Практична робота №1

Тема: Моделі обчислювальних процесів.

Мета: Вивчення моделей обчислювальних процесів

Теоретичні відомості

Модель - це фізична або інформаційна система, що представляє собою об'єкт дослідження адекватно цілям дослідження.

Фізичні моделі утворюються із сукупності матеріальних об'єктів. Для їх побудови використовуються різні властивості, причому природа застосовуваних в моделі матеріальних елементів не обов'язково та ж, що і в досліджуваному об'єкті. Прикладом фізичної моделі є макет.

Абстрактна модель - це опис об'єкта дослідження на деякій мові. Абстрактність моделі виявляється в тому, що її компонентами є поняття, а не фізичні елементи, (наприклад: словесні описи, креслення, схеми, графіки, таблиці, програми, алгоритми, математичні описи).

Необхідна умова для переходу від дослідження об'єкта до дослідження моделі та подальшого перенесення його результатів на об'єкт дослідження - вимога адекватності моделі та об'єкта. Адекватність передбачає відтворення моделлю з необхідною повнотою всіх властивостей об'єкта, істотних для цілей даного дослідження.

Трудомісткість алгоритмів - кількість обчислювальної роботи необхідної для реалізації алгоритму.

Складність алгоритму - мінімальна кількість інформації, необхідної для його опису. Зазвичай в практиці складність алгоритму визначається довжиною запису алгоритму в термінах певної алгоритмічної системи. Наприклад, складність алгоритму можна характеризувати числом операторів в програмі або числом команд програми в машинному коді.

Складність завдання складається з складності алгоритму та кількості даних. Кількість даних, що відносяться до задачі, характеризується числом байтів, за

допомогою яких представляються дані. Маючи в своєму розпорядженні відомості про складність алгоритму і кількості даних, можна визначити потребу завдання в ресурсах пам'яті.

Якщо складність алгоритму характеризує потребу алгоритму в пам'яті, то трудомісткість - його потреба в часі, пов'язаному з періодом роботи сукупності пристроїв, засобами яких реалізується алгоритм. Трудомісткість алгоритму тому, іноді називають складністю обчислень. Оцінюється трудомісткість алгоритму кількістю операцій, виконуваних з метою обробки, введення і виведення інформації в процесі виконання завдання. Кожній реалізації алгоритму властивий елемент випадковості, пов'язаний з тим, що вихідні дані представляють собою в загальному випадку випадкову вибірку з безлічі вихідних даних, до яких застосуємо алгоритм. Тому повна характеристика трудомісткості передбачає опис кількості операцій, які виконуються за одну реалізацію алгоритму, випадковими величинами, тобто передбачає визначення законів розподілу числа операцій у реалізації алгоритму. Отримання таких відомостей про алгоритм - складний і тривалий процес. У зв'язку з цим трудомісткість зазвичай характеризують наближено, наприклад, тільки математичними очікуваннями числа виконуваних операцій.

У першому наближенні трудомісткість алгоритму можна охарактеризувати такою сукупністю параметрів:

B - середня кількість процесорних операцій, виконуваних за одну реалізацію алгоритму (при одному прогоні програми);

N1, N2, ..., Nn - середня кількість звернень до файлів F1, F2, ..., Fn відповідно за одну реалізацію алгоритму;

Q1, Q2, ..., Qn - середня кількість інформації (байтів) передаються за одне звернення до файлів F1, F2, ..., Fn відповідно.

Всі оператори алгоритму підрозділяються на функціональні, переходу, вводу/ виводу.

Функціональний оператор - задає сукупність обчислювальних операцій. Оператор переходу - задає правило вибору одного з можливих шляхів розвитку

обчислювального процесу, відповідного поточним значенням даних, відносини між якими представляються предикатами.

Оператор введення / виведення - задає звернення до певного файлу з метою передачі деякої кількості даних.

Перші два типи операторів задають сукупність обчислювальних операцій над даними і відносяться до класу операторів, які називаються основними.

Сукупність операторів алгоритму та зв'язків між ними наочно представляються графом алгоритму, вершини якого відповідають операторам алгоритму, а дуги відображають зв'язки між операторами. Серед вершин графа виділяють початкову, кінцеву та операторні.

Номери вершин графа позначимо 0,1, ..., k, де 0 - початкова, k - кінцева вершина графа. Номери 1,2, ..., k-1 ідентифікують оператори алгоритму. Таким чином, граф алгоритму дає наочне уявлення структури алгоритму, визначаючи множина операторів $V = \{v_1, ..., v_{k-1}\}$ та дуг $D = \{(i, j)\}$; $i = 0, ..., k-1$; $j = 1, ..., k$, зв'язуючих операторів.

Переходи між операторами v_i і v_j розглядаємо як випадкові події і характеризуємо можливостями p_{ij}, тобто кожна дуга (i, j) графа алгоритму позначається числом p_{ij}.

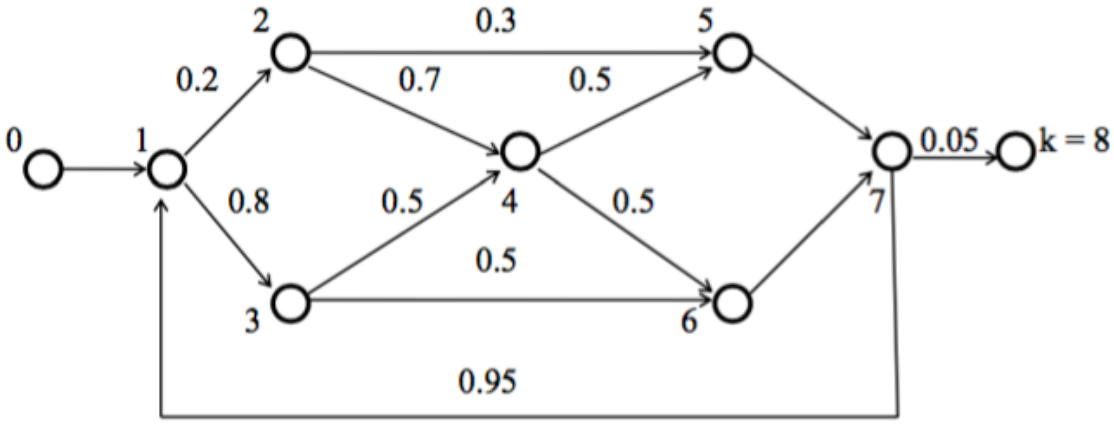
Середнє число $n_1, ..., n_{k-1}$ перебувань марковського процесу в неповоротних станах $S_1, ..., S_{k-1}$ (тобто шукане середнє число $n_1, ..., n_{k-1}$ звернень до операторів $v_1, .. ., v_{k-1}$) визначається коренями системи лінійних алгебраїчних рівнянь:

$$\begin{cases} n_1 = 1 & +p_{11}n_1 & +p_{21}n_2 & +\ldots & +p_{k-1,1}n_{k-1} \\ n_2 = & p_{12}n_1 & +p_{22}n_2 & +\ldots & +p_{k-1,2}n_{k-1} \\ \ldots & & \ldots & & \\ n_{k-1} = & p_{1,k-1}n_1 & +p_{2,k-1}n_2 & +\ldots & +p_{k-1,k-1}n_{k-1} \end{cases}$$

Після перетворень отримаємо:

$$\begin{cases} (p_{11}-1)n_1 + p_{21}n_2 + \ldots + p_{k-1,1}n_{k-1} = -1 \\ p_{12}n_1 + (p_{22}-1)n_2 + \ldots + p_{k-1,2}n_{k-1} = 0 \\ \ldots \\ p_{1,k-1}n_1 + p_{2,k-1}n_2 + \ldots + (p_{k-1,k-1}-1)n_{k-1} = 0 \end{cases}$$

Нехай для деякої конкретної реалізації граф алгоритму має вигляд:



Маємо систему з семи лінійних алгебраїчних рівнянь:

$$\begin{cases} -n_1 & & & & & & +0.95n_7 & = -1 \\ 0.2n_1 & -n_2 & & & & & & = 0 \\ 0.8n_1 & & -n_3 & & & & & = 0 \\ & 0.7n_2 & +0.5n_3 & -n_4 & & & & = 0 \\ & 0.3n_2 & & +0.5n_4 & -n_5 & & & = 0 \\ & & 0.5n_3 & +0.5n_4 & & -n_6 & & = 0 \\ & & & & n_5 & +n_6 & -n_7 & = 0 \end{cases}$$

Рішення системи рівнянь визначає середнє число влучень обчислювального процесу в стану S_1, ..., S_7:

$$n_1 = 20, \quad n_3 = 16, \quad n_5 = 6.6, \quad n_7 = 20.$$
$$n_2 = 4, \quad n_4 = 10.8, \quad n_6 = 13.4,$$

Нехай всі оператори алгоритму - основні, а кількість операцій - k_i, породжуваних оператором v_i, постійна і дорівнює 1. Тоді трудомісткість $\theta_{ОСН} = \sum_{i=1}^{7} k_i n_i = 20 + 4 + ... + 20 = 90.8$ алгоритму буде дорівнює операцій.

Якщо, наприклад, оператори v_4 і v_7 є операторами введення/виведення, а кількість операцій, породжуваних кожним з основних операторів, так само $k_1 = 500$, $k_2 = 500$, $k_3 = 50$, $k_5 = 300$, $k_6 = 20$, відповідно, то трудомісткість по основним операторам складе величину $\theta_{ОСН} = 500 * 20 + 500 * 4 + 50 * 16 + 300 * 6.6 + 20 * 13.4 = 15\,048$ операцій.

Нехай при зверненні з оператора v_4 до файлу F_4 передається l_4 = 1800 байт даних, а при зверненні з оператора v_7 до файлу F_7 передається l_7 = 2540 байт. Тоді трудомісткість по вводу/виводу визначається, як $\theta_{B/B} = \sum_{4,7} \theta_{B/B}^{(h)} = \theta_{B/B}^{(4)} + \theta_{B/B}^{(7)}$. В свою чергу $\theta_{B/B}^{(4)} = \sum_{v_4 \in S_4} n_i \ell_i = n_4 \ell_4 = 10.8*1800 = 19440$ байт. Аналогічно $\theta_{B/B}^{(7)} = n_7 \ell_7 = 20*2540 = 50800$ байт. Тоді $\theta_{B/B} = 19440 + 50800 = 70240$ байт.

Хід роботи

1. Ознайомитись з матеріалами практичної роботи
2. Побудувати по таблиці 1, у відповідності з варіантом, граф алгоритму.
3. Побудувати математичну модель обчислювального процесу для оцінки трудоємності алгоритму.
4. Підготовити звіт по виконаній роботі.

Таблиця 1

Вариант	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
P_{12}	0.2	0.1	1	0.5	1	1	0.3	1	0.2	0.3	0.1	1	1	0.2
P_{13}	0.2	0.3		0.5			0.7		0.4	0.7	0.1			0.8
P_{14}	0.6	0.6							0.4		0.8			
P_{23}			0.1		0.3	0.6		0.2				0.2	0.5	
P_{24}			0.3	0.4	0.7	0.4	0.5	0.3		0.7		0.8	0.5	0.3
P_{25}	1	1	0.6	0.6			0.5	0.5	1	0.3	1			07
P_{34}					1							1		
P_{35}	0.1	0.3		0.3		0.1	0.1		0.1	0.6	0.2		0.2	0.2
P_{36}	0.9	0.3	1			0.9	0.2	1	0.2		0.8		0.8	0.2
P_{37}		0.4		0.7			0.7		0.7	0.4				0.6
P_{45}					0.5							0.3		
P_{46}	1		1	1	0.5	0.3	1	1		1	1	0.7	0.2	1
P_{47}		1				0.7			1				0.8	
P_{56}		1	1	1			1	1	1	1				1
P_{57}	1				1	1					1	1	1	
P_{61}			0.9					0.8						
P_{67}	1	1	0.1		1	1	1	0.2	1		1	1	1	1
P_{68}				1						1				
P_{71}	0.9	0.8					0.8		0.8		0.9			0.9
P_{72}					0.8							0.5		
P_{78}				1		1				1			1	
P_{81}				0.9		0.7				0.8			0.6	

Вариант	15	16	17	18	19	20	21	22	23	24	25	26	27	28
P_{12}	0.2	0.1	1	0.5	1	1	0.3	1	0.2	0.3	0.1	1	1	0.2
P_{13}	0.2	0.3		0.5			0.7		0.4	0.7	0.1			0.8
P_{14}	0.6	0.6							0.4		0.8			
P_{23}			0.1		0.3	0.6		0.2				0.2	0.5	
P_{24}			0.3	0.4	0.7	0.4	0.5	0.3		0.7		0.8	0.5	0.3
P_{25}	1	1	0.6	0.6			0.5	0.5	1	0.3	1			07
P_{34}					1							1		
P_{35}	0.1	0.3		0.3		0.1	0.1		0.1	0.6	0.2		0.2	0.2
P_{36}	0.9	0.3	1			0.9	0.2	1	0.2		0.8		0.8	0.2
P_{37}		0.4		0.7			0.7		0.7	0.4				0.6
P_{45}					0.5							0.3		
P_{46}	1		1	1	0.5	0.3	1	1		1	1	0.7	0.2	1
P_{47}		1				0.7			1				0.8	
P_{56}		1	1	1			1	1	1	1				1
P_{57}	1				1	1					1	1	1	
P_{61}			0.9				0.1	0.8			0.3			0.6
P_{67}	1	1	0.1		1	1	0.9	0.2	1		0.7	1	1	0.4
P_{68}				1						1				
P_{71}	0.8	0.6	0.7		0.3		0.8		0.8		0.9	0.2		0.9
P_{72}	0.1	0.2		0.3	0.5	0.2		0.7	0.1	0.6		0.5	0.2	
P_{78}				0.7		0.8				0.4			0.8	
P_{81}				0.9		0.7				0.8			0.6	

Таблиця 2

вариант	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	100	200	120	320	250	300	100	100	200	150	100	100	300	100
2	200	150	120	200	300	200	400	200	300	150	200	120	200	200
3	120	300	150	200	200	300	500	100	200	200	100	200	300	300
4	300	250	200	150	150	800	400	200	100	200	250	300	200	300
5	100	200	300	150	300	300	300	300	250	400	500	300	250	250
6	300	300	600	300	400	200	250	300	250	300	300	150	200	150
7	100	800	300	100	100	200	200	400	300	200	200	100	150	200
8	-	-	-	800	-	900	-	-	-	200	-	-	200	-

вариант	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	200	400	240	640	500	600	200	200	400	300	200	200	600	200
2	400	300	240	400	600	400	800	400	600	300	400	240	400	400
3	240	60	300	400	400	600	900	200	400	400	200	400	600	600
4	600	500	400	300	300	900	800	400	200	400	500	600	400	600
5	200	400	600	300	600	600	600	600	500	800	900	600	500	500
6	600	600	900	600	400	400	500	600	500	600	600	300	400	300
7	200	700	600	200	100	400	400	800	300	400	400	100	150	400
8	-	-	-	900	-	800	-	-	-	700	-	-	600	-

Затемнення комірки означає, що відповідний оператор являється оператором вводу/виводу

Практична робота №2

Тема: Знаходження оптимальної модифікації комп'ютерної системи

Мета: Розробка ефективної структури ОС для обчислення арифметичних виразів

Теоретичні відомості

Критерії ефективності КС. Як критерії ефективності розв'язання задачі (обчислення арифметичних виразів) будемо розглядати:

• коефіцієнт прискорення

$$K_n = T_0 / T_n, \qquad (1)$$

де T_0 – час розв'язання задачі на традиційній ЕОМ (однопроцесорній), який дорівнює сумі часу виконання операцій додавання, множення та ділення, які складають обчислювальну задачу, що розглядається;

T_n – час розв'язання задачі в ОС;

• коефіцієнт завантаження процесорів (процесорних елементів)

$$K_з = T_0 / (N * T_n), \qquad (2)$$

де N – кількість процесорів (процесорних елементів) в КС.

Для визначення перерахованих показників ефективності достатньо знати не абсолютні величини часу виконання різних арифметичних операцій, а їх відносні співвідношення. Для цього введемо такі позначення:

$$\alpha = t_m / t_c; \ \beta = t_g / t_c, \qquad (3)$$

де t_c – час виконання операції додавання (сума);

t_m – час виконання операції множення;

t_g – час виконання операції ділення.

Приклад. Значну частину програм розв'язання інженерних і науково-технічних задач складають обчислення арифметичних виразів. Будь-який арифметичний вираз із змінними можна графічно подати у вигляді дерева. Дерево виразу складається з набору вузлів (вершин), кожний з яких містить

окрім даних покажчики на вузли нижнього рівня. Верхній вузол (корінь дерева) відповідає операції, яка виконується останньою. З нього починається побудова дерева.

На рис. 1 зображено дерево арифметичного виразу:

$$a + bc + d / (e + f) + gh. \quad (5)$$

Час обчислення даного арифметичного виразу на традиційній ЕОМ можна визначити таким чином:

$$T_0 = 4t_c + 2t_m + t_g. \quad (6)$$

Розглянемо можливість скорочення часу обчислення арифметичного виразу за рахунок організації паралельного виконання операцій і використання обчислювальних систем різних структур.

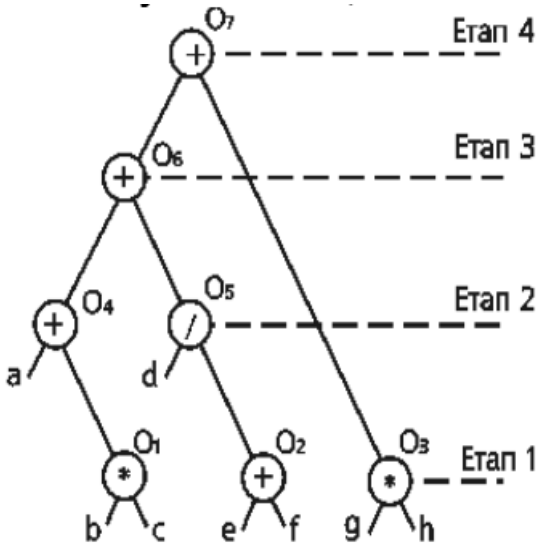


Рисунок 1 – Дерево арифметичного виразу

Обчислення виразу (5) здійснюється за чотири етапи.

На першому етапі можливе паралельне виконання трьох операцій:

$$O_1 = bc,\ O_2 = e + f,\ O_3 = gh.$$

На другому етапі можливе паралельне виконання двох операцій:

$$O_4 = a + O_1;\ O_5 = d/O_2.$$

На третьому етапі можливе паралельне виконання однієї операції:

$$O_6 = O_4 + O_5.$$

На четвертому етапі виконується операція $O_7 = O_6 + O_3$.

Ранг задачі обчислення арифметичного виразу (5), який визначається кількістю виконуваних паралельно на кожному етапі операцій, змінюється від трьох на першому етапі до однієї на четвертому.

Розглянемо організацію обчислення арифметичного виразу (5) в різних обчислювальних системах для випадку N=3, α=3, β=3.

Час розв'язання задачі, що розглядається в традиційній ЕОМ при даних α і β, складе на основі формули (6) $T_0 = 13t_c$.

Для варіанта М1 є можливість організації паралельного виконання всіх трьох операцій на етапі 1 (рис. 1).

На рис. 2. зображені часові діаграми завантаженості процесорів системи.

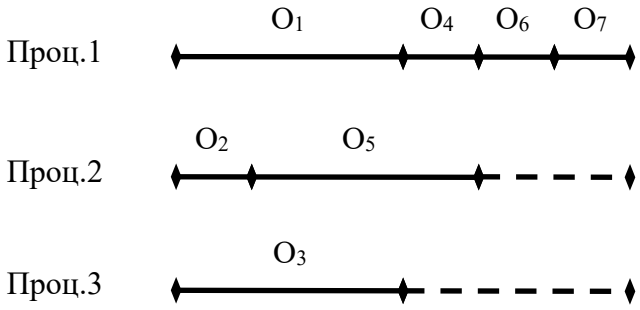


Рисунок 2 – Часові діаграми завантаженості процесорів системи М2.1

При такому розподілі операцій по процесорах час розв'язання задачі

$$T_{M2.1} = t_c + t_g + 2t_c = 6t_c.$$

За рахунок зміни порядку виконання операцій при обчисленні арифметичних виразів можливе скорочення часу їх обчислення. Тобто знаходження більш оптимального алгоритму. Така оптимізація базується на тому, що деякі операції та операнди починаються певними законами, враховуючи які, можна здійснювати перетворення початкового виразу. Можна виділити такі закони, як комутативний, асоціативний, дистрибутивний та ін.

Використовуючи співвідношення для прийнятих N, α і β, визначимо коефіцієнти прискорення і завантаження:

$$K_П = T_0 / T_{M2.1} = 2{,}16,$$

$$K_3 = T_0 / (NT_{M2.1}) = 0{,}72.$$

Для другого варіанта організації роботи системи (М2) характерне використання “вектор-інструкції”. При цьому необхідно враховувати, що час виконання кожної “вектор-інструкції” ($t_{вi}$) визначається часом виконання самої тривалої команди в цій “вектор-інструкції”.

Послідовність і компоненти “вектор-інструкції” для реалізації в комп’ютерних системах типу М2 арифметичного виразу, зображеного деревом на рис. 1 подані в табл. 1.

Таблиця 1

“Вектор-інструкції” для реалізації арифметичного виразу в КС типу М2

Вектор-інструкція			$t_{вi}$
Процесор 1	Процесор 2	Процесор 3	
$O_1 = bc$	$O_2 = e + f$	$O_3 = g + h$	t_m
$O_4 = a + O_1$	$O_5 = d / O_2$	-	t_g
$O_6 = O_4 + O_3$	-	-	t_c
$O_7 = O_6 + O_5$	-	-	t_c

Часові діаграми завантаження процесорів системи показано на рис. 3.

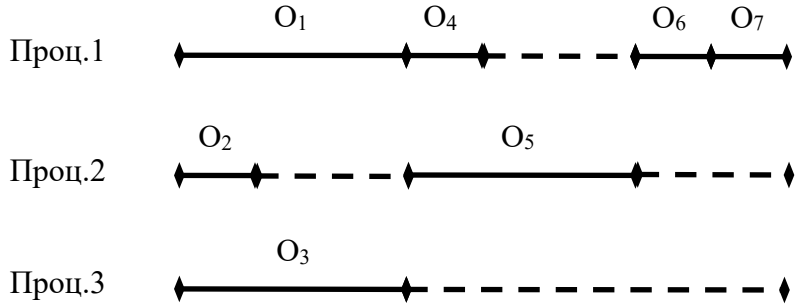


Рисунок 3 – Часові діаграми завантаженості процесорів системи М2

Час розв’язання задачі

$$T_{M2.2} = t_m + t_g + 2t_c = 8t_c.$$

Значення коефіцієнтів прискорення і завантаження:

$$K_П = T_0 / T_{M2.2} = 1{,}63;$$

$$K_3 = T_0 / (NT_{M2.2}) = 0{,}54.$$

Обчислювальна система типу М3 в кожний момент часу допускає лише виконання однакових операцій. Тому доцільно звести дерево вихідного арифметичного виразу до вигляду, зручного для обробки в обчислювальній системі типу М3 (рис. 4). Обчислення арифметичного виразу, який розглядається в обчислювальній системі типу М3, здійснюється в п’ять етапів.

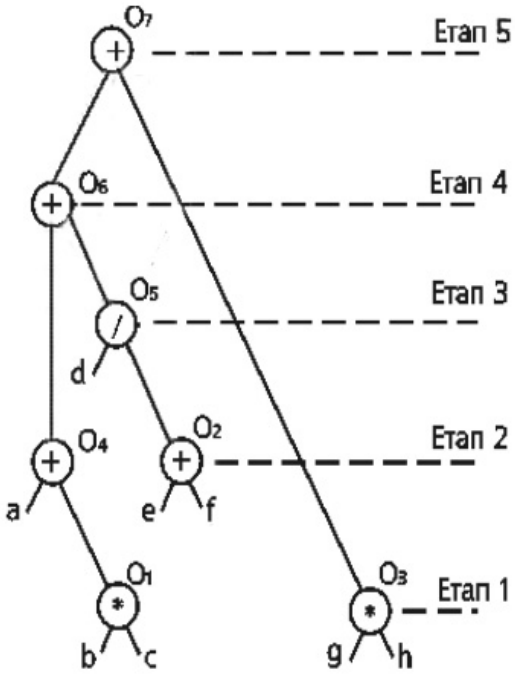
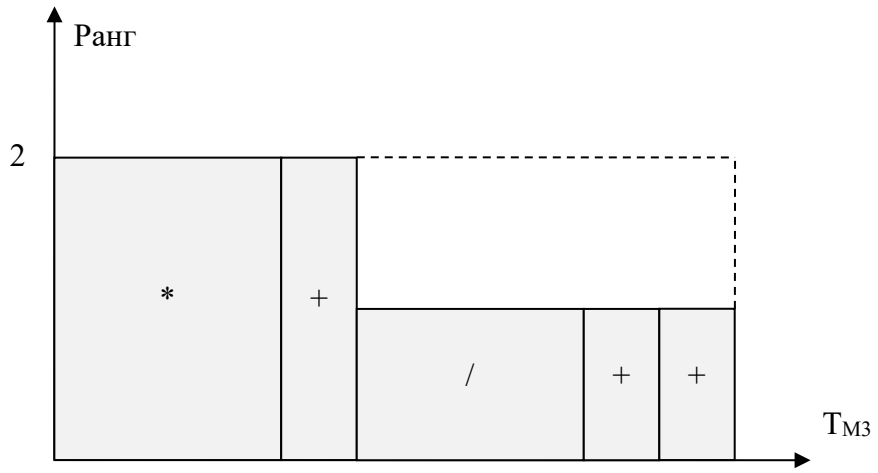


Рисунок 4 – Дерево арифметичного виразу для системи М3

При цьому

$$T_{M3} = t_m + t_g + 3t_c = 9t_c.$$

На рис. 5 зображена просторово-часова діаграма розв’язання задачі. Як видно, ранг задачі не перевищує 2 і один процесор при N=3 не буде працювати. Пунктиром на просторово-часовій діаграмі позначена площа, що відповідає роботі, яку можна виконати на обчислювальній системі з N=3 за час T_{M3}, заштрихована площа просторово-часової діаграми відповідає роботі, виконаній системою в дійсності. Їх відношення і визначає коефіцієнт завантаження процесорів.



Для розглянутої задачі

$$K_П = 1,37;$$

$$K_З = 0,46.$$

Таблиця 2

Значення коефіцієнтів прискорення і завантаження під час розв'язання задачі обчислення арифметичного виразу в різних КС

Тип КС	**$K_П$**	**$K_З$**
М1	2,2	0,73
М2	1,6	0,54
М3	1,37	0,46

Аналіз результатів ефективності різних структур обчислювальної системи під час розв'язання задачі, що розглядається, дозволяє зробити такі висновки:

- використання обчислювальної системи типу М1 дозволяє розв'язати задачу за мінімальний час;

- за ступенем використання обладнання (завантаження процесорів) перевагу слід віддати системі типу М3.

Одержані оцінки характеризують системи при розв'язанні конкретної задачі. Природно припустити, що для таких арифметичних виразів та змін

параметрів α і β, а також значення коефіцієнтів прискорення та завантаження $K_П$ і $K_З$ змінюються.

Хід роботи

1. Накреслити дерево заданого арифметичного виразу.
2. Визначити час T_0 обчислення виразу в традиційній ЕОМ.
3. Для багатопроцесорної ОС конкретного типу обчислити значення T_N , $K_П$, $K_З$, та звести їх у таблицю.

Завдання

Варіант	**N**	**A**	**β**	**Вираз**
1	3	2	4	A+B*C+D/E+F(G+H)+K/L
2	3	3	4	A*B*C+D*E+F(G+H)+K+L
3	4	2	3	(A+B)/(C+D)*E*F+G/H+K/L
4	3	2	4	(A+B)/(C+D*E)+F+(G+H)/K*L
5	2	3	4	A/B*(C+D)+E*F*G*H+K/L
6	3	3	5	A+B*C(D+E/F)+G*H/(K+L)
7	4	2	4	(A+B*C+D*E)/(F+G*H+K*L)
8	3	3	4	A*B+C*D+E*F+G*H+K/L
9	3	2	3	A+B+C+D+E/(F+G/(H+K*L))
10	4	2	4	A(B+C(D+E(F+G(H+K/L))))
11	2	3	4	A+B*C+D/E+F(G+H)+K/L
12	3	2	4	A+B*C(D+E/F)+G*H/(K+L)
13	4	3	3	A*B*C+D*E+F(G+H)+K+L
14	3	2	5	A+B+C+D+E/(F+G/(H+K*L))
15	3	3	4	(A+B)/(C+D*E)+F+(G+H)/K*L
16	4	3	3	(A+B*C+D*E)/(F+G*H+K*L)
17	3	2	5	(A+B)/(C+D)*E*F+G/H+K/L
18	4	2	3	A*B+C*D+E*F+G*H+K/L
19	3	3	4	A/B*(C+D)+E*F*G*H+K/L

Практична робота №3

Тема: Архітектура комп'ютерних конвеєрних систем

Мета: Аналіз структур конвеєрних обчислювальних систем

Теоретичні відомості

Конвеєр являє собою процесор, розділений на P частин (шарів, рядків), виконуючих послідовно етапи кожної обчислювальної задачі (операції). В той час як j-й шар процесора виконує j-й етап деякої k-ї задачі, (j-1)-й шар може виконувати (j-1)-й етап (k+1)-ї задачі, (j-2)-й шар може виконувати (j-2)-й етап (k+2)-ї задачі і так далі, тобто 1-й шар може у цей же час виконувати 1-й етап (i+j-1)-ї задачі (де j=2,...,P).

Таким чином, кожна задача або операція виконується за P етапів при проходженні усіх P шарів конвеєрного процесора. В обчислювальних системах (ОС) з конвеєрним процесором може виконуватися одночасно декілька (P) операцій по перетворенню даних, що відповідає визначенню обчислюваної системі. Проте ці операції в будь-який момент часу обов'язково знаходяться на різних етапах виконання й у принципі не можуть починатися всі одночасно.

Конвеєрні системи з роздільним керуванням, орієнтовані на паралелізм незалежних гілок (тип К1). На рис. 5.1 зображена конвеєрна система з роздільним керуванням. Вона складається з P пристроїв керування (ПК) – по кількості шарів, на які розділений конвеєрний процесор. Принципово важливим є наявність P роздільних вузлів формування адреси наступної інструкції.

Пристрій керування приєднаний до шарів конвеєрного процесора через кільцевий комутатор. Комутатор влаштований так, що в 1-му такті роботи ОС до 1-го шару процесора підключено 1-й ПК, який починає виконувати свої задачі (операції); в 2-му такті 1-й ПК переключається на 2-й шар процесора, а до 1-го шару підключається 2-й ПК і т.д. У p-му такті 1-й ПК підключено до P-го шару процесора, де завершується виконання 1-ї задачі (операції), до (P-1)-го шару підключається 2-й ПК,..., до 1-го шару підключено P-й ПК. Після закінчення 1-ї

задачі (операції) 1-й ПК в (р+1)-му такті 1-й пристрій знову підключається до 1-го шару процесора, де починається виконання 2-ї задачі (операції) і т.д.

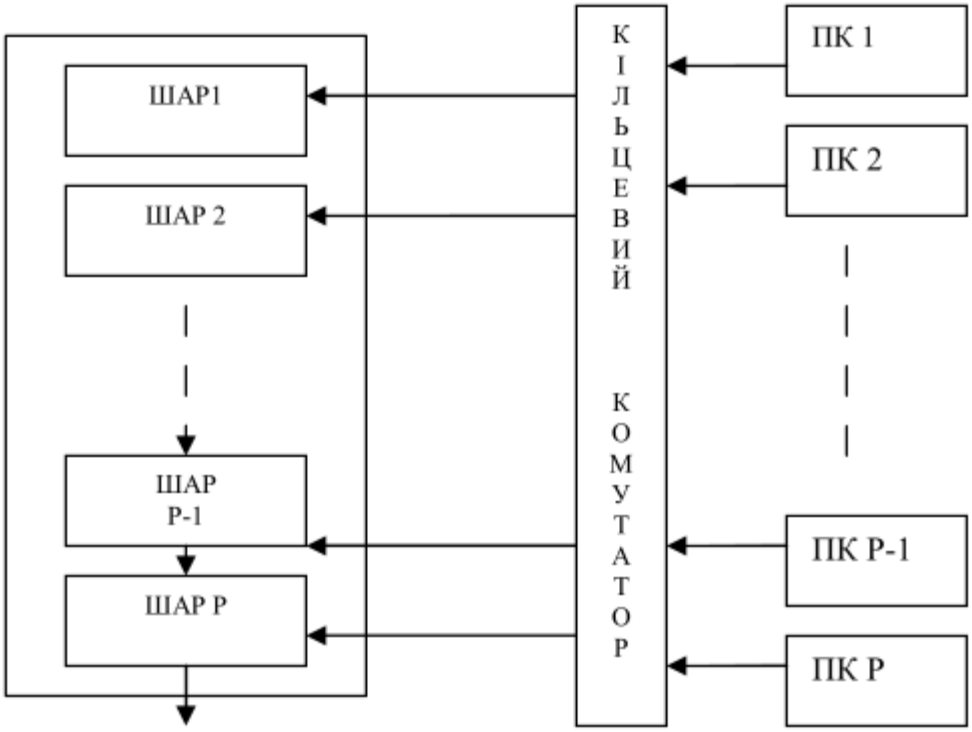


Рисунок 3.2 – Конвеєрна система з роздільним керуванням

Розглянутий конвеєрний процесор є багатофункціональним, тобто дозволяє виконувати одночасно різні операції. Проте при цьому тривалість такту роботи шарів процесора залежить від виконуваних операцій та буде визначатися часом виконання самої тривалої операції.

Конвеєрні системи з загальним керуванням, орієнтовані на використання паралелізму суміжних операцій. Пристрій керування (рис. 5.2) один, але регістрів для збереження інструкцій (РгІ) стільки ж, скільки шарів є в процесорі. 1-ша інструкція програми зчитується в РгІ(1), та в 1-му шарі процесора починається виконання. В наступному такті 1-ша інструкція передається в РгІ(2), її виконання продовжує 2-й шар процесора, а до РгІ(1) зчитується 2-а інструкція програми та в 1-му шарі конвеєрного процесора починається її виконання і т.д.

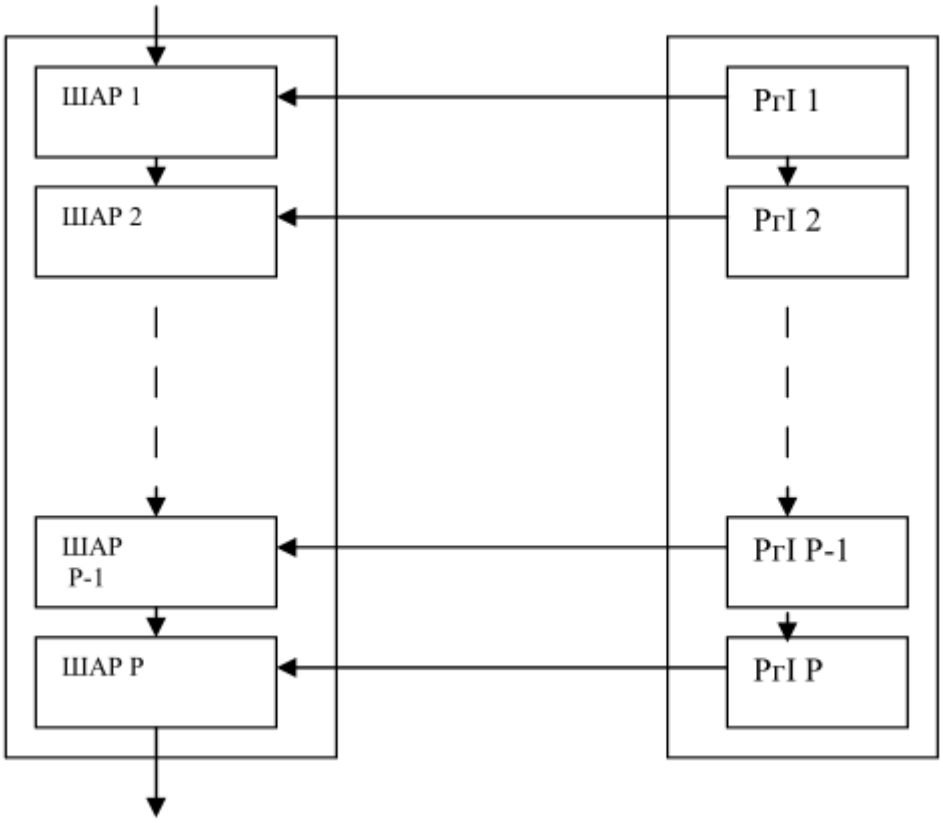


Рисунок 3.2 – Конвеєрна система з загальним керуванням

Розглянутий конвеєрний процесор системи типу К2 теж є багатофункціональним. Будемо розрізняти динамічну та статичну перебудови процесора для виконання різних операцій.

Динамічна перебудова конвеєра (тип К2.1) дозволяє виконувати різні операції одночасно в різних шарах конвеєра. Але так само, як і для систем типу К1, такт конвеєра в кожний момент часу буде визначатися часом виконання самої тривалої операції (у випадку використання конвеєра зі змінним тактом) або часом виконання самої тривалої з усіх операцій, на виконання якої орієнтовано даний багатофункціональний конвеєр (у випадку використання конвеєра з постійним тактом).

У випадку статичної перебудови конвеєра (тип К2.2) на виконання нової операції необхідно дочекатися звільнення конвеєра від попередньої операції та тільки після цього завантажувати наступну операцію (якщо вона відрізняється від виконуваної).

У випадку конвеєра з постійним тактом його тривалість дорівнює часу виконання найтривалішої операції, яка в принципі може бути виконана багатофункціональним конвеєром.

Критерії ефективності конвеєрної системи. Як критерії ефективності розв'язання задачі (обчислення арифметичних виразів) будемо розглядати:

• коефіцієнт прискорення

$$K_n = T_0/T_N, \qquad (1)$$

де T_0 – час розв'язання задачі в традиційній ЕОМ (однопроцесорній), який дорівнює сумі часу виконання операцій додавання, множення та ділення;

T_N – час розв'язання задачі в конвеєрній системі;

• коефіцієнт завантаження конвеєра

$$K_З = T_0/(N*T_n), \qquad (2)$$

де N – кількість шарів в конвеєрі.

Приклад. Зробимо аналіз функціонування конвеєрних ОС різних типів для заданого арифметичного виразу

$$(A+B)+C/D+G*(K/L+M+N) \qquad (3)$$

Будь-який арифметичний вираз із змінними можна графічно подати у вигляді дерева.

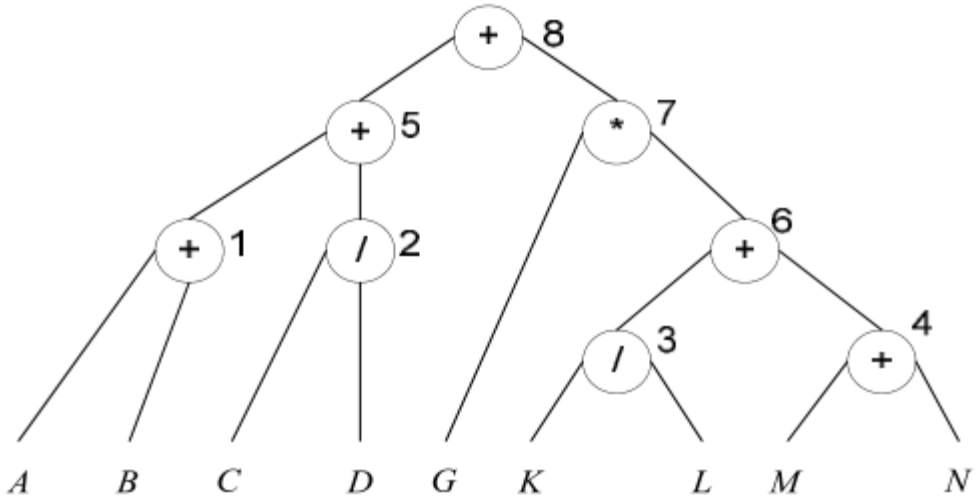


Рисунок 3.3 – Дерево арифметичного виразу

Час обчислення даного арифметичного виразу в традиційній ЕОМ можна визначити таким чином:

$$T_0 = 5T_C + 2T_g + T_m,$$

де T_C – час операції додавання, T_g – час операції ділення, T_m – час операції множення.

Нехай задано $\tau_c = 1$, $\tau_g = 5\tau_c$, $\tau_m = 2\tau_c$,

де τ_c – час операції додавання в одному шарі конвеєра, τ_g – час операції ділення в одному шарі конвеєра, τ_m – час операції множення в одному шарі конвеєра.

Відповідно $T_C = N*\tau_c$; $T_g = N*5*\tau_c$; $T_m = N*2*\tau_c$.

Тоді при послідовному виконанні всіх операцій даного виразу в конвеєрі з N=4, де N – кількість шарів конвеєра

$$T_0 = 5*4*\tau_c + 2*4*5*\tau_c + 4*2*\tau_c = 68\tau_c.$$

1) Розглянемо діаграму роботи конвеєра з динамічною перебудовою, наведеного на рис. 3.3, для випадку з N=4 (рис. 3.4).

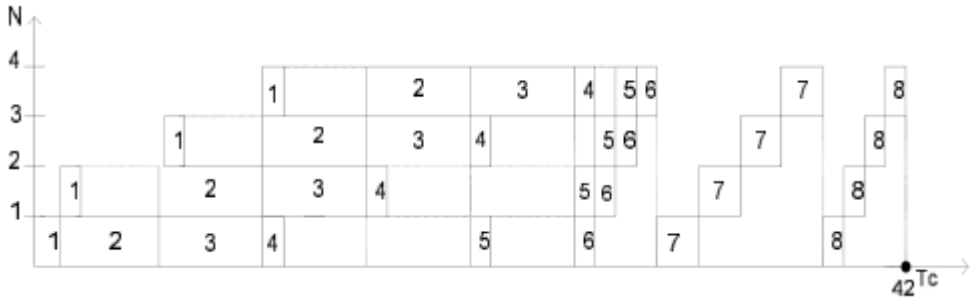


Рисунок 3.4 – Діаграма роботи конвеєра з динамічною перебудовою

Використовуючи вирази (1) та (2), визначимо коефіцієнти прискорення та завантаження:

$$K_{ï} = T_0 / T_{äèí} = \frac{68\tau_{ñ}}{42\tau_{ñ}} \approx 1{,}62$$

$$K_{ç} = T_0 / (N \cdot T_{äèí}) = \frac{68\tau_{ñ}}{4 \cdot 42\tau_{ñ}} \approx 0{,}405$$

2) Розглянемо діаграму роботи конвеєра зі статичною перебудовою (рис. 3.5).

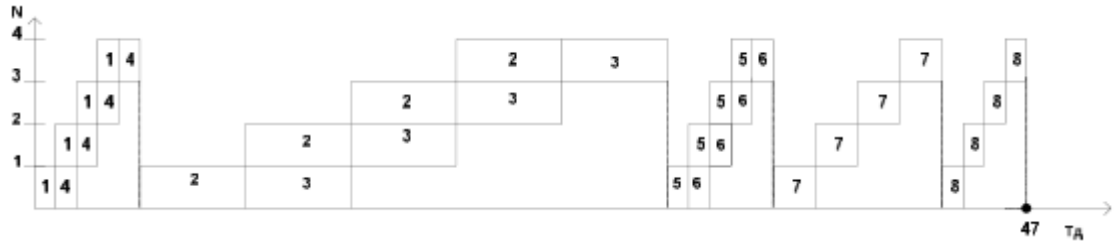


Рисунок 3.5 – Діаграма роботи конвеєра зі статичною перебудовою

Використовуючи вирази (1) та (2), визначимо коефіцієнти прискорення та завантаження:

$$K_{ï} = T_0 / T_{äèí} = \frac{68\tau_{ñ}}{47\tau_{ñ}} \approx 1{,}45$$

$$K_{ç} = T_0 / (N \cdot T_{äèí}) = \frac{68\tau_{ñ}}{4 \cdot 47\tau_{ñ}} \approx 0{,}362$$

3) Розглянемо діаграму роботи конвеєра з постійним тактом (рис. 3.6).

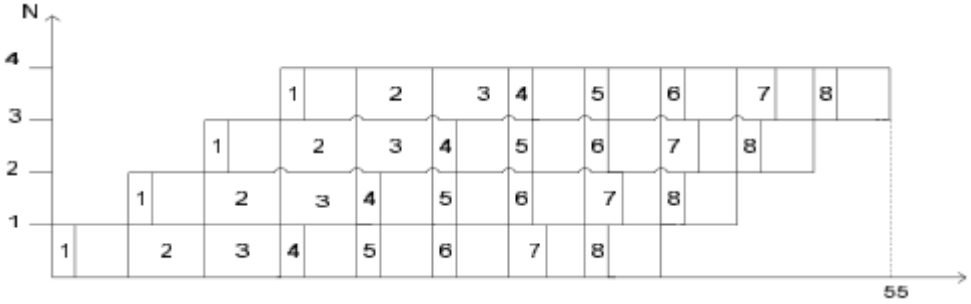


Рисунок 3.6 – Діаграма роботи конвеєра з постійним тактом

Використовуючи вирази (1) та (2), визначимо коефіцієнти прискорення та завантаження:

$$K_{ï} = T_0 / T_{äèí} = \frac{68\tau_{ñ}}{55\tau_{ñ}} \approx 1{,}24$$

$$K_{ç} = T_0 / (N \cdot T_{äèí}) = \frac{68\tau_{ñ}}{4 \cdot 55\tau_{ñ}} \approx 0{,}309$$

В табл. 3.1 наведено значення коефіцієнтів прискорення та завантаження під час розв’язання задачі обчислення арифметичного виразу в конвеєрах різних типів.

Таблиця 6.1

Значення коефіцієнтів прискорення та завантаження

Тип конвеєра	**$К_п$**	**$К_з$**
З динамічною перебудовою К2.1	1,62	0,405
Зі статичною перебудовою К2.2	1,45	0,362
З постійним тактом К1	1,24	0,309

Аналіз результатів ефективності конвеєрів різних типів під час розв’язання задачі, що розглядається, дозволяє зробити такі висновки:

• використання конвеєру типу К2.1 дозволяє розв’язати задачу за мінімальний час;

• за ступенем використання обладнання (завантаження конвеєра) перевагу слід віддати конвеєру типу К1.

Хід роботи

4. Накреслити дерево заданого арифметичного виразу.
5. Визначити час T_0 обчислення виразу в традиційній ЕОМ.
6. Виконати задачу оптимальної завантаженості (задачу планування обчислень) для кожного типу конвеєрних ОС, що розглядаються.
7. Обчислити значення для типів конвеєру, що розглядаються, та звести їх у таблицю
8. Зробити аналіз функціонування конвеєрів різних типів на підставі коефіцієнтів

Завдання

Варіант	**N**	τ_m	τ_g	**Вираз**
1	3	2	3	(A+B+C)*(D+G)/E+L*K/F
2	2	3	4	A*B+C*D+G*K(L+H)*E
3	3	2	4	(A+B/C*G)*(K+E+L)/R+D
4	2	3	5	A*B/C+D*E(G+K/L)+M
5	4	2	5	A+B+C/D+G*(K/L+M+N)
6	2	2	4	A+B*(C+D*E*(G+L/K))+N
7	3	4	5	A*(B+C/D)+G*K*L+M/N
8	2	3	5	A/B+(C+D*E)*(G+K/L*M)
9	4	2	3	(A*B+C/D+G*K)(M+N+E)
10	4	2	4	A/B+C/D+G*(K+L*(M+N))
11	3	2	4	A+B*C+D/E+F(G+H)+K/L
12	3	3	4	A*B*C+D*E+F(G+H)+K+L
13	4	2	3	(A+B)/(C+D)*E*F+G/H+K/L
14	3	2	4	(A+B)/(C+D*E)+F+(G+H)/K*L
15	2	3	4	A/B*(C+D)+E*F*G*H+K/L
16	3	3	5	A+B*C(D+E/F)+G*H/(K+L)
17	4	2	4	(A+B*C+D*E)/(F+G*H+K*L)
18	3	3	4	A*B+C*D+E*F+G*H+K/L
19	3	2	3	A+B+C+D+E/(F+G/(H+K*L))
20	4	2	4	A(B+C(D+E(F+G(H+K/L))))

Практична робота №4

Тема: Архітектура паралельних обчислювальних систем

Мета: Аналіз паралельних обчислювальних систем.

Теоретичні відомості

В основі паралельних обчислювальних систем (ОС) полягає ідея, що вирішення задачі на паралельних процесорах забере в р- разів менше часу, ніж на одному комп'ютері. Ця ідея довгий час не могла бути реалізована у повному обсязі через відсутність потужної елементної та конструкторсько-технологічної бази, а також відповідного системного та прикладного програмного забезпечення. Положення різко змінилось зі створенням великих та надвеликих інтегральних схем, мікропроцесорів, мікро-ЕОМ, а також розвитку паралельного програмування та паралельних обчислювальних методів.

Для побудови паралельних ОС використовують серійні мікропроцесори та мікро-ЕОМ, а також розробляють відповідним чином орієнтовані мікропроцесори та мікро-ЕОМ (трансп'ютери). Позитивною якістю паралельних ОС є їх нарощуваність.

Важливою класифікаційною ознакою паралельних ОС є спосіб керування сукупністю процесорів. В обчислювальних системах типу ОКМД (одиночний потік команд – множинний потік даних) всі р процесорів ОС знаходяться під керуванням головного, керуючого процесора. У кожному такті всі процесори виконують одну й ту саму команду або простоюють. Таким чином, в одному потоці команд обробляється багато потоків даних, що проходять через процесори. До них відносять також ОС, процесори яких виконують однакові програми й обмінюються між собою даними синхронно.

У паралельних ОС класу МКМД (множинний потік команд – множинний потік даних) окремі процесори працюють під керуванням своїх власних пристроїв керування і виконують різні гілки програм.

Іншою важливою ознакою архітектури паралельних ОС є тип пам’яті – загальна або локальна. У паралельних ОС із загальною пам’яттю всі процесорні елементи (ПЕ) мають пам’ять, що ПЕ ділять між собою для доступу (рис. 4.1).

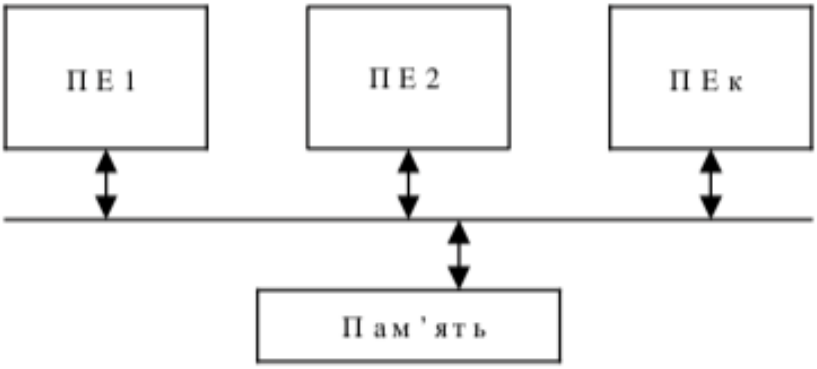


Рисунок 4.1 – Структура паралельної ОС із загальною пам’яттю

Перевагою таких ОС є швидкий обмін даними між процесорами. Недоліком є затримки доступу до пам’яті при одночасному звертанні різних процесорів, при цьому тривалість затримки збільшується з ростом кількості процесорів. Цій недолік можна мінімізувати, якщо кожному процесору надати кеш-пам’ять (рис. 4.2.).

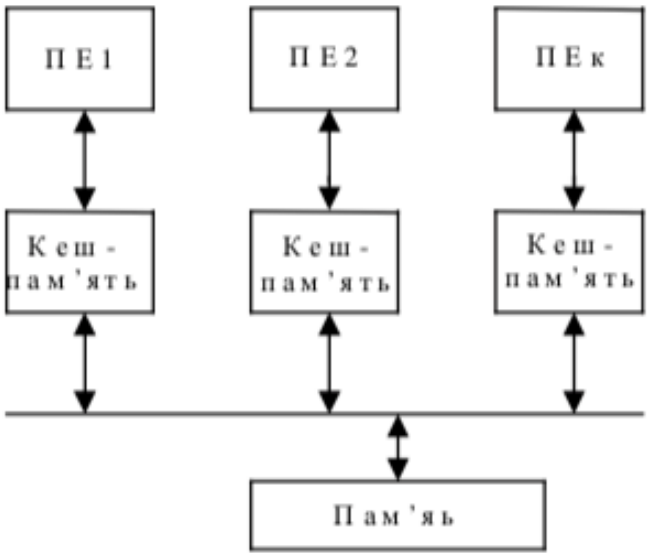


Рисунок 4.2 – Структура паралельної ОС із кеш-пам’яттю

Кожний ПЕ може мати власну локальну пам’ять для збереження своїх програм і проміжних результатів. Тоді загальна пам’ять використовується для

даних, необхідних більш ніж одному ПЕ. Всі взаємодії між ПЕ виконуються через загальну пам’ять. Її недолік полягає в тому, що кілька процесорів можуть вимагати доступ до цієї пам’яті одночасно. Дозвіл отримує тільки один ПЕ. У цьому випадку виникають конфліктні затримки, пов’язані із забезпеченням доступу до пам’яті, які можуть зростати зі збільшенням кількості процесорів.

Для зменшення конфліктів використовується розподілена пам’ять банків з локально-адресованими просторами (рис. 4.3).

Рисунок 4.3 – Структура паралельної ОС із розподіленою пам’яттю банків з локально-адресованими просторами

Розподілена пам’ять поділена на банки з локально- адресованими просторами. Це дозволяє з деякою затримкою у часі майже одночасно звертатись різним ПЕ до різних банків. Комутаційна мережа необхідна для підключення ПЕ до різних банків пам’яті під час виконання паралельної програми.

Як комутаційна мережа ОС із розподіленою пам’яттю може застосовуватись багатошинна топологія та перехресна комутація.

Багатошинна топологія передбачає наявність m незалежних шин для підключення до m банків пам’яті. (рис. 4.4). У цьому випадку виникають конфлікти при захваті шини декількома ПЕ, що є серйозною проблемою при

збільшенні кількості ПЕ. Це достатньо громіздка структура. Перевагою такого способу комутації є висока надійність. Така топологія цілком прийнятна для високопродуктивних ОС. Пропускна здатність зростає пропорційно кількості шин. У порівнянні з одношинною архітектурою управління мережею з декількома шинами складніше через необхідність запобігання конфліктів, що виникають, коли в парах вузлів, що обмінюються по різних шинах, присутній загальний вузол.

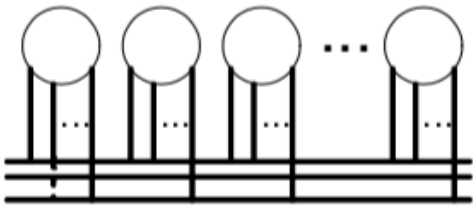


Рисунок 4.4 – Шинна топологія ОС

Перехресний комутатор (рис. 4.5) дозволяє домогтися повної зв’язаності ОС за рахунок міжпроцесорних обмінів через загальні банки пам’яті. Вертикальні лінії перехресного комутатора підключені до m блоків пам’яті, а горизонтальні – до n процесорів. На перетинанні вертикальних і горизонтальних ліній установлені перемикачі. Однак їх число, рівне n*m, для більших систем стає непрактичним.

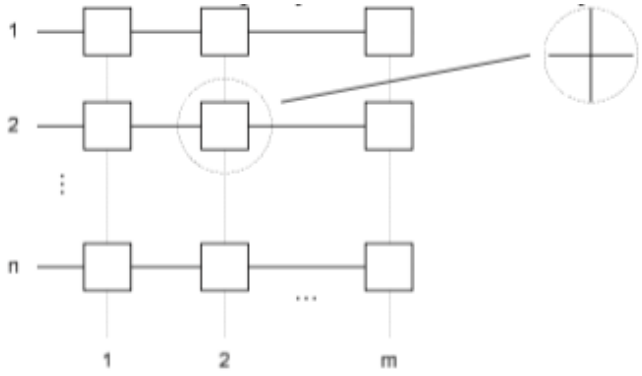


Рисунок 4.5 – Топологія перехресної комутації

Використання перехресної комутації ОС на основі матричного комутатора являє собою класичний приклад одноступінчатої динамічної мережі. Матричний комутатор використовується для зв’язку ПЕ з банками пам’яті. Це найшвидший

спосіб доступу ПЕ до банків пам’яті. Недоліками є висока вартість та залежність надійності ОС від надійності комутатора.

При використанні розподіленої пам’яті кількість ПЕ, що одночасно отримують дозвіл на доступ до пам’яті дорівнює кількості банків цієї пам’яті. Конфлікти до пам’яті зберігаються лише у випадках одночасного доступу декількох ПЕ до одного й того ж банку пам’яті. Таким чином, структура системи на рис. 4.3 є найбільш досконалою, але нарощуваність (масштабованість) цих систем обмежена (до 64 ПЕ).

Повністю конфлікти доступу до пам’яті усунуті в архітектурах із локальною пам’яттю. У таких системах кожний процесор має власну пам’ять та адресується лише до неї. Процесори обмінюються між собою шляхом передавання відповідних повідомлень. Такі системи мають необмежену масштабованість.

Можна виділити 4 переваги паралельних ОС у порівнянні із послідовними:

1.Паралельна архітектура – єдиний спосіб побудови надпродуктивних ОС.

2.Відношення продуктивність/вартість у паралельних ОС вищий, ніж у послідовних.

3.Паралельні ОС дозволяють нарощувати продуктивність в результаті модульної конструкції.

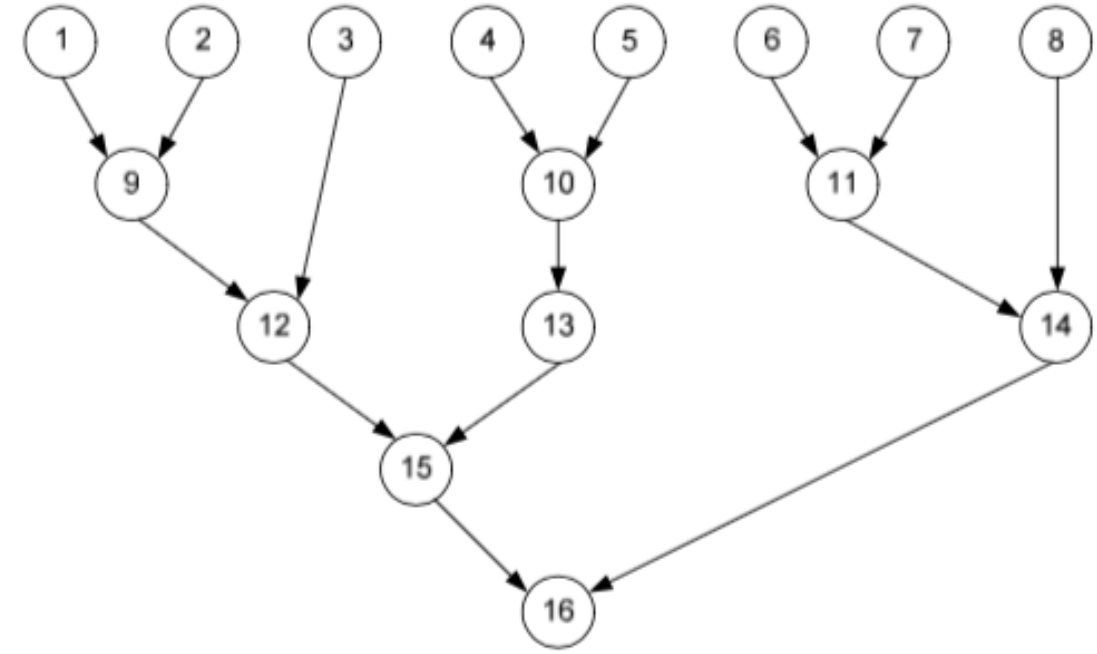
4.Паралельні ОС є відмовостійкими.

Вихідні дані:

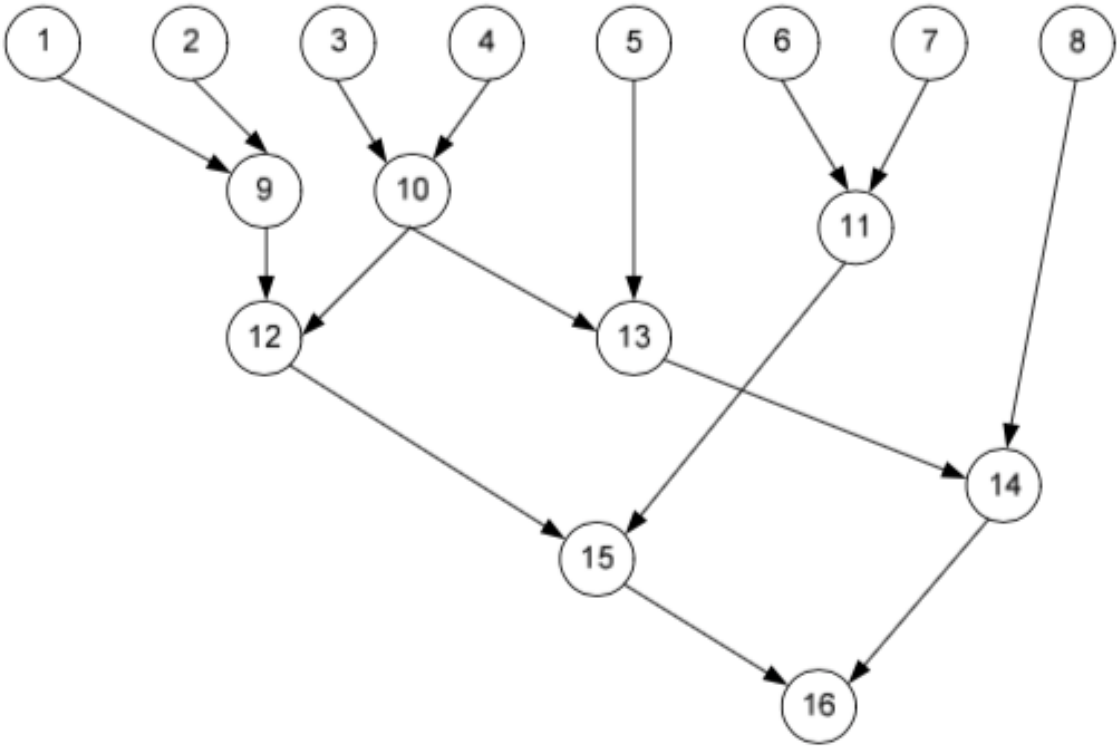
а) граф обчислювальної задачі (рис. 4.6, 4.7).

Вихідні дані визначають за останньою цифрою залікової

книжки. Остання цифра залікової книжки визначає граф обчислювальної задачі. Цифрам 0, 1 відповідає граф на рис. 4.6, цифрам 2, 3 відповідає граф на рис. 4.7, цифрам 4, 5 відповідає граф на рис. 4.8, цифрам 6, 7 відповідає граф на рис. 4.9, цифрам 8, 9 відповідає граф на рис. 4.10.

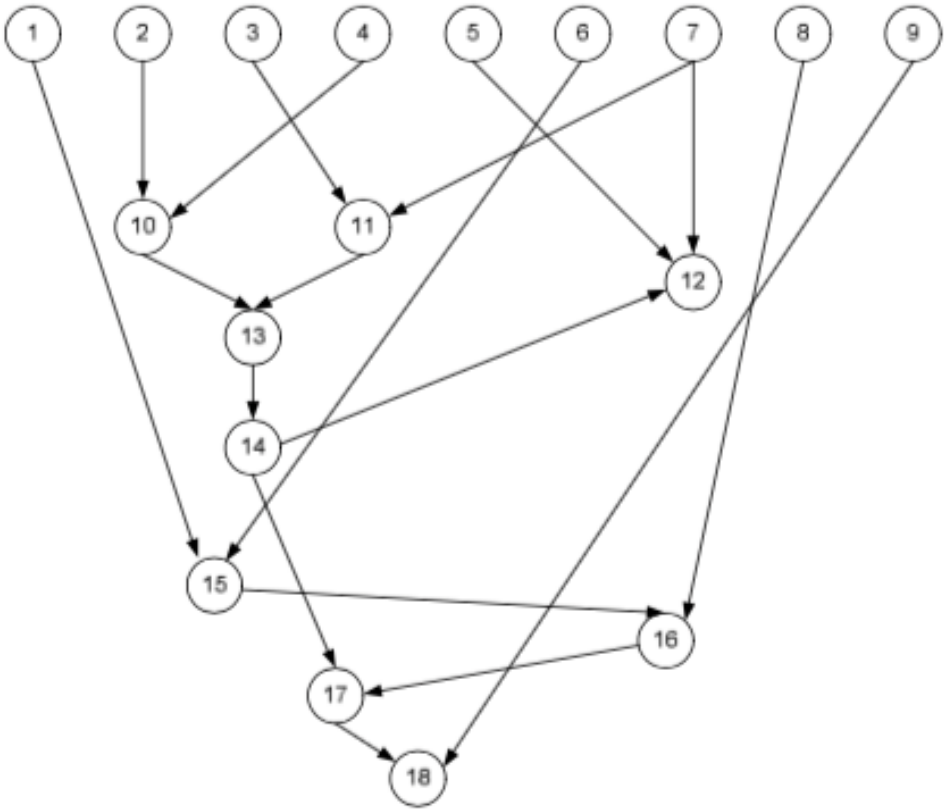


а

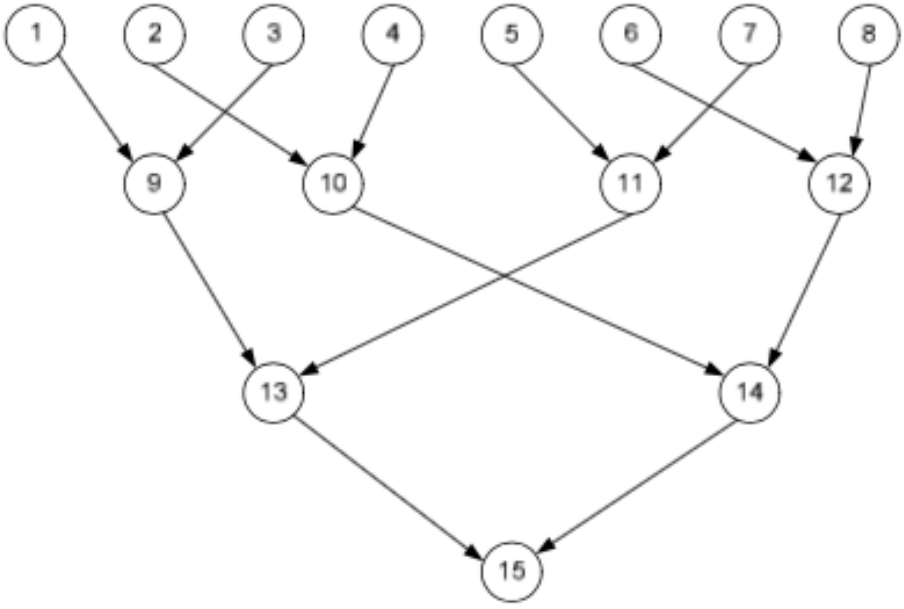


б

Рисунок 7.6 – Граф обчислювальної задачі: а – остання цифра залікової книжки 0; б – остання цифра залікової книжки 1

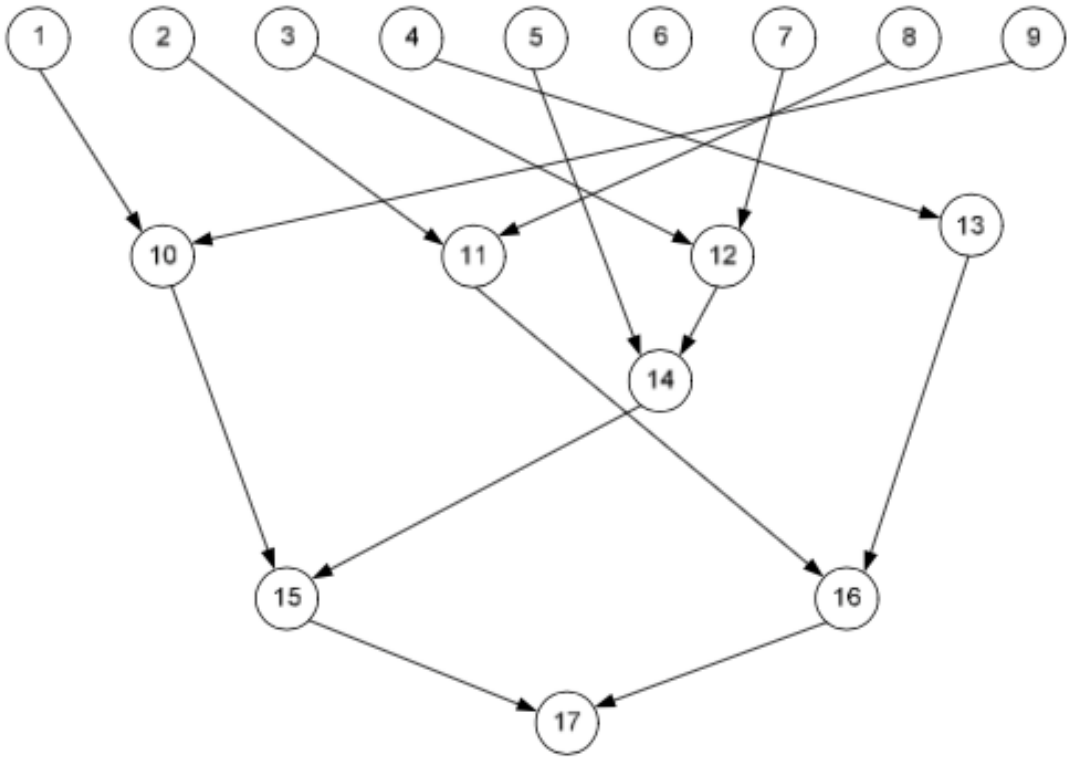


а

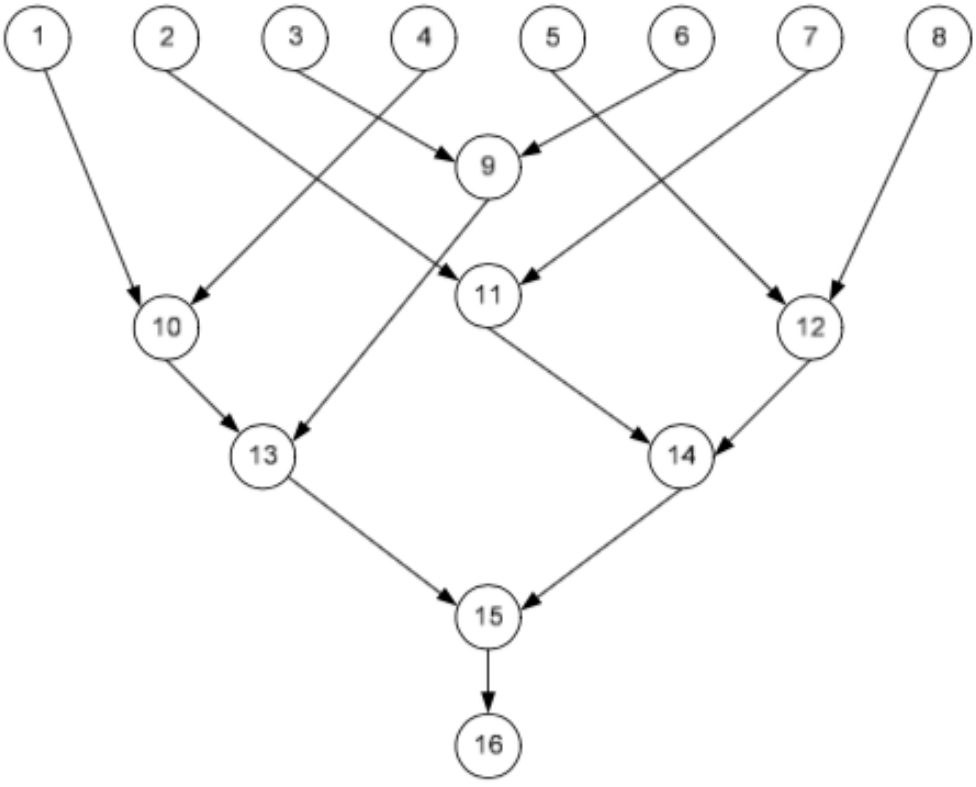


б

Рисунок 7.7 – Граф обчислювальної задачі: а – остання цифра залікової книжки 2; б – остання цифра залікової книжки 3

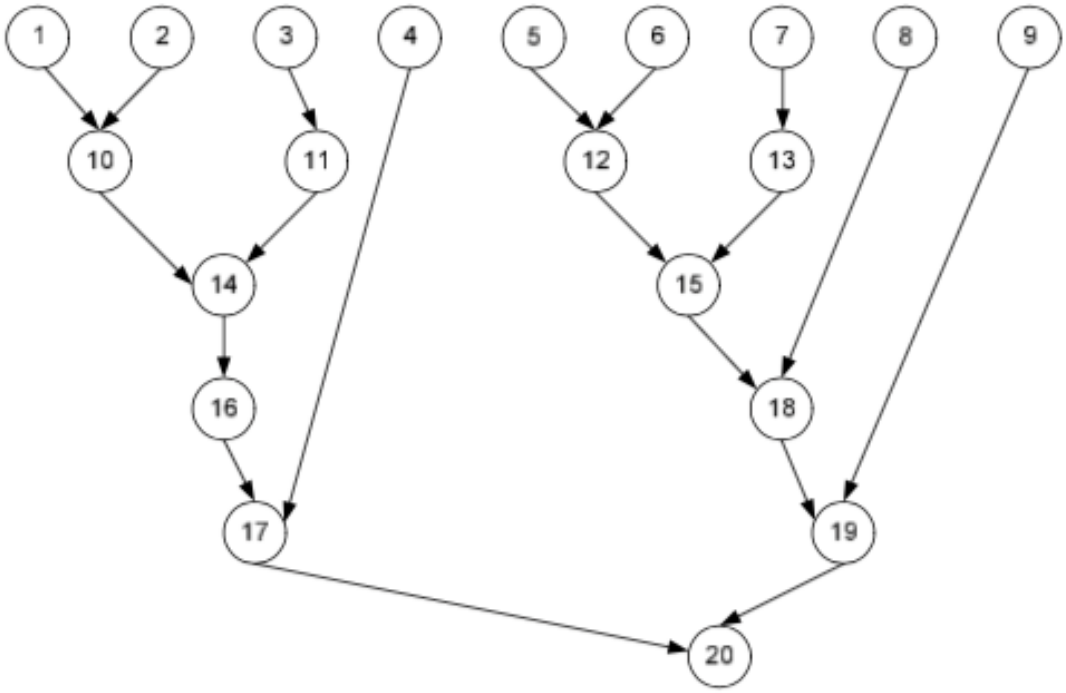


а

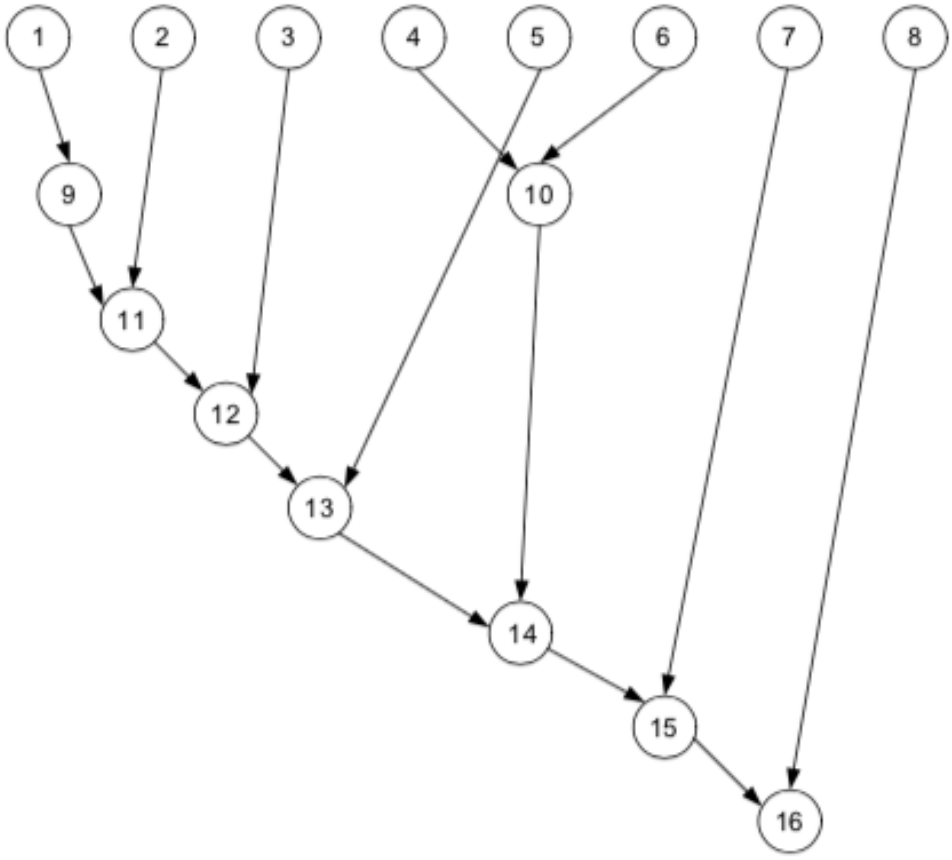


б

Рисунок 7.8 – Граф обчислювальної задачі: а – остання цифра залікової книжки 4; б – остання цифра залікової книжки 5

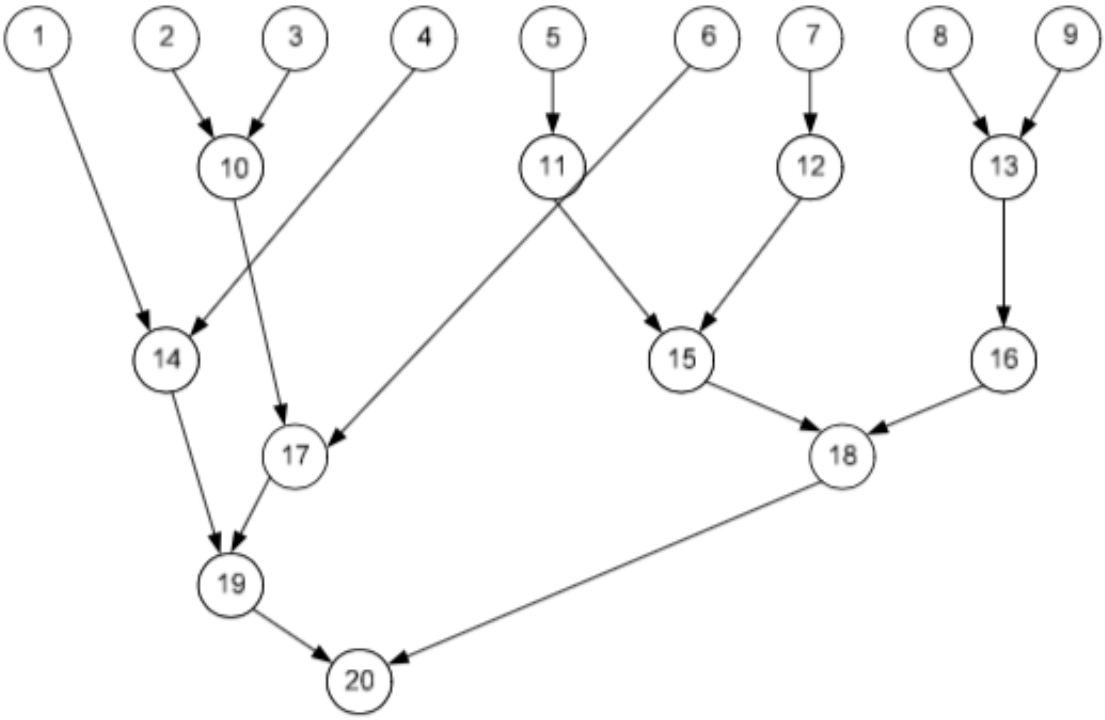


а

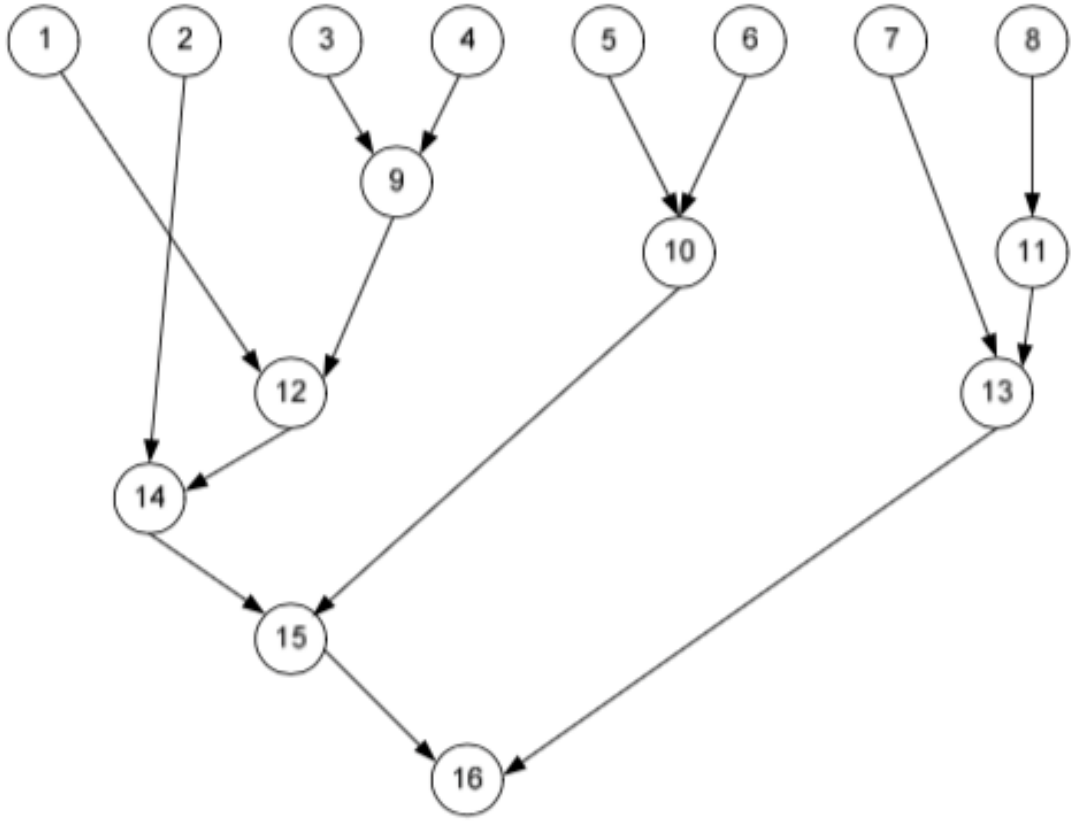


б

Рисунок 7.9 – Граф обчислювальної задачі: а – остання цифра залікової книжки 6; б – остання цифра залікової книжки 7



а



б

Рисунок 4.10 – Граф обчислювальної задачі: а – остання цифра залікової книжки 8; б – остання цифра залікової книжки 9

Хід роботи

1. Вивчити теоретичні відомості.

2. Отримати допуск до роботи.

3. Вибрати за методичними вказівками варіант завдання.

4. Нарисувати граф обчислювальної задачі та визначити оптимальну кількість процесорів, яка потрібна для її вирішення.

Практична робота №5

Тема: Аналіз функціонування багатопроцесорних КС різних топологій

Мета: Аналіз функціонування та ефективності обчислювальних систем різних топологій.

Теоретичні відомості

Найбільш важливим аспектом паралельних ОС з локальною пам’яттю є те, як взаємодіють між собою окремі процесори. Конфігурація системи міжпроцесорного зв’язку істотно впливає на порядок підключення ліній зв’язку, що з’єднують окремі процесори. Організація внутрішніх комунікацій ОС називається топологією.

Залежно від того, чи залишається конфігурація взаємозв’язків незмінною (доки виконується відповідне завдання), розрізняють ОС зі статичною та динамічною топологіями. В статичних ОС структура взаємозв’язків фіксована. В ОС із динамічною топологією в процесі обчислень конфігурація взаємозв’язків за допомогою програмних засобів може бути оперативно змінена.

До статичних топологій відносять такі, де між двома вузлами можливий лише один чи декілька фіксованих шляхів, тобто немає комутуючих пристроїв. З можливих критеріїв класифікації статичних ОС обирають їх розмірність. З цих позицій розрізняють:

• одновимірні топології (лінійний масив);

• двовимірні топології (кільце, зірка, дерево, решітка);

• багатовимірні топології (повнозв’язана топологія);

• гіперкубічну топологію.

У простій лінійній топології вузли системи утворюють одновимірний масив та з’єднані у ланцюг (рис. 5.1).

Рисунок 5.1 – Лінійна топологія ОС

У лінійній топології час передавання повідомлення залежить від відстані між вузлами, а відмова одного з них може призвести до неможливості передавання повідомлення. З цього приводу в лінійних ОС використовують відмовостійкі вузли, які при відмові ізолюють себе від мережі, дозволяючи повідомленню обминути несправний вузол.

Стандартна кільцева топологія (рис. 5.2) являє собою лінійний ланцюг, кінці якого з'єднані між собою. Недолік: додавання чи вилучення вузла потребує демонтажу мережі.

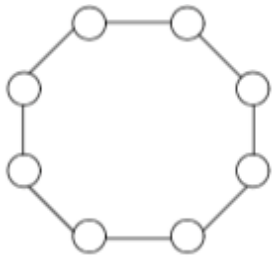


Рисунок 5.2 – Кільцева топологія ОС

Зіркоподібна організація вузлів (рис. 5.3) рідко використовується для об'єднання процесорів багатопроцесорної ОС, але добре працює, коли потік інформації йде від декількох вторинних вузлів, з'єднаних з одним первинним вузлом, наприклад при підключенні терміналів. Загальна пропускна здатність мережі звичайно обмежується швидкодією концентратора. Основна перевага зіркоподібної схеми у тому, що конструктивне виконання вузлів на кінцях cbcntvb може бути дуже простим.

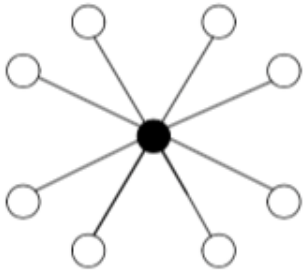


Рисунок 5.3 – Зіркоподібна топологія ОС

У деревоподібній топології (рис. 5.4) система будується за схемою двійкового дерева, де кожний вузол більш високого рівня зв'язаний з двома

вузлами наступного за порядком більш низького рівня. Вузол, що знаходиться на більш високому рівні, називають батьківським, а два підключених до нього нижчерозташованих вузла – дочірніми. В свою чергу, кожний дочірній вузол виступає як батьківський для двох вузлів наступного більш низького рівня. Кожний вузол зв’язаний лише з двома дочірніми та одним батьківським.

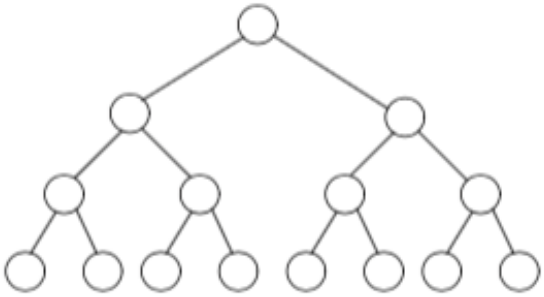


Рисунок 5.4 – Деревоподібна топологія ОС

При великих об’ємах передачі даних між несуміжними вузлами деревоподібна топологія виявляється недостатньо ефективною, оскільки повідомлення повинні проходити через один чи декілька проміжних процесорів. На більш високих рівнях системи ймовірність затору через недостатньо високу пропускну здатність ліній зв’язку вище.

ОС з решітчастою топологією (рис. 5.5) орієнтовані на задачі, пов’язані з обробкою масивів. Їх конфігурація визначається видом та розмірністю масиву.

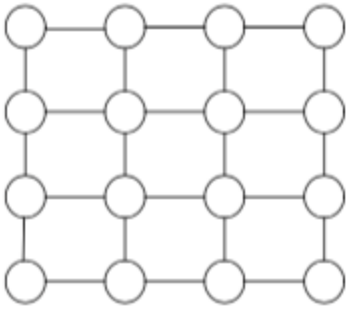


Рисунