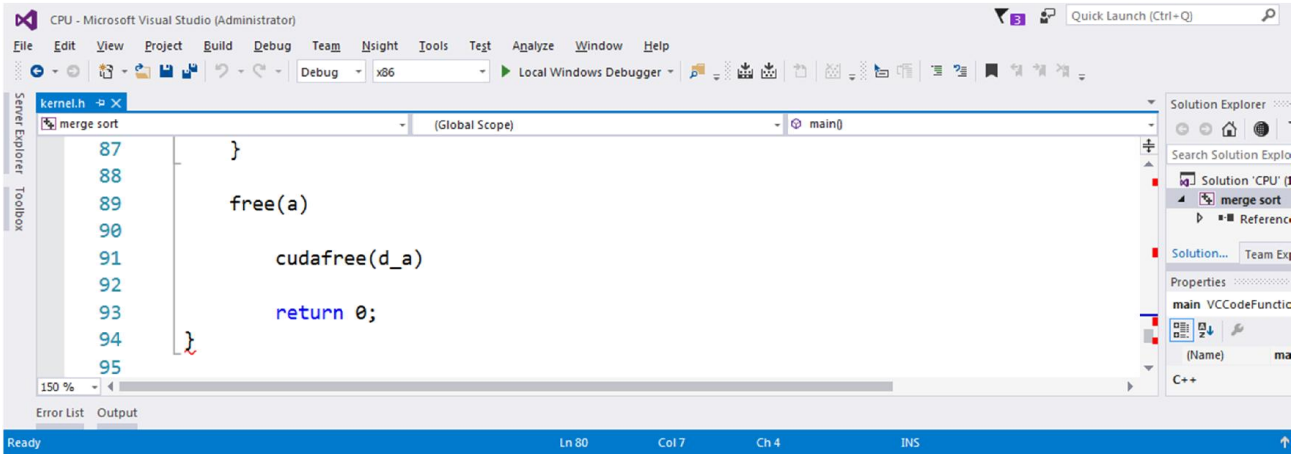

Рисунок 3.15 – Очищення пам’яті на GPU та CPU процесорах

Для сортування масиву даних методом злиття ,із кількістю 5 елементів, створена наступна функція:

```
cout << "Vvedit elementu\n";

    int* a = (int*)malloc(a * sizeof(int));

    for (i = 0; i < 5; i++)
    {
            cin >> a[i];
    }

    int* d_a;
    cudaMalloc((void**)&d_a, a * sizeof(int));

    cudaMemcpy(d_a, a, a * sizeof(int),
```

```
        cudaMemcpyHostToDevice);


    mergesort <<< low, high, mid >>>(d_a, 0, 4);

    cudaMemcpy(a, d_a, a * sizeof(int),
        cudaMemcpyDeviceToHost);

    cout << "Vidsortovanuu masuv\n";
    for (i = 0; i < 5; i++);
    {
        cout << a[i] << "\n";
    }

    free(a);

    cudafree(d_a);

    return 0;

}
```

У головному вікні програми (рис.3.16) з'явилась стрічка «Введіть елементи,які сортуватимуться на GPU». Користувач даного програмного забезпечення вводить із клавіатури цілі числа, що повинні відсортуватись на графічному процесорі.

