

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТИ З ЛАБОРАТОРНИХ РОБІТ

Виконав
студент групи КН-23-1
Полинько І.М.
Перевірила
доцент кафедри АІС
Істоміна Н. М.

Кременчук 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 1

Виконав

студент групи КН-23-1

Полинько І. М.

Перевірила

доцент кафедри АІС

Істоміна Н. М.

Кременчук 2025

Лабораторна робота №1

Тема: Оцінювання ефективності розпаралеленого алгоритму

Мета: набути навичок оцінювання ефективності розпаралелювання довільного алгоритму.

Хід роботи:

1. Для виконання завдання вибрати довільний процес, що може бути поданий у вигляді алгоритму (можна взяти бізнес процес із завдання для дипломної роботи).
 2. Скласти детальний послідовний алгоритм вибраного процесу. Додати складений алгоритм до звіту.
 3. Розпаралелити послідовний алгоритм використовуючи концепцію необмеженого паралелізму. Додати складений алгоритм до звіту.
 4. Для отриманого алгоритму знайти значення таких характеристик:
 - загальна кількість операцій N ;
 - кількість послідовних операцій n_s ;
 - кількість паралельних операцій n_p ;
 - частка послідовних операцій β ;
 - сумарна висота паралельної форми;
 - ширина паралельної форми;
 - максимальне можливе прискорення R .
- Якщо в алгоритмі наявні декілька паралельних форми, обчислюємо їх інтегральні показники (підсумовуємо їх).

Завдання 1-2:

Створимо детальний послідовний алгоритм приготування за рецептом.

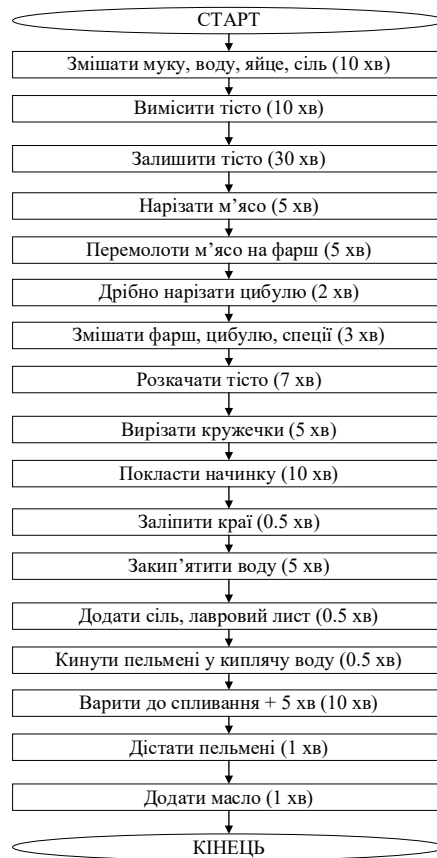


Рисунок 1.1 - Послідовний алгоритм у середовищі Visio

Завдання 3:

Розпаралелимо послідовний алгоритм:

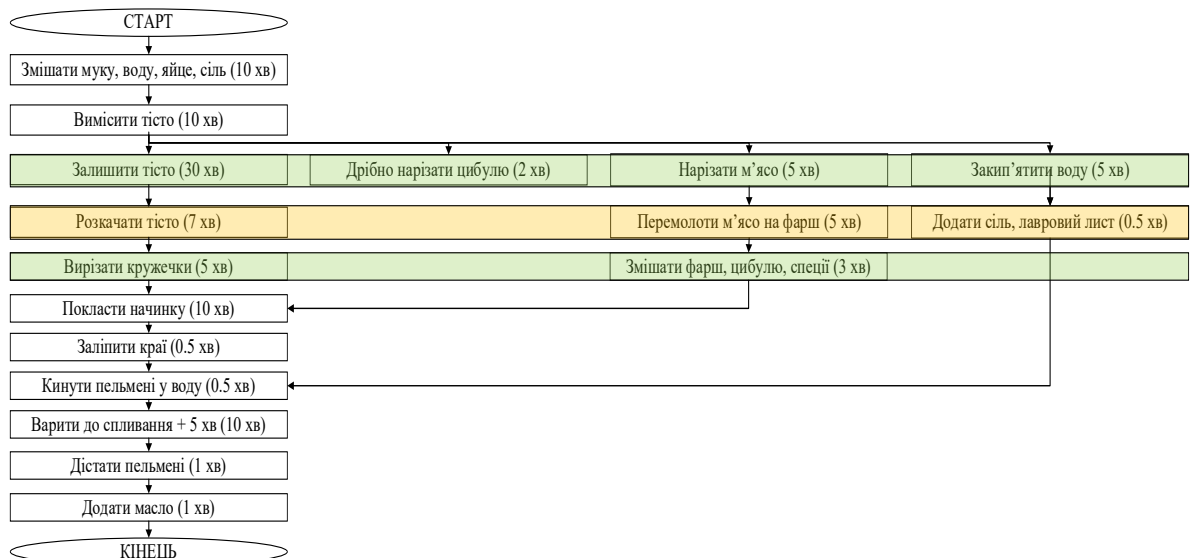


Рисунок 1.2 – Необмежено паралельний алгоритм

Масштаб часу наступний:

- 0.5 хв. – 5 мм.;
- 1 хв. – 5 мм.;
- 2 хв. – 10 мм.;
- 3 хв. – 10 мм.;
- 5 хв. – 15 мм.;
- 7 хв. – 20 мм.;
- 10 хв. – 25 мм.;
- 30 хв. – 35 мм.

Наведемо оновлену діаграму алгоритму з урахуванням масштабу часу.

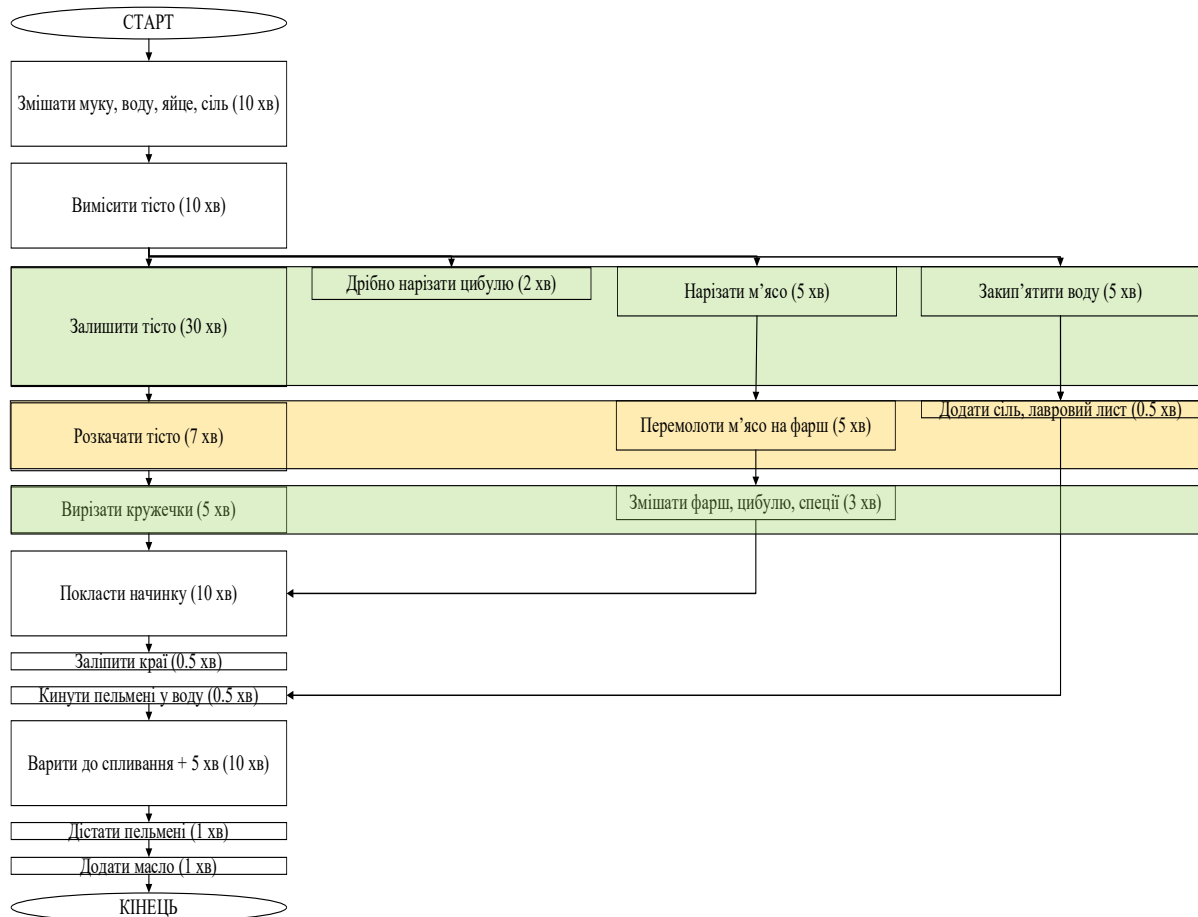


Рисунок 1.3 – Необмежено паралельний алгоритм
з урахуванням масштабу часу

Наведемо діаграму обмеженого паралелізму:

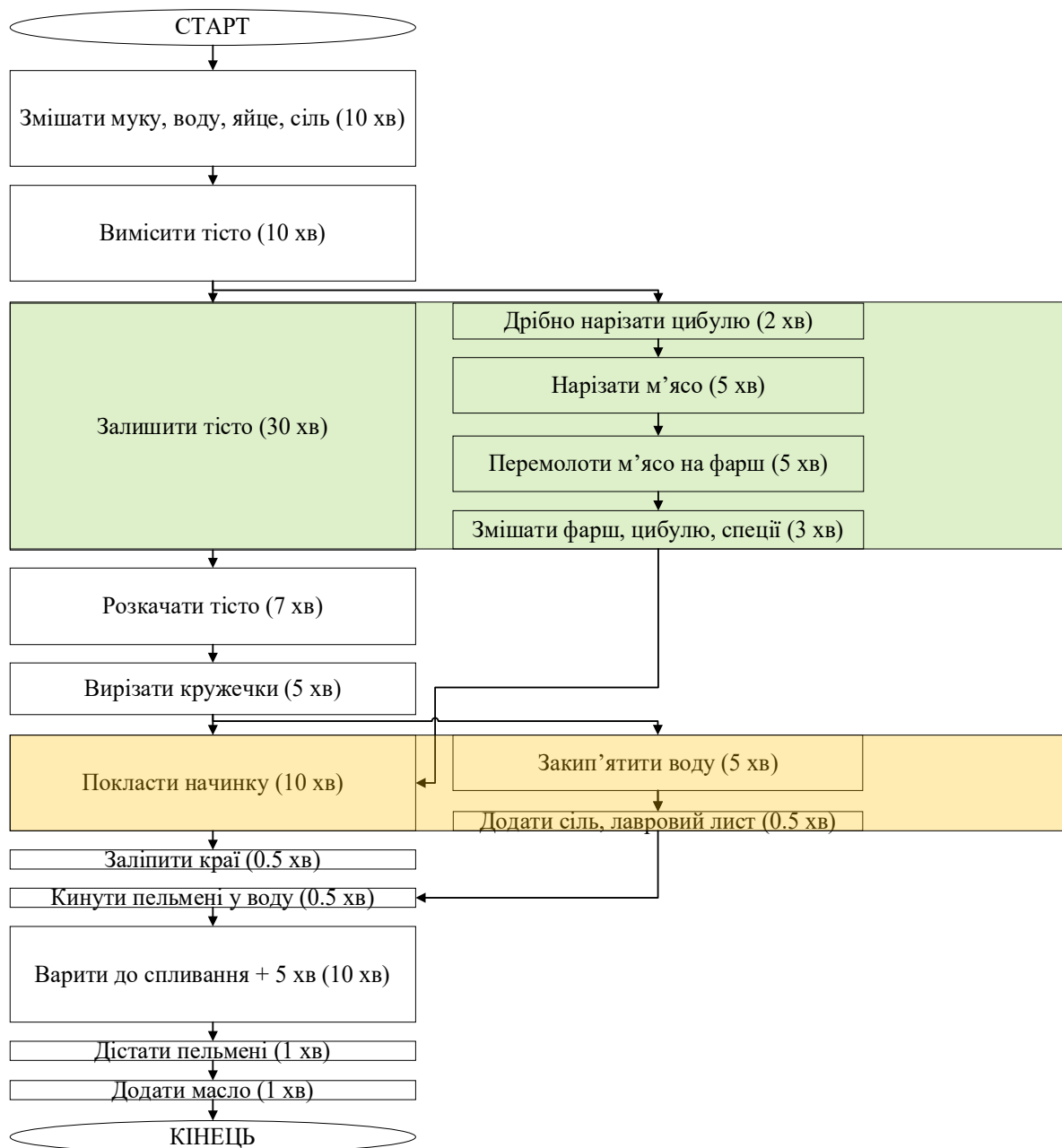


Рисунок 1.4 – Обмежено паралельний алгоритм

Завдання 4:

Знаходимо значення характеристик для усіх алгоритмів.

Таблиця 1 – Послідовний алгоритм

Номер дії	Зміст	Вартість послідовна, хв
1	Змішати муку, воду, яйце, сіль	10
2	Вимісити тісто	10
3	Залишити тісто	30
4	Нарізати м'ясо	5
5	Перемолоти м'ясо на фарш	5
6	Дрібно нарізати цибулю	2
7	Змішати фарш, цибулю, спеції	3
8	Розкачати тісто	7
9	Вирізати кружечки	5
10	Покласти начинку	10
11	Заліпити краї	0,5
12	Закип'ятити воду	5
13	Додати сіль, лавровий лист	0,5
14	Кинути пельмені у киплячу воду	0,5
15	Варити до спливання + 5 хв	10
16	Дістати пельмені	1
17	Додати масло	1

Оцінювання послідовного алгоритму:	
Загальна кількість операції	17
Кількість послідовних операцій	17
Кількість паралельних операцій	
Частка послідовних операцій	1
Частка паралельних операцій	
Сумарна висота паралельної форми	
Ширина паралельної форми	
Загальна вартість роботи	105,5
Вартість послідовних операцій	105,5
Вартість паралельних операцій	
Максимальне можливе прискорення	26,375

Таблиця 2 – Необмежено паралельний алгоритм

Номер дії	Зміст	Вартість ФП1	Вартість ФП2	Вартість ФП3	Вартість ФП4	Вартість рядка
1	Змішати муку, воду, яйце, сіль	10				10
2	Вимісити тісто	10				10
3	Залишити тісто, дрібно нарізати цибулю, нарізати м'ясо, закип'ятити воду	30	2	5	5	30
4	Розкачати тісто, перемолоти м'ясо на фарш, додати сіль і лавровий лист	7		5	0,5	7
5	Вирізати кружечки, змішати фарш та цибулю зі спеціями	5		3		5
6	Покласти начинку	10				10
7	Заліпити краї	0,5				0,5
8	Кинути пельмені у киплячу воду	0,5				0,5
9	Варити до спливання + 5 хв	10				10
10	Дістати пельмені	1				1
11	Додати масло	1				1

Оцінювання паралельного алгоритму:	
Висота паралельної форми	11
Кількість послідовних операцій	8
Кількість паралельних операцій	9
Частка послідовних операцій	47%
Частка паралельних операцій	53%
Ширина паралельної форми	4
Загальна вартість роботи	105,5
Вартість послідовних операцій	43
Вартість паралельних операцій	42
Вартість розпаралеленого алгоритму	85
Прискорення при необ. парал.	1,24
Максимальне можливо прискорення	26,4

Таблиця 3 – Обмежено паралельний алгоритм

Номер дії	Зміст	Вартість ФП1	Вартість ФП2	Вартість рядка
1	Змішати муку, воду, яйце, сіль	10		10
2	Вимісити тісто	10		10
3	Залишити тісто, дрібно нарізати цибулю, нарізати м'ясо, перемолоти м'ясо на фарш, змішати фарш та цибулю зі спеціями	30	15	30
4	Розкачати тісто	7		7
5	Вирізати кружечки	5		5
6	Покласти начинку, закип'ятити воду, додати сіль та лавровий лист	10	5,5	10
7	Заліпити краї	0,5		0,5

8	Кинути пельмені у киплячу воду	0,5		0,5
9	Варити до спливання + 5 хв	10		10
10	Дістати пельмені	1		1
11	Додати масло	1		1

Оцінювання паралельного алгоритму:	
Висота паралельної форми	11
Кількість послідовних операцій	9
Кількість паралельних операцій	8
Частка послідовних операцій	53%
Частка паралельних операцій	47%
Ширина паралельної форми	2
Загальна вартість роботи	105,5
Вартість послідовних операцій	44
Вартість паралельних операцій	41
Вартість розпаралеленого алгоритму	85
Прискорення при необ. парал.	1,24
Максимальне можливо прискорення	52,8

Висновки:

На цій лабораторній роботі ми оцінили ефективність розпаралеленого алгоритму та набули навичок оцінювання ефективності розпаралелювання довільного алгоритму. У моєму варіанті розпаралелювання необмеженим паралелізмом не надало достатньо ефективного приросту, прискоривши процес в 1.24 рази, при максимально можливих 26.4. Але метод обмеженого паралелізму надає більшого прискорення за рахунок вдвічі меншої кількості пристроїв, видаючи те саме прискорення в 1.24 рази.

Контрольні питання:

1. Поясніть закон Амдала в загальному виді.

Закон Амдала визначає максимальне прискорення, яке можна отримати від розпаралелювання програми.

2. Поясніть закон Амдала з точки зору написання програм.

У програмуванні закон Амдала означає, що навіть якщо велика частина коду виконується паралельно, серійна (непаралельна) частина обмежує максимальне прискорення.

3. Що таке внутрішній паралелізм?

Внутрішній паралелізм – це можливість виконання кількох операцій одночасно всередині одного процесора або пристрою (наприклад, конвеєризація в процесорах).

4. Поясніть концепцію необмеженого паралелізму.

Необмежений паралелізм – ідея, що теоретично можливо досягти будь-якого рівня прискорення, якщо є достатньо процесорів і відсутні послідовні обмеження.

5. Скільки пристроїв необхідно для реалізації вашого розпаралеленого алгоритму?

Для реалізації необмеженого паралелізму мені знадобилося чотири пристрої, а для обмеженого – два пристрої.

6. Як оцінюється ефективність розпаралелювання?

Ефективність розпаралелювання оцінюється коефіцієнтом прискорення:

$$E = \frac{Z}{P}, \quad (1.1)$$

де Z – загальна вартість роботи, P – вартість розпаралеленого алгоритму.

7. Що таке ширина паралельної форми?

Ширина паралельної форми – це кількість пристроїв, що використовується для розпаралелювання.

8. Що таке висота паралельної форми?

Висота паралельної форми – це кількість усіх процесів, враховуючи їх розпаралелення.

9. Як обчислюється частка паралельних операцій?

Частка паралельних операцій обчислюється наступним чином:

$$C = \frac{Z}{P}, \quad (1.2)$$

де Z – загальна кількість операцій, P – кількість паралельних операцій.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 2

Виконав

студент групи КН-23-1

Полинько І. М.

Перевірила

доцент кафедри АІС

Істоміна Н. М.

Кременчук 2025

Лабораторна робота №2

Тема: Обчислення характеристик функціональних пристроїв

Мета: набути навичок обчислення характеристик функціональних пристроїв і систем побудованих на них.

Хід роботи:

1. Для системи, що складається з функціональних пристроїв (характеристики ФП наведені у таблиці 2.1), визначити:

- кількість пристроїв;
- вартість роботи кожного ФП;
- вартість роботи системи;
- завантаженість кожного ФП;
- пікову продуктивність системи;
- реальну продуктивність системи;
- зважену завантаженість кожного ФП;
- зважену завантаженість системи;
- максимальну пікову продуктивність у системі;
- прискорення.

2. Увести до системи ще один ФП з максимальними з можливих характеристик. Оцінити, як зміниться прискорення та реальна продуктивність системи.

3. Увести до початкової системи ще один ФП з мінімальними з можливих характеристик. Оцінити як зміниться прискорення та реальна продуктивність системи.

4. До звіту додати лістинги розрахунків для початкової системи та систем з додатковими ФП.

Таблиця 2.1 – Варіанти характеристик функціональних пристроїв

Варіант 16	ФП1	ФП2	ФП3	ФП4	ФП5	ФП6	ФП7	ФП8	ФП9	ФП10
Пікова продуктивність ФП	8000	8000	8000	8000	8000	8000	8000	8000	8000	8000
Реальна продуктивність ФП	6985	6281	6472	7047	7150	5093	4910	5713	5141	6856
Вартість однієї операції, нс	2	2	2	2	2	2	2	2	2	2

Завдання 1:

▲	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Варіант 16	ФП1	ФП2	ФП3	ФП4	ФП5	ФП6	ФП7	ФП8	ФП9	ФП10	Кількість пристроїв	10
2	Пікова	8000	8000	8000	8000	8000	8000	8000	8000	8000	8000	Пікова продуктивність	80000
3	Реальна	6985	6281	6472	7074	7150	5093	4910	5713	5141	6856	Реальна	61675
4	Вартість однієї	2	2	2	2	2	2	2	2	2	2		
5													
6	Вартість роботи	13970	12562	12944	14148	14300	10186	9820	11426	10282	13712	Вартість роботи	123350
7	Завантаженість	87,31%	78,51%	80,90%	88,43%	89,38%	63,66%	61,38%	71,41%	64,26%	85,70%		
8	Зважена	11,33%	10,18%	10,49%	11,47%	11,59%	8,26%	7,96%	9,26%	8,34%	11,12%	Зважена	100,00%
9												Максимальна пікова	8000
10												Прискорення	7,709375

Рисунок 2.1 – Базова система

Завдання 2:

▲	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Варіант 16	ФП1	ФП2	ФП3	ФП4	ФП5	ФП6	ФП7	ФП8	ФП9	ФП10	Кількість пристроїв	10
2	Пікова	8000	8000	8000	8000	8000	8000	8000	8000	8000	16000	Пікова продуктивність	88000
3	Реальна	6985	6281	6472	7074	7150	5093	4910	5713	5141	14006	Реальна	68825
4	Вартість однієї	2	2	2	2	2	2	2	2	2	2		
5													
6	Вартість роботи	13970	12562	12944	14148	14300	10186	9820	11426	10282	28012	Вартість роботи	137650
7	Завантаженість	87,31%	78,51%	80,90%	88,43%	89,38%	63,66%	61,38%	71,41%	64,26%	87,54%		
8	Зважена	10,15%	9,13%	9,40%	10,28%	10,39%	7,40%	7,13%	8,30%	7,47%	20,35%	Зважена	100,00%
9												Максимальна пікова	16000
10												Прискорення	4,3015625

Рисунок 2.2 – Система «Кращий ФП»

Завдання 3:

▲	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Варіант 16	ФП1	ФП2	ФП3	ФП4	ФП5	ФП6	ФП7	ФП8	ФП9	ФП10	Кількість пристроїв	10
2	Пікова	8000	8000	8000	8000	8000	8000	8000	8000	8000	7500	Пікова продуктивність	79500
3	Реальна	6985	6281	6472	7074	7150	5093	4910	5713	5141	5855	Реальна	60674
4	Вартість однієї	2	2	2	2	2	2	2	2	2	2		
5													
6	Вартість роботи	13970	12562	12944	14148	14300	10186	9820	11426	10282	11710	Вартість роботи	121348
7	Завантаженість	87,31%	78,51%	80,90%	88,43%	89,38%	63,66%	61,38%	71,41%	64,26%	78,07%		
8	Зважена	11,51%	10,35%	10,67%	11,66%	11,78%	8,39%	8,09%	9,42%	8,47%	9,65%	Зважена	100,00%
9												Максимальна пікова	8000
10												Прискорення	7,58425

Рисунок 2.3 – Система «Гірший ФП»

Завдання 4:

12	<i>Зведена</i>	Base	ФП+	ФП-
13	Прискорення	7,709375	4,301563	7,58425
14	Вартість роботи	123350	137650	121348
15	Реальна	61675	68825	60674

Рисунок 2.4 – Лістинг розрахунків

Висновки:

На цій лабораторній роботі ми обчислювали характеристики функціональних пристроїв, набути навичок обчислення характеристик функціональних пристроїв і систем побудованих на них. У моєму варіанті покращення функціонального пристрою призвело до незначного збільшення вартості роботи та великого зменшення прискорення, а зменшення функціонального пристрою призвело до мінімальних змін параметрів у порівнянні з базовою системою.

Контрольні питання:**1. Що таке вартість роботи?**

Вартість роботи – це кількість елементарних операцій, необхідних для виконання певної задачі або алгоритму.

2. Як оцінюється вартість окремої логічної операції?

Вартість логічної операції визначається за кількістю тактів або умовних одиниць, які потрібні для її виконання. У спрощеному аналізі приймається, що базова логічна операція (наприклад, AND, OR, NOT) має вартість 1 умовну одиницю.

3. Як оцінюється вартість виконання вибраного алгоритму або програмного коду?

Вартість виконання алгоритму визначається як сума вартостей усіх операцій, що виконуються, з урахуванням кількості повторень (циклів, рекурсій).

$$C = \sum_{i=1}^n c_i \cdot k_i, \quad (2.1)$$

де c_i – вартість i -тої операції, k_i – кількість її виконань.

4. Що таке функціональний пристрій?

Функціональний пристрій – це апаратний або логічний модуль, здатний виконувати певний набір обчислювальних операцій, таких як додавання, множення, передача даних тощо.

5. Приклади системи, що складається з декількох функціональних пристроїв:

- Центральний процесор (CPU), графічний процесор (GPU) та блок прискорення нейронних обчислень (TPU).
- Багатоядерний процесор, де кожне ядро – окремий функціональний пристрій.
- Вбудована система з арифметичним блоком, блоком пам'яті та блоком введення/виведення.

6. Що таке простий функціональний пристрій?

Простий функціональний пристрій – це пристрій, що виконує лише одну типову операцію за один такт, наприклад, додавання двох чисел. Його архітектура не дозволяє виконання складних або комбінованих операцій.

7. Як обчислюється реальна та пікова продуктивність функціонального пристрою?

- Пікова продуктивність – це максимально можлива кількість операцій, яку пристрій здатен виконати за одиницю часу:

$$P_{peak} = n \cdot f, \quad (2.2)$$

де n – кількість операцій за такт, f – тактова частота.

- Реальна продуктивність – фактична кількість операцій, виконаних за одиницю часу при заданому навантаженні:

$$P_{peak} = \frac{N}{T}, \quad (2.3)$$

де N – кількість виконаних операцій, T – час виконання.

8. Як обчислюється завантаженість функціонального пристрою?

Завантаженість показує ступінь використання пристрою:

$$U = \frac{P_{real}}{P_{peak}} \cdot 100\%, \quad (2.4)$$

9. Сформулюйте базовий закон Амдала.

Закон Амдала визначає теоретичне обмеження прискорення паралельної обчислювальної системи:

$$S = \frac{1}{(1-\alpha) + \frac{\alpha}{p}}, \quad (2.5)$$

де α – частка алгоритму, що може бути паралелізована, p – кількість паралельних пристроїв.

10. Як обчислюється завантаженість системи з декількох функціональних пристроїв?

$$U_{sys} = \frac{\sum P_{real,i}}{\sum P_{peak,i}} \cdot 100\%, \quad (2.6)$$

де $P_{real,i}$ – реальна продуктивність i -го пристрою, $P_{peak,i}$ – пікова продуктивність i -го пристрою.

11. Як обчислюється реальна продуктивність системи з декількох функціональних пристроїв?

$$P_{real,sys} = \sum P_{real,i}, \quad (2.7)$$

12. Як обчислюється пікова продуктивність системи з декількох функціональних пристроїв?

$$P_{peak,sys} = \sum P_{peak,i}, \quad (2.8)$$

13. Як обчислюється прискорення системи з декількох функціональних пристроїв?

$$S = \frac{T_1}{T_p}, \quad (2.9)$$

де T_1 – час виконання задачі на одному пристрої, T_p – час виконання задачі на паралельній системі.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 3

Виконав

студент групи КН-23-1

Полинько І. М.

Перевірила

доцент кафедри АІС

Істоміна Н. М.

Кременчук 2025

Лабораторна робота №3

Тема: Обробка даних на конвеєрному пристрої

Мета: набути навичок оцінювання обробки даних на конвеєрних пристроях.

Хід роботи:

Обробка даних на конвеєрному пристрої складається з 10 стадій. Кількість тактів, необхідних для проходження кожної стадії, представлена у таблиці 3.1. Ініціалізація конвеєра потребує тактів, а тривалість одного такту складає 5 нс.

Під час виконання лабораторної роботи потрібно розв'язати такі завдання:

1. Обчислити кількість тактів, необхідних для виконання 1000 операцій обробки даних за умови, що пристрій працює у послідовному режимі.
2. Обчислити кількість тактів, необхідних для виконання 1000 операцій обробки даних за умови, що пристрій працює у конвеєрному режимі.
3. Визначити найменшу кількість операцій, при виконанні яких у конвеєрному режимі досягається прискорення не менше за 90 % від граничного прискорення.
4. Підрахувати пікову продуктивність системи.

Таблиця 3.1 – Тривалості у тактах для конвеєрного пристрою

№ варіанта	Номер стадії									
	№1	№2	№3	№4	№5	№6	№7	№8	№9	№10
	Тривалість у тактах									
16	3	9	2	5	10	7	6	2	7	1

Завдання 1-4:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Кількість елементів	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	усього стадій
2	Вартість 1 такту, нс	5	5	5	5	5	5	5	5	5	5	10
3												
4	Показник	Стадія 1	Стадія 2	Стадія 3	Стадія 4	Стадія 5	Стадія 6	Стадія 7	Стадія 8	Стадія 9	Стадія	усього тактів
5	Кількість тактів	3	9	2	5	10	7	6	2	7	1	52
6	Кількість тактів при	3000	9000	2000	5000	10000	7000	6000	2000	7000	1000	52000
7	Кількість тактів при	1002	1008	1001	1004	1009	1006	1005	1001	1006	1000	10042
8												
9												
10	Пікова	0,02										
11	Відносна пікова	0,1										
12	Граничне	5,2										
13	90 % від граничного	4,68										
14	мінімальна кількість	6										
15	Прискорення	5,178										

Рисунок 3.1 – Конвеєр

Висновки:

На цій лабораторній роботі ми оброблювали дані на конвеєрному пристрої, набули навичок оцінювання обробки даних на конвеєрних пристроях. Створили модель даних конвеєрної обробки за власним варіантом, та обчислили кількість тактів, необхідних для виконання 1000 операцій обробки даних, за умови, що пристрій працює у послідовному режимі, а також за умови що пристрій працює у конвеєрному режимі. Визначили найменшу кількість операцій, при виконанні яких у конвеєрному режимі досягається прискорення не менше за 90 % від граничного прискорення та підраховували пікову продуктивність системи.

Контрольні питання:**1. Загальна кількість тактів при послідовній обробці даних**

Визначається як добуток кількості оброблюваних елементів на кількість етапів обробки кожного елемента. Кожен елемент проходить усі етапи послідовно, без перекриття з іншими.

2. Загальна кількість тактів при обробці даних на паралельних процесорах

Визначається як кількість тактів, необхідна для обробки всіх елементів, розподілених між процесорами. Кожен процесор обробляє частину задач незалежно, тому загальний час визначається за найзавантаженішим процесором.

3. Загальна кількість тактів при конвеєрній обробці

Визначається як сума часу розгону конвеєра (що дорівнює кількості стадій конвеєра) та часу обробки решти елементів (по одному такту на кожен наступний елемент).

4. Відносна продуктивність при конвеєрній обробці

Характеризує співвідношення між часом виконання задачі на послідовному пристрої та на конвеєрі. Дає оцінку ефективності використання конвеєра порівняно з послідовною обробкою.

5. Продуктивність конвеєрної обробки

Визначається як середня кількість результатів, що видаються конвеєром за один такт. При достатньо великій кількості даних продуктивність наближається до одиниці.

6. Пікова продуктивність

Максимально можлива продуктивність, що досягається при повному завантаженні конвеєра або системи. Характеризує теоретичну межу системи без урахування затримок.

7. Прискорення

Характеризує, у скільки разів зменшується час виконання задачі при використанні конвеєрної або паралельної обробки. Визначається як відношення часу виконання на послідовному пристрої до часу на вдосконаленому.

8. Граничне прискорення

Визначається як максимально можливе прискорення при ідеальному розподілі задач і нескінченному обсязі вхідних даних. Залежить від архітектури системи та кількості функціональних блоків.

9. Вплив характеристик конвеєра на продуктивність

На продуктивність впливають такі параметри: кількість стадій, час виконання окремих стадій, балансованість навантаження між стадіями, наявність конфліктів, об'єм оброблюваних даних. При зростанні кількості оброблюваних елементів ефективність конвеєра зростає.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 4

Виконав

студент групи КН-23-1

Полинько І. М.

Перевірила

доцент кафедри АІС

Істоміна Н. М.

Кременчук 2025

Лабораторна робота №4

Тема: Розпаралелювання на основі ациклічних графів

Мета: набути навичок розпаралелювання алгоритмів, що ґрунтується на побудові ациклічних графів.

Хід роботи:

Під час лабораторної роботи необхідно виконати такі дії:

1. Знайти еквівалентний вираз для наведеного у табл. 4.1.
2. Скласти послідовний ОАГ арифметичного виразу.
3. Скласти паралельний ОАГ за умови необмеженого паралелізму.
4. Скласти паралельний ОАГ за умови застосування трьох ФП.

Таблиця 4.1 – Рівняння для створення ациклічних графів

№ пор.	Рівняння
16	$((((x\{6\}/x\{2\})-(x\{3\}*x\{4\}))+(((x\{5\}/x\{6\})-(x\{7\}*x\{8\}))*((x\{9\}/x\{10\})+((x\{1\}/x\{2\})+(x\{3\}*x\{4\}))))))$

Завдання 1-4:

На рисунку 4.1 наведено еквівалентний вираз та послідовний ОАГ арифметичного виразу.

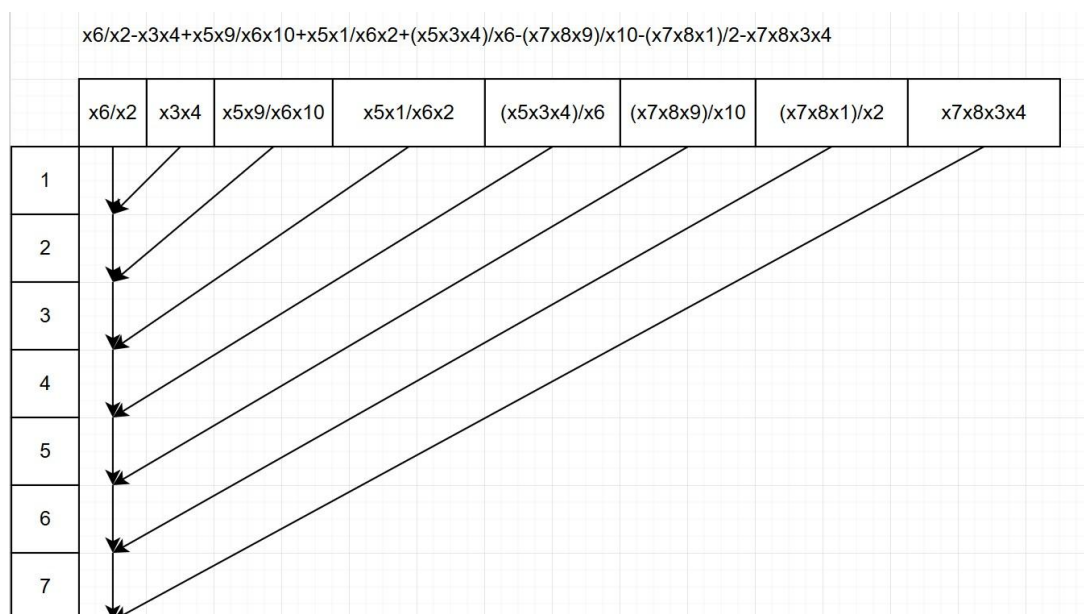


Рисунок 4.1 – Послідовний ОАГ арифметичного виразу

На рисунку 4.2 наведено паралельний ОАГ за умови необмеженого паралелізму.

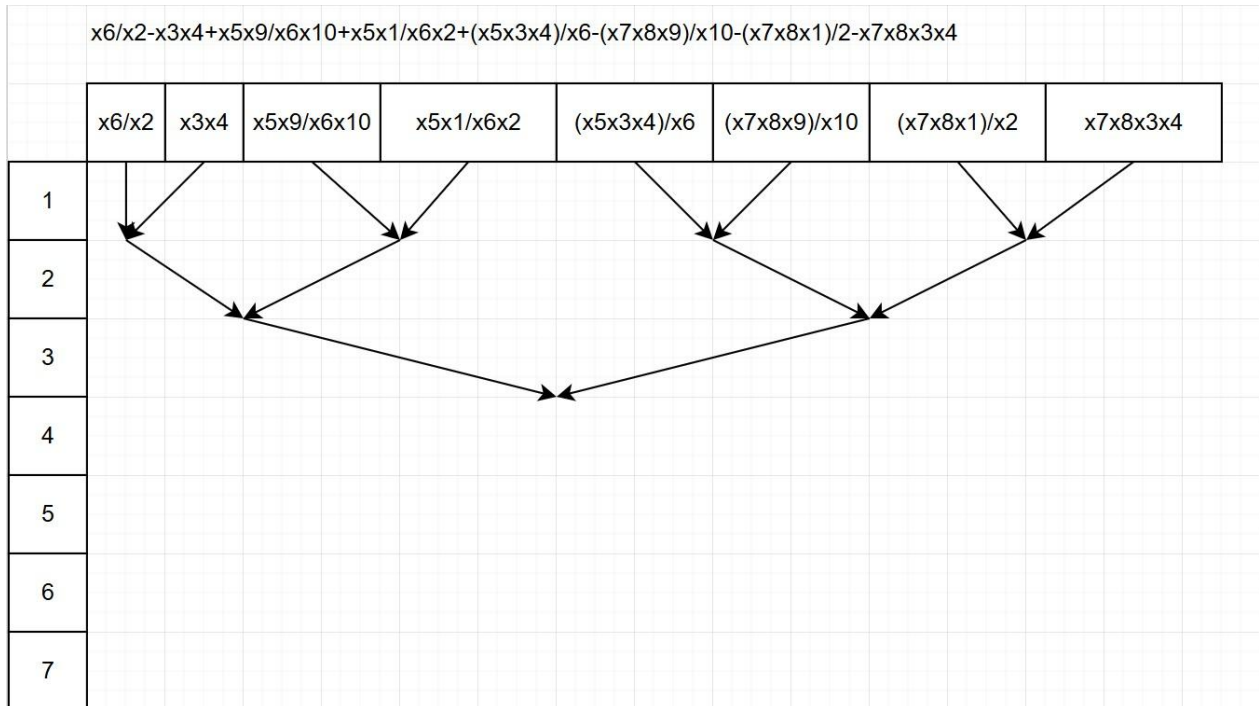


Рисунок 4.2 – Паралельний ОАГ за умови необмеженого паралелізму

На рисунку 4.3 наведено паралельний ОАГ за умови застосування трьох ФП.

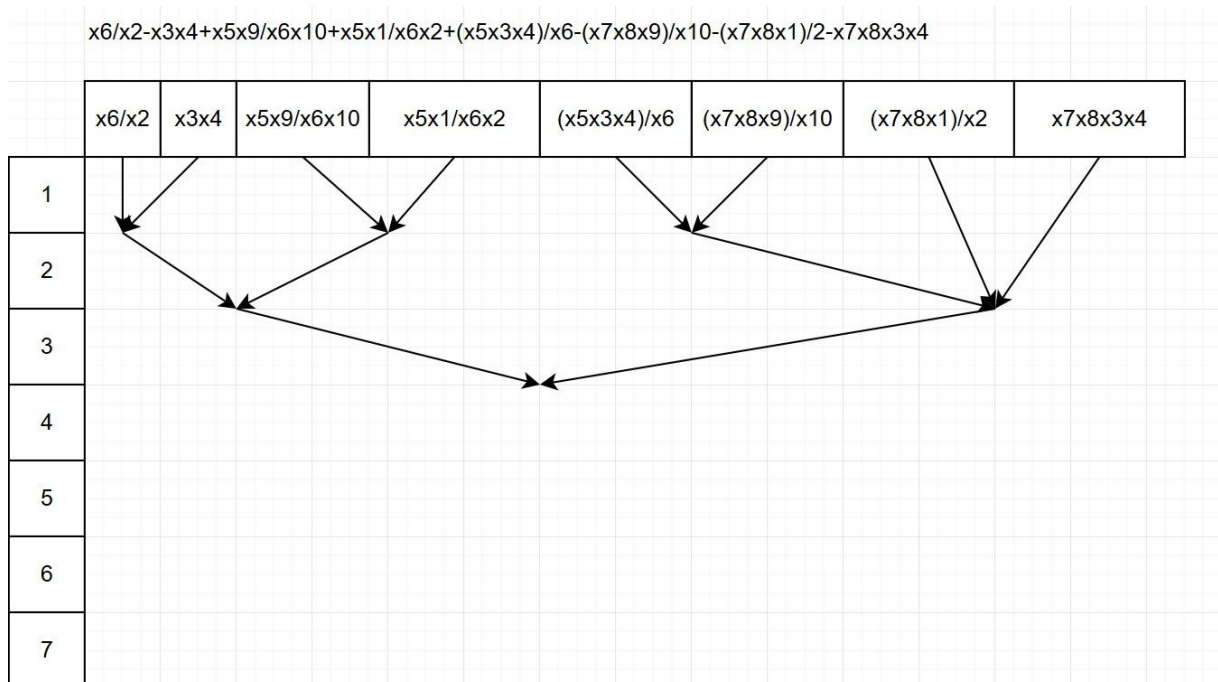


Рисунок 4.3 – Паралельний ОАГ за умови застосування трьох ФП

Висновки:

На цій лабораторній роботі ми проводили розпаралелювання на основі ациклічних графів, набули навичок розпаралелювання алгоритмів, що ґрунтуються на побудові ациклічних графів. Навели еквівалентний вираз, створили три графи, а саме: послідовний ОАГ арифметичного виразу, паралельний ОАГ за умови необмеженого паралелізму та паралельний ОАГ за умови застосування трьох ФП. Послідовний граф є найдовший, а паралельні схожі між собою, навіть з обмеженням в три ФП.

Контрольні питання:

1. Що таке альтернований вираз?

Альтернований вираз – це вираз, у якому операції чергуються за певним законом. У контексті алгебри або комп'ютерних обчислень, це часто означає чергування знаків (наприклад: $a - b + c - d + e$). Альтернування також може стосуватись типу операцій (наприклад, чергування додавання і множення).

2. Поясніть поняття адитивних і мультиплікативних виразів.

- Адитивні вирази – це вирази, у яких основними операціями є додавання та віднімання (наприклад: $a + b - c + d$).
- Мультиплікативні вирази – це вирази, побудовані переважно з множенням або діленням (наприклад: $a * b / c * d$).

3. Що таке схема Горнера?

Схема Горнера – це спосіб ефективного обчислення значення полінома.
Поліном

$$P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

представляється у вигляді:

$$P(x) = (\dots((a_nx + a_{n-1})x + a_{n-2})x + \dots) + a_0.$$

Це дозволяє зменшити кількість множень і додавань, зокрема до n множень і n додавань.

4. Поясніть закони асоціативності, комутативності та дистрибутивності.

- Асоціативність: Порядок об'єднання операндів не змінює результат.

Наприклад: $(a + b) + c = a + (b + c)$

- Комутативність: Порядок операндів не впливає на результат.

Наприклад: $a + b = b + a$

- Дистрибутивність: Одна операція може бути «розподілена» через іншу.

Наприклад: $a * (b + c) = a * b + a * c$

5. Що таке орієнтований граф?

Орієнтований граф (діграф) – це граф, у якому кожне ребро має напрямок, тобто йде від однієї вершини до іншої. Наприклад, якщо є ребро з А до В, то це не означає, що є шлях з В до А.

6. Що таке ациклічний граф?

Ациклічний граф – це граф, який не містить жодного циклу, тобто немає шляху, що починається і закінчується в одній і тій самій вершині. Якщо він ще й орієнтований, то його називають орієнтованим ациклічним графом (DAG).

7. Поясніть алгоритм Винограду.

Алгоритм Винограду – це оптимізований алгоритм множення матриць, який зменшує кількість множень за рахунок попередніх обчислень часткових добутків (розкладки по рядках і стовпцях). Він ефективніший за класичний алгоритм при великих обсягах даних, але складніший у реалізації.

8. Поясніть лему Брента.

Лема Брента оцінює, як можна розпаралелити обчислення у графі залежностей. Якщо є обчислювальний граф з роботою W (загальна кількість операцій) і довжиною критичного шляху D (глибина), то при P процесорах час виконання можна обмежити як:

$$T \leq W/P + D.$$

Це означає, що прискорення обмежене як кількістю процесорів, так і внутрішніми залежностями.

9. Побудуйте орієнтований ациклічний граф для схеми Горнера.

Для полінома $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, схема Горнера:

$$(((a_3x + a_2)x + a_1)x + a_0)$$

Орієнтований граф:

$$a_3 \rightarrow *x \rightarrow +a_2 \rightarrow *x \rightarrow +a_1 \rightarrow *x \rightarrow +a_0$$

10. Побудуйте орієнтований ациклічний граф алгоритму логарифмічного додавання.

Логарифмічне додавання – це метод паралельного додавання n чисел, де суми обчислюються попарно:

Наприклад, для 8 чисел:

Рівень 0: a b c d e f g h

Рівень 1: a+b c+d e+f g+h

Рівень 2: (a+b)+(c+d) (e+f)+(g+h)

Рівень 3: всі суми разом

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 5

Виконав

студент групи КН-23-1

Полинько І. М.

Перевірила

доцент кафедри АІС

Істоміна Н. М.

Кременчук 2025

Лабораторна робота № 5

Тема: Дослідження ієрархічної організації пам'яті

Мета: набути навичок оцінювання взаємодії з пам'яттю комп'ютера.

Хід роботи:

Отже у межах лабораторної роботи потрібно виконати такі дії:

1. Створити програмний застосунок, що проводить перемноження двох довільних матриць (із глибиною та шириною не менше, ніж 250 елементів) згідно з варіантом 1 перебору циклу.
2. До звіту підготувати таблицю для занесення експериментальних даних (табл. 5.1).
3. Провести 5 дослідів з першим варіантом застосунку.
4. Змінити застосунок так, щоб перебір циклу здійснювався згідно з варіантом 2.
5. Провести 5 дослідів із другим варіантом застосунку.
6. Оцінити, наскільки порядок доступу до даних впливає на швидкість роботи застосунку.
7. У звіті навести характеристики свого процесора та оперативної пам'яті.
8. До звіту додати код двох варіантів застосунків.

Завдання:

Створимо програмний застосунок до завдання, що проводить перемноження двох довільних матриць (із глибиною та шириною не менше, ніж 250 елементів) згідно з варіантом 1 перебору циклу.

```
using System;
using System.Diagnostics;

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Default;

        int L = 250; // Розмір матриці

        // Ініціалізація матриць
        double[,] arrayA = GenerateRandom(L, L);
        double[,] arrayB = GenerateRandom(L, L);
```

```

double[,] arrayR = new double[L, L];

// Вимірювання часу
Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

Variant1(arrayA, arrayB, arrayR, L);

stopwatch.Stop();

Console.WriteLine($"Час виконання: {stopwatch.ElapsedMilliseconds} мс");
}

static double[,] GenerateRandom(int rows, int cols)
{
    Random rand = new Random();
    double[,] matrix = new double[rows, cols];


    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            matrix[i, j] = rand.NextDouble() * 100;

    return matrix;
}

static void Variant1(double[,] arrayA, double[,] arrayB, double[,] arrayD,
int L)
{
    for (int i = 0; i < L; i++)
        for (int k = 0; k < L; k++)
            for (int j = 0; j < L; j++)
                arrayD[i, j] += arrayA[i, k] * arrayB[k, j];
}
}


```

На рисунках 5.1 – 5.5 наведено 5 дослідів з першим варіантом застосунку.



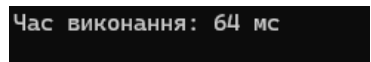
Час виконання: 68 мс

Рисунок 5.1 – Перший дослід з першим варіантом застосунку



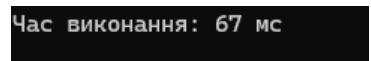
Час виконання: 67 мс

Рисунок 5.2 – Другий дослід з першим варіантом застосунку



Час виконання: 64 мс

Рисунок 5.3 – Третій дослід з першим варіантом застосунку



Час виконання: 67 мс

Рисунок 5.4 – Четвертий дослід з першим варіантом застосунку

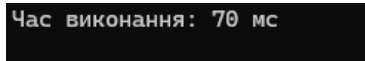


Рисунок 5.5 – П'ятий дослід з першим варіантом застосунок

Створимо програмний застосунок до завдання, що проводить перемноження двох довільних матриць (із глибиною та шириною не менше, ніж 250 елементів) згідно з варіантом 2 перебору циклу.

```
using System;
using System.Diagnostics;

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Default;

        int L = 250; // Розмір матриці

        // Ініціалізація матриць
        double[,] arrayA = GenerateRandom(L, L);
        double[,] arrayB = GenerateRandom(L, L);
        double[,] arrayR = new double[L, L];

        // Вимірювання часу
        Stopwatch stopwatch = new Stopwatch();

        stopwatch.Start();

        Variant2(arrayA, arrayB, arrayR, L);

        stopwatch.Stop();

        Console.WriteLine($"Час виконання: {stopwatch.ElapsedMilliseconds} мс");
    }

    static double[,] GenerateRandom(int rows, int cols)
    {
        Random rand = new Random();
        double[,] matrix = new double[rows, cols];

        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                matrix[i, j] = rand.NextDouble() * 100;

        return matrix;
    }

    static void Variant2(double[,] arrayA, double[,] arrayB, double[,] arrayD,
int L)
    {
        for (int j = 0; j < L; j++)
            for (int k = 0; k < L; k++)
                for (int i = 0; i < L; i++)
                    arrayD[i, j] += arrayA[i, k] * arrayB[k, j];
    }
}
```


На рисунках 5.6 – 5.10 наведено 5 дослідів з другим варіантом застосунку.

Час виконання: 69 мс

Рисунок 5.6 – Перший дослід з другим варіантом застосунку

Час виконання: 72 мс

Рисунок 5.7 – Другий дослід з другим варіантом застосунку

Час виконання: 68 мс

Рисунок 5.8 – Третій дослід з другим варіантом застосунку

Час виконання: 70 мс

Рисунок 5.9 – Четвертий дослід з другим варіантом застосунку

Час виконання: 72 мс

Рисунок 5.10 – П'ятий дослід з другим варіантом застосунку

Таблиця 5.1 – Отримані експериментальні дані

Перебір циклу	Витрачений час, мс					Середнє значення
	номер досліду					
	№ 1	№ 2	№ 3	№ 4	№ 5	
Варіант 1	68	67	64	67	70	67,2
Варіант 2	69	72	68	70	72	70,2

Характеристики мого процесора та оперативної пам'яті:

Процесор: 12th Gen Intel(R) Core(TM) i5-12450H 2.00 GHz

Оперативна пам'ять: 16 GB DDR4

Висновки:

На цій лабораторній роботі ми досліджували ієрархічну організацію пам'яті, набули навичок оцінювання взаємодії з пам'яттю комп'ютера. Створили дві програми мовою C# з двома варіантами множення матриць. Вирахували швидкість обчислення процесором операції, запустивши 5 операцій і обчисливши середнє значення. Перший варіант спрацював у більшості досліджень – швидше за другий.

Контрольні питання:**1. Принципи організації ієрархічної пам'яті ЕОМ**

Ієрархічна пам'ять комп'ютера складається з кількох рівнів, що відрізняються швидкістю доступу, обсягом і вартістю:

- Регістр процесора (найшвидший, найменший): безпосередньо вбудований у процесор, використовується для збереження операндів обчислень.
- Кеш-пам'ять (L1, L2, L3): швидкий проміжний буфер між процесором і оперативною пам'яттю.
- Оперативна пам'ять (RAM): основна пам'ять для зберігання програм і даних під час роботи.
- Накопичувачі (SSD/HDD): зберігання даних на тривалий час.
- Мережеве сховище та хмарна пам'ять (найповільніші): зберігають великі обсяги даних.

2. Поняття «кеш»

Кеш – це швидка проміжна пам'ять, яка зберігає найбільш часто використовувані дані та інструкції, щоб прискорити доступ до них. Існує три основні рівні кешу:

- L1 (найшвидший, найменший): безпосередньо у ядрі процесора.
- L2 (швидкий, більший): також у процесорі, але спільний для кількох ядер.

– L3 (найповільніший серед кешів, найбільший): може бути спільним для всього процесора.

3. Як зберігаються елементи матриці в оперативній пам'яті?

Елементи матриці зберігаються в оперативній пам'яті у вигляді послідовного блоку пам'яті:

– Для двовимірної матриці (як у нашому випадку) зберігається послідовність рядків (рядкова організація).

– Наприклад, для матриці 3×3 :

$A[0,0], A[0,1], A[0,2], A[1,0], A[1,1], A[1,2], A[2,0], A[2,1], A[2,2]$

– Це дозволяє швидко звертатися до сусідніх елементів у пам'яті.

4. Як змінити код, щоб розмірність матриць задавав користувач?

```
Console.WriteLine("Введіть розмір матриці (L): ");
```

```
int L = int.Parse(Console.ReadLine());
```

5. Як організувати генерацію елементів матриці за допомогою функції «random»?

```
static double[,] GenerateRandom(int rows, int cols)
```

```
{
```

```
    Random rand = new Random();
```

```
    double[,] matrix = new double[rows, cols];
```

```
    for (int i = 0; i < rows; i++)
```

```
        for (int j = 0; j < cols; j++)
```

```
            matrix[i, j] = rand.NextDouble() * 100; // Випадкові значення від 0 до 100
```

```
    return matrix;
```

```
}
```

6. Як викликають і зупиняють роботу класу «Stopwatch»?

```
stopwatch.Start(); // Початок вимірювання часу
```

```
stopwatch.Stop(); // Завершення вимірювання часу
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра автоматизації та інформаційних систем

Навчальна дисципліна
«ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 6

Виконав

студент групи КН-23-1

Полинько І. М.

Перевірила

доцент кафедри АІС

Істоміна Н. М.

Кременчук 2025

Лабораторна робота № 6

Тема: Використання класів та параметричних потоків

Мета: набути навичок розпаралелювання програмного коду на основні використання потоків і класів мовою C#.

Хід роботи:

Під час лабораторної роботи необхідно виконати такі дії:

1. Створіть консольний застосунок з кодом згідно з прикладом. У звіті наведіть власний код та «прінтскрін» роботи програми.

2. Змініть код так: головний потік очікує натиснення клавіші для завершення; усі потоки працюють із затримкою – 400 мс; ім'я потоку стало ST; ім'я делегата – st; слово Second писалося у такому самому рядку, що й слово Primary; на кожні 2 Primary приходилося одне Second. У звіті наведіть власний код і «прінтскрін» роботи програми з внесеними змінами.

3. Визначте рядки, які містять: створення нового класу; визначення змінних нового класу; створення вторинного потоку; створення методу вторинного потоку.

Завдання 1:

Створимо консольний застосунок з кодом згідно з прикладом.

```
using System;
using System.Threading;

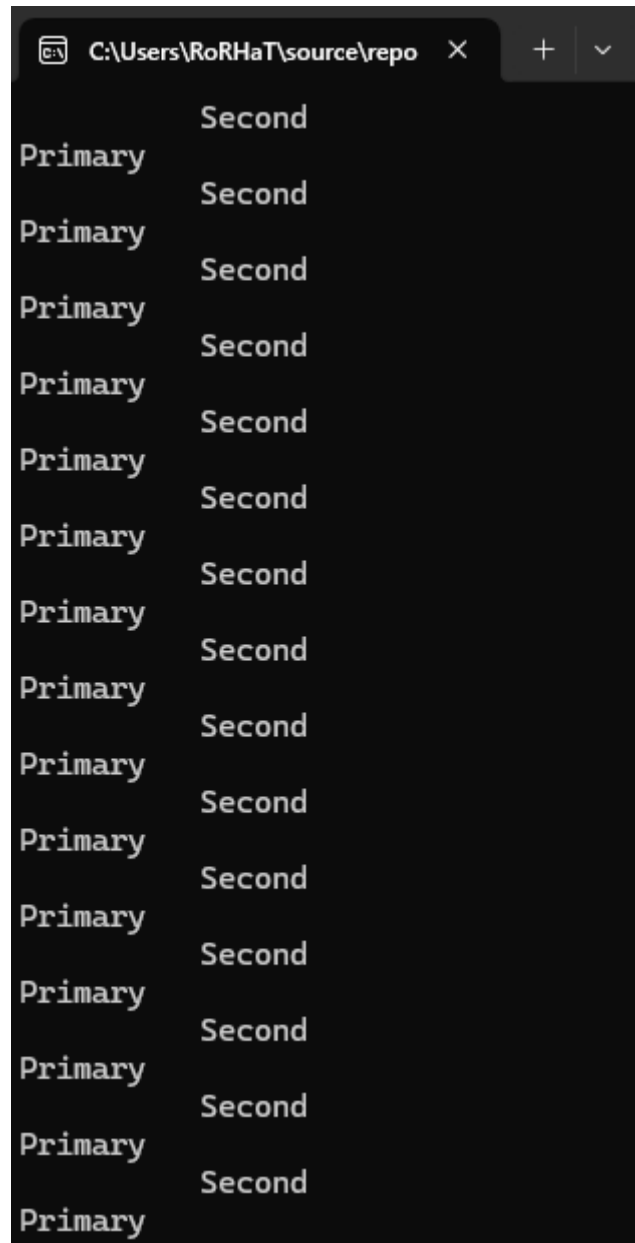
namespace LB6PROPolynkoKN23
{
    class Program
    {
        static void Sec()
        {
            while (true)
            {
                Console.WriteLine(new string(' ', 10) + "Second");
                Thread.Sleep(200);
            }
        }

        static void Main()
        {
            ThreadStart sec = new ThreadStart(Sec);
            Thread thread = new Thread(sec);
            thread.Start();

            while (true)
```

```
{  
    Console.WriteLine("Primary");  
    Thread.Sleep(200);  
}  
}  
}
```

На рисунку 6.1 – наведена робота консольного застосунку.



```
C:\Users\RoRHaT\source\repo X + v  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second  
Primary Second
```

Рисунок 6.1 – Робота потоків

Завдання 2:

Створимо програмний застосунок до завдання, що проводить перемноження двох довільних матриць (із глибиною та шириною не менше, ніж 250 елементів) згідно з варіантом 2 перебору циклу.

```
using System;
using System.Threading;

namespace LB6PROPolynkoKN23
{
    class Program
    {
        static bool isRunning = true; // Прапорець для завершення потоку

        static void Sec()
        {
            while (isRunning)
            {
                Thread.Sleep(800); // 800 мс, бо дві "Primary" по 400 мс = 800 мс
                Console.Write(" Second");
            }
        }

        static void Main()
        {
            ThreadStart st = new ThreadStart(Sec); // Ім'я делегата - st
            Thread ST = new Thread(st); // Ім'я потоку - ST
            ST.Start();

            int primaryCount = 0;

            while (isRunning)
            {
                Console.Write("Primary");
                primaryCount++;

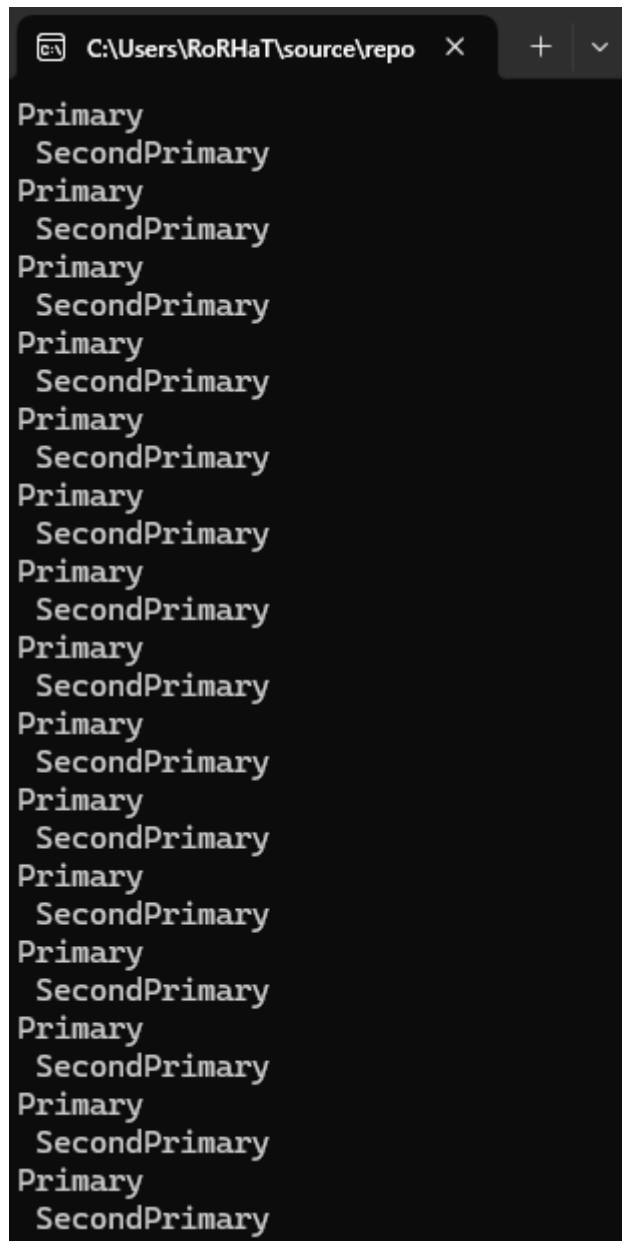
                // Виводимо "Second" після кожних двох "Primary" у тому ж рядку
                if (primaryCount % 1 == 0)
                {
                    Console.WriteLine(); // Перехід на новий рядок після пари
                }

                Thread.Sleep(400);

                // Завершення програми при натисненні клавіші
                if (Console.KeyAvailable)
                {
                    Console.ReadKey();
                    isRunning = false;
                }
            }

            // Чекаємо завершення потоку "ST"
            ST.Join();
            Console.WriteLine("\nЗавершення програми...");
        }
    }
}
```

На рисунку 6.2 – наведена робота зміненого коду.



```
C:\Users\RoRHaT\source\repo
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
Primary
SecondPrimary
```

Рисунок 6.2 – Робота потоків у розширеному кодї

Завдання 3:

1. Створення нового класу:

```
class Program
```

2. Визначення змінних нового класу:

```
static bool isRunning = true; // Прапорець для завершення потоку
```

3. Створення вторинного потоку:

```
ThreadStart st = new ThreadStart(Sec); // Ім'я делегата - st
```



```
Thread ST = new Thread(st); // Ім'я потоку - ST
ST.Start();
```

4. Створення методу вторинного потоку:

```
static void Sec()
```

Висновки:

На цій лабораторній роботі ми використовували класи та параметричні потоки, набули навичок розпаралелювання програмного коду на основі використання потоків і класів мовою C#. Створили два консольних застосунки та провели перевірку роботи потоків. Зазвичай, основний потік працює по чергово з вторим потоком, але інколи основний потік спрацьовує частіше.

Контрольні питання:

1. Що таке потік?

Потік – це окрема послідовність команд, що виконується незалежно від інших потоків у межах одного процесу. У C# потоки дозволяють виконувати код паралельно.

2. Що таке делегат?

Делегат – це тип, що представляє метод із певною сигнатурою. Він дозволяє зберігати посилання на метод і викликати його динамічно.

3. Що таке метод?

Метод – це блок коду з іменем, який виконує певні інструкції та може приймати параметри та повертати значення.

4. Що таке функція?

Функція – це метод, який обов'язково повертає значення. У C# функція є різновидом методу, який має тип повернення, відмінний від void.

5. Які існують стани потоків?

Основні стани потоків у C#:

- Unstarted – потік створено, але ще не запущено.
- Running – потік виконується.

- `WaitSleepJoin` – потік очікує або спить.
- `Stopped` – потік завершив роботу.
- `Aborted` – потік примусово завершений (застаріло).

6. Що таке асиметричні потоки?

Асиметричні потоки – це потоки, де один потік є основним і керує іншими вторинними потоками, які виконують підзадачі.

7. Що таке симетричні потоки?

Симетричні потоки – це потоки, які працюють незалежно та мають однаковий пріоритет і можливості.

8. Що таке гонки?

Гонки (race condition) – це проблема, коли кілька потоків одночасно змінюють одні й ті ж дані, що призводить до непередбачуваних результатів.

9. Поясніть, як підключається бібліотека потоків.

У C# бібліотека потоків (*System.Threading*) підключається директивою *using System.Threading;* на початку файлу.

10. Поясніть, як задається метод виклику делегата.

Метод виклику делегата задається шляхом створення об'єкта делегата з посиланням на метод із відповідною сигнатурою:

```
ThreadStart myDelegate = new ThreadStart(MyMethod);
```

11. Поясніть принципи створення вторинних потоків.

Вторинний потік створюється за допомогою класу `Thread`, де через делегат визначається метод для виконання:

```
Thread myThread = new Thread(MyMethod);  
myThread.Start();
```

12. Як призупинити виконання потоку?

Виконання потоку можна призупинити за допомогою методу *Thread.Sleep(milliseconds);*, що тимчасово зупиняє потік на вказаний час.