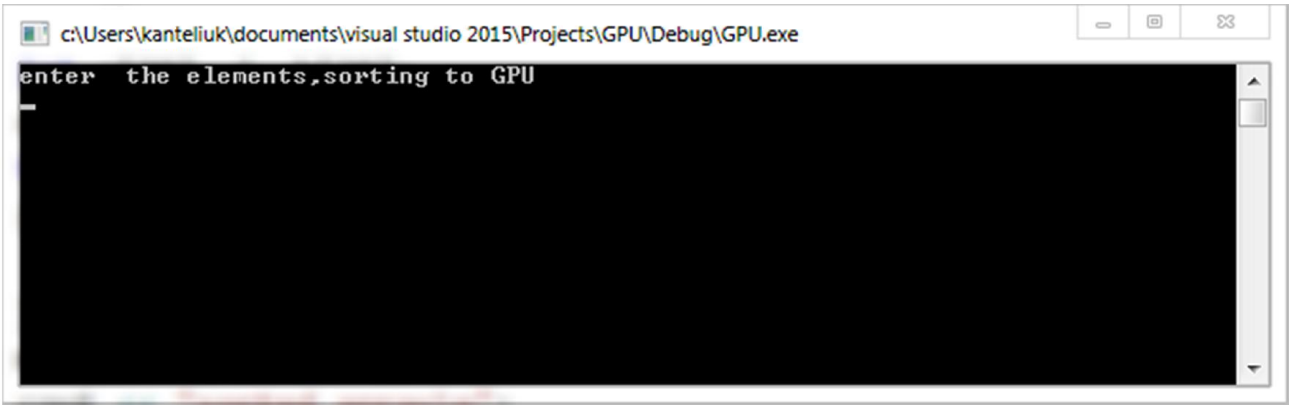

Рисунок 3.16 – Вікно програми «Введіть елементи,які сортуватимуться на GPU»

Користувач на запит програми «Введіть елементи,які сортуватимуться на GPU» набирає із клавіатури 5 чисел, у даному випадку це : 89,5648,24,8,335.

Програма сортування даних методом злиття відсортувала інформацію та видала користувачу результат (рис.3.17).

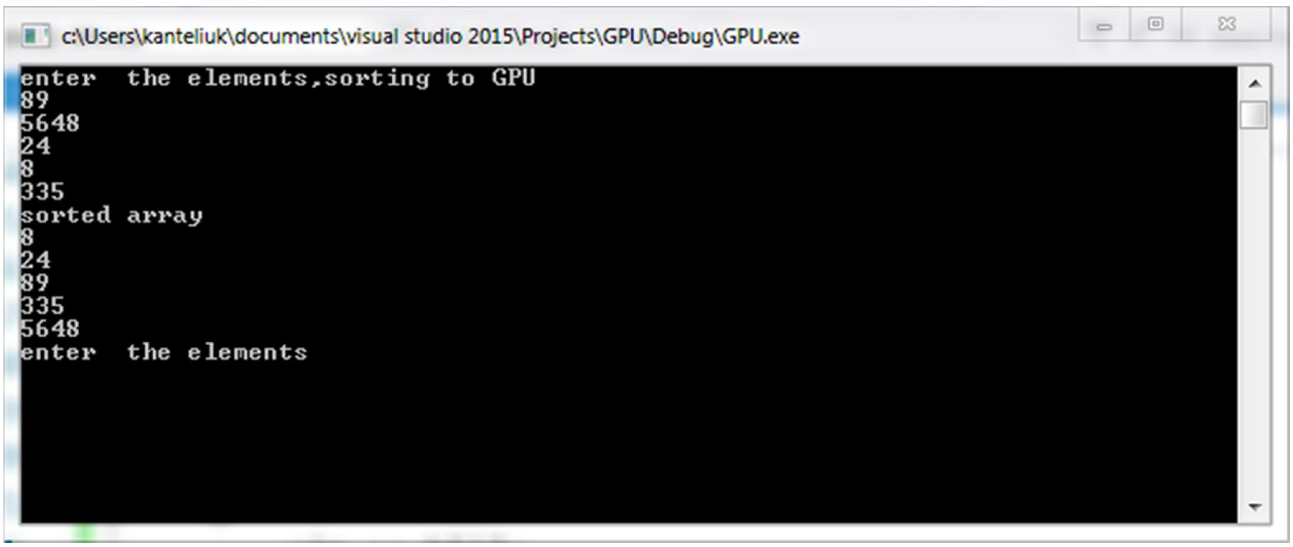

Рисунок 3.17 – Відсортований масив даних методом злиття на графічному процесорі

Для сортування масиву даних методом злиття ,із кількістю 20 елементів, створена наступна функція:

```
cout << "Vvedit elementu\n";

    int* a = (int*)malloc(a * sizeof(int));
    for (i = 0; i < 20; i++)
    {
        cin >> a[i];
    }
    int* d_a;
    cudaMalloc((void**)&d_a, a * sizeof(int));

    cudaMemcpy(d_a, a, a * sizeof(int),

        cudaMemcpyHostToDevice);

    mergesort <<< low, high, mid >>>(d_a, 0, 19);

    cudaMemcpy(a, d_a, a * sizeof(int),

        cudaMemcpyDeviceToHost);
```

```
        cout << "Vidsortovanuu masuv\n";
for (i = 0; i < 20; i++);
        {
                cout << a[i] << "\n";
        }
        free(a);

        cudafree(d_a);
        return 0;
}
```

3.3 Тестування та верифікація розробленого програмного забезпечення

Тестування програмного забезпечення - це процес, що використовується для виміру якості розроблюваного програмного забезпечення. До цього процесу входить запуск програми з метою знайдення помилок. Якість - це суб'єктивне поняття,тому,тестування не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. Існує багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів - це по суті дослідницький процес, а не тільки створення і виконання рутинної процедури. Тестова діяльність, що пов'язана з аналізом результатів розробки програмного забезпечення, називається статичним тестуванням. На етапі статичного тестування перевіряється вся документація, отримана як результат життєвого циклу програми. Це і технічне завдання, і специфікація, і вихідний текст програми на мові програмування. Воно передбачає перевірку програмних кодів, контроль та перевірку програми без запуску на комп'ютері. Динамічні методи застосовуються в процесі безпосереднього виконання програми. Коректність програмного засобу перевіряється на безлічі тестів або наборів підготовлених вхідних даних. При прогоні кожного тесту збираються та аналізуються дані про відмови та збої в роботі програми.

Основними системними характеристиками ноутбука, на якому тестувалось дане програмне забезпечення,наведені у наступній таблиці:

Таблиця 3.1 - Системні характеристики ноутбука Asus P53S

Технічні характеристики	Опис
Екран	15.6" (1366x768) WXGA HD
Процесор	Двоядерний Intel Core i5-2450M (2.5 ГГц)
Тип оперативної пам'яті	DDR3 1333 МГц 4096 MB
Графічний адаптер	nVidia GeForce GT 520M, 1 ГБ
Об'єм накопичувача	500 ГБ
Мережеві адаптери	Wi-Fi 802.11 B / G / N, Bluetooth 4.0, Ethernet (10/100)
Роз'єми і порти введення-виведення	3 порта USB 2.0 / VGA / HDMI / LAN (RJ-45)

Для перевірки правильності роботи розробленого програмного продукту, проведено тестування з різними умовами.

Оскільки, сортування злиттям полягає у тому, щоб на початку сортування вхідний масив чисел розбити на N/2 упорядкованих масиви.

Функцію «mid = (low + high) / 2» замінимо на «mid = (low + high)» (рис.3.18).

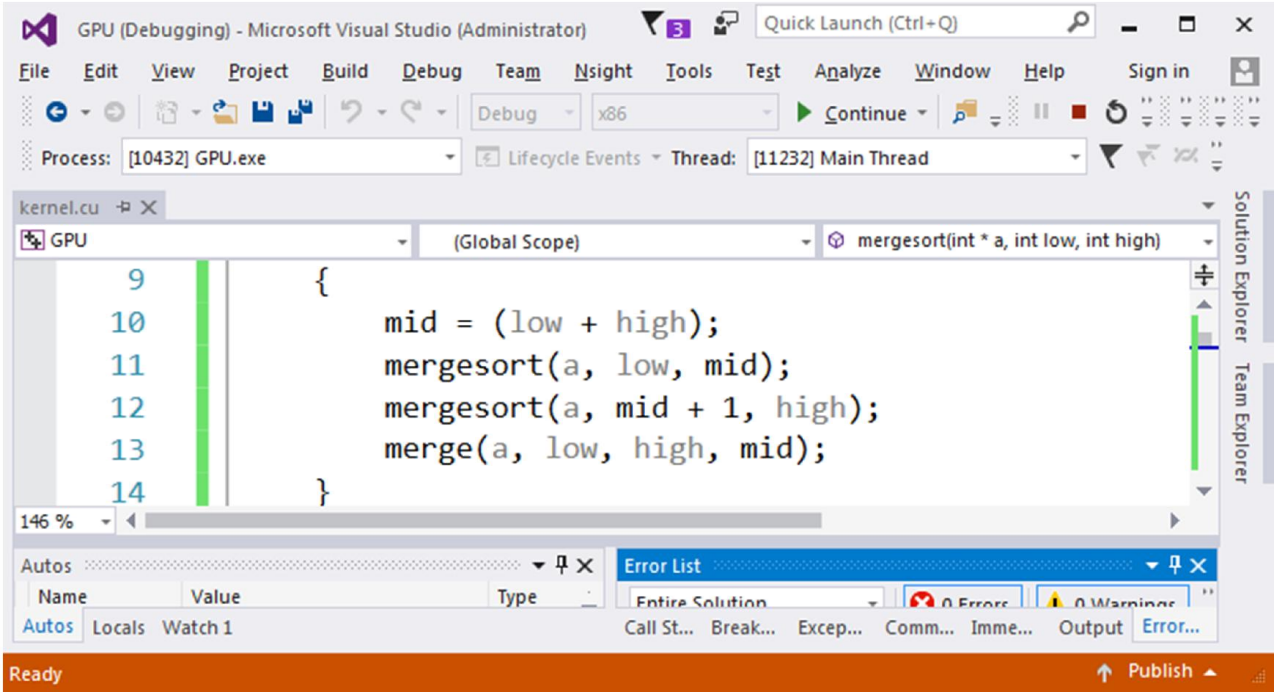

Рисунок 3.18 – Заміна функції

На рисунку 3.19 зображено запуск розробленої програми сортування даних на центральному процесорі CPU та створення застосунку для проведення паралельного сортування методом злиття.

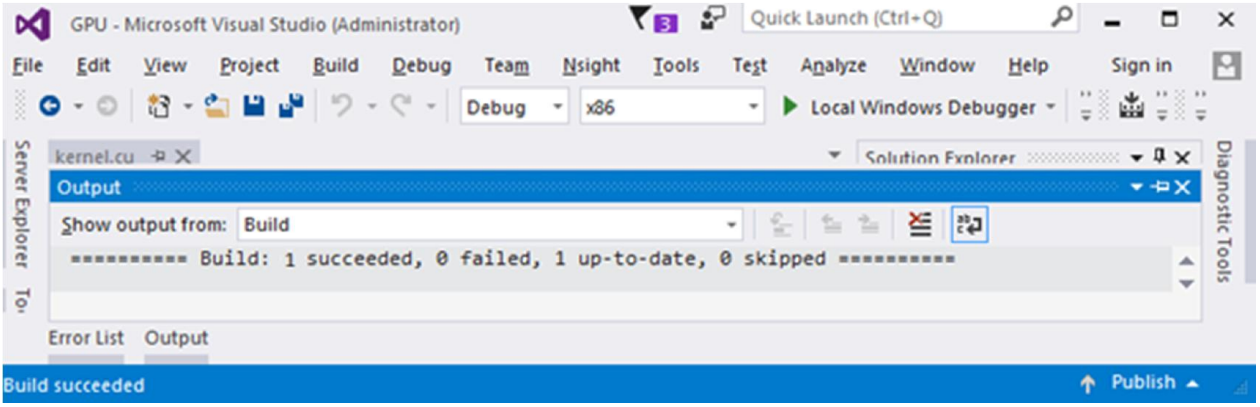

Рисунок 3.19 – Компіляція програми та створення застосунку для проведення паралельного сортування методом злиття

Із рисунка 3.19 видно :

- 1 succeeded – проект успішно запущений в роботу;
- 0 failed – у коді програми немає помилок;
- 1 up-to-date – одне додаткових уточнення у програмному коді;
- 0 skipped – немає пропущених функцій .

На рисунку 3.20 введено із клавіатури 20 довільних чисел для їх сортування методом злиття

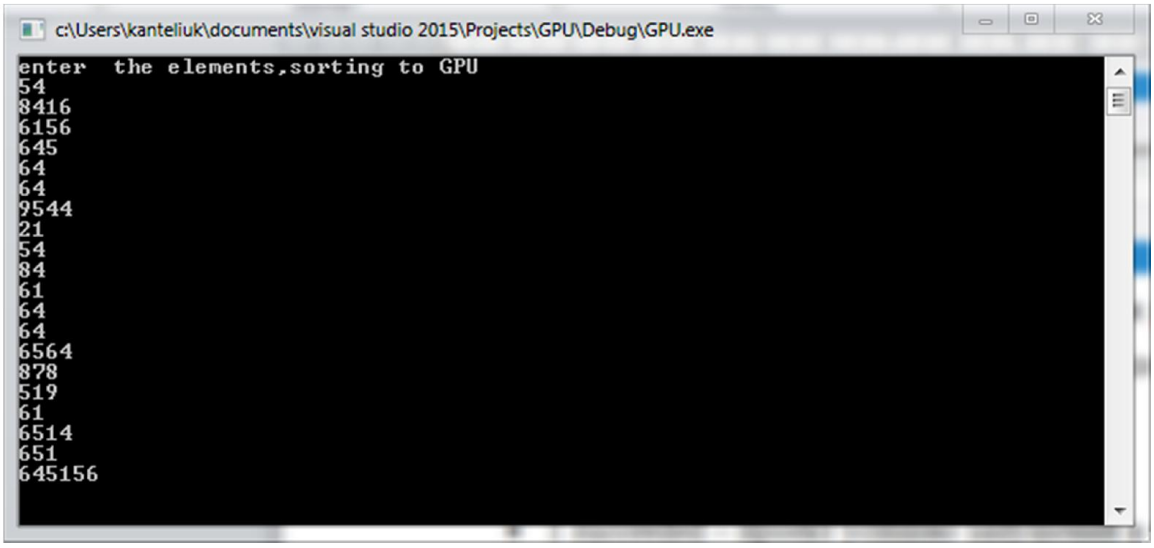

Рисунок 3.20 – Ввід елементів у зміненому коді програми

Заміна функції «mid = (low + high) / 2» на «mid = (low + high)» видає помилку (рис.3.21).

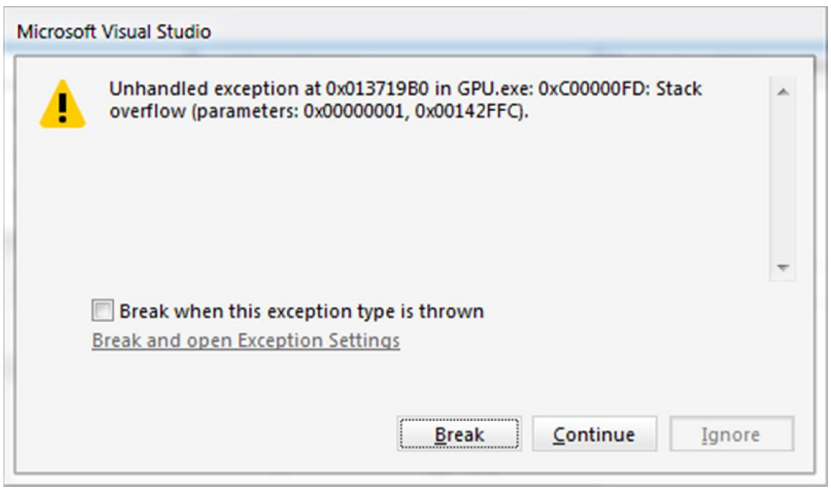

Рисунок 3.21 – Помилка роботи програми при зміні вихідного коду

Користувач на запит програми «enter the elements,sorting to GPU» (рис.3.22) набирає із клавіатури 20 чисел, у даному випадку це : 32,5684,7845,24,558,9642,124,65,87,55,12,205,89,54,662,56,44,21,891,5.

Програма сортування даних методом злиття відсортувала інформацію та видала користувачу результат (рис.3.23).

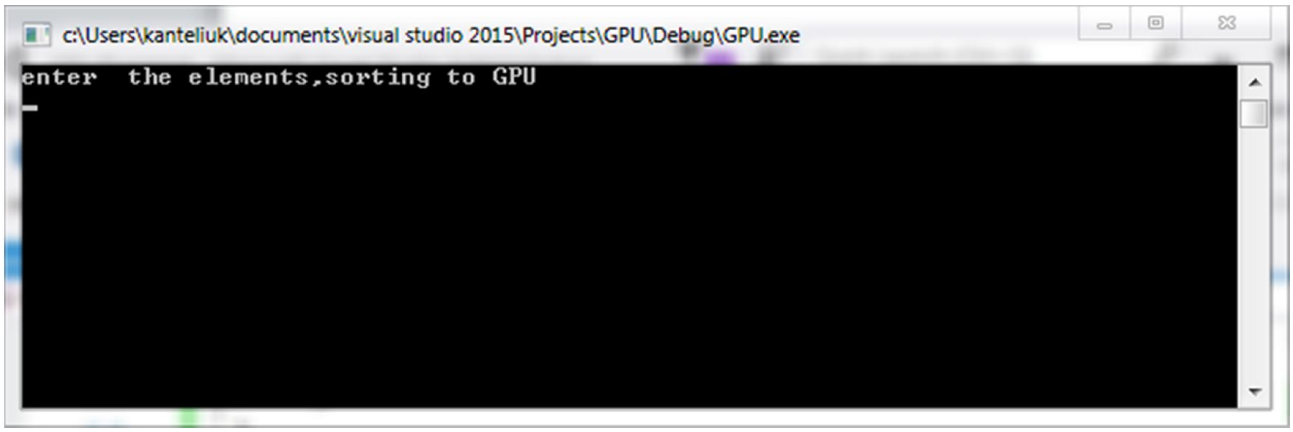

Рисунок 3.22 – Ввід елементів,що будуть сортуватись на GPU

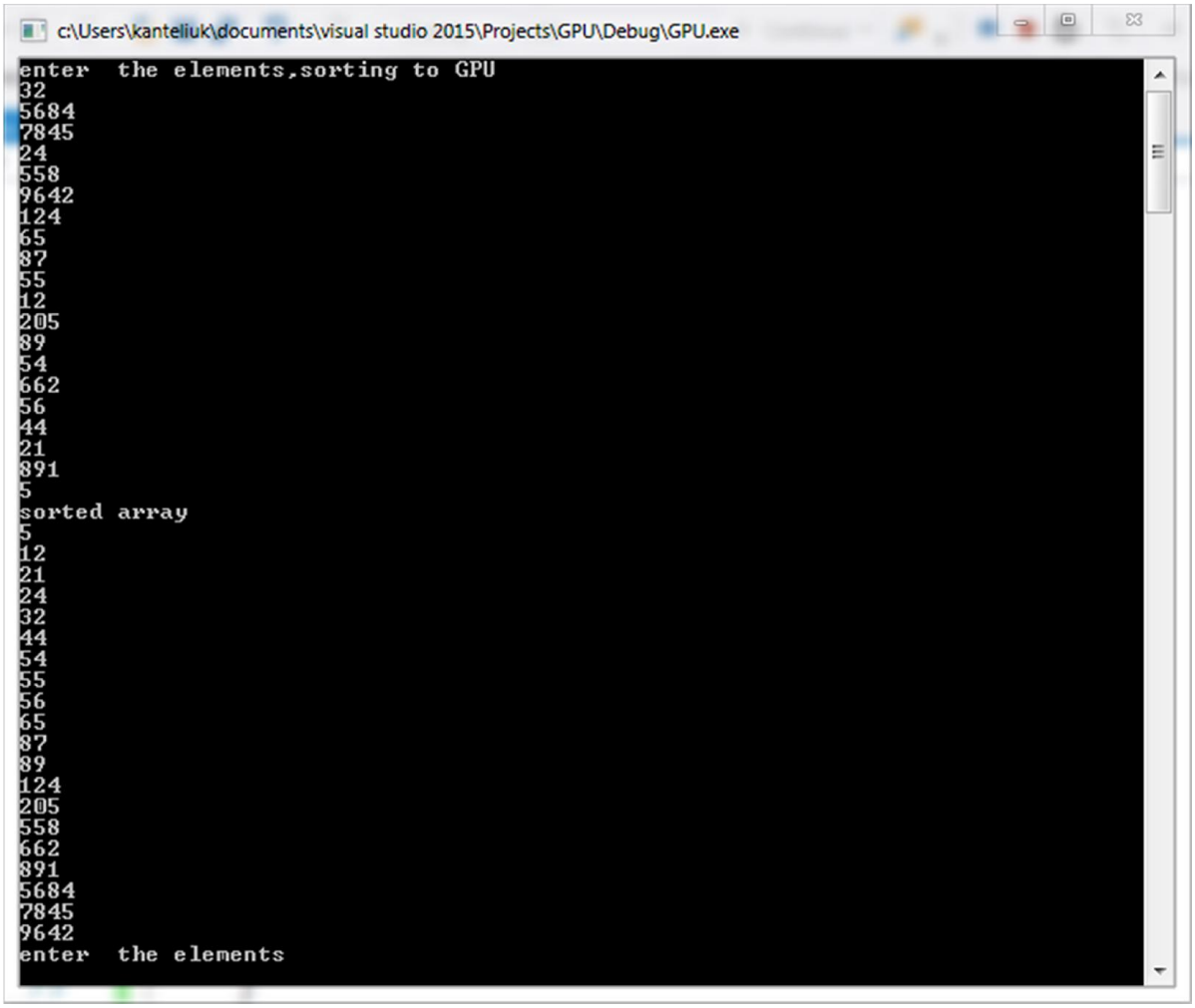

Рисунок 3.23 – Відсортований масив даних методом злиття на графічному процесорі

Відсортований масив даних методом злиття на графічному процесорі виконаний правильно,у порядку зростання введених чисел із клавіатури.

ВИСНОВКИ

1. Паралельне сортування масивів даних з використанням графічного процесора GPU та програмної моделі CUDA можна забезпечити при комплексному підході, який охоплює: дослідження методів і алгоритмів паралельного сортування великих масивів даних; графові моделі алгоритмів паралельного сортування масивів даних; архітектуру графічного процесора GPU та програмну модель CUDA.

2.Сортування масивів даних методом злиття ґрунтується на однотипних базових елементарних операціях порівняння двох чисел та їх перестановки, кількість яких рівна .

3.Для розробки програмних засобів паралельного сортування масивів даних з використанням графічного процесора GPU та програмної моделі CUDA алгоритм сортування необхідно його подати у вигляду потокового графу, який забезпечить виявлення паралелізму та можливість управляти ним для знаходження оптимальних просторово-часових рішень.

4.Розроблення високоефективних паралельних структур для сортування інтенсивних потоків даних у реальному часі методом злиття найдоцільніше здійснювати при інтегрованому підході, який охоплює методи, алгоритми, структури і НВІС-технологію та враховує особливості конкретного застосування.

5.У потоковому пристрої сортування даних узгодження інтенсивності надходження даних із інтенсивністю сортування досягається зміною кількості каналів двошляхового злиття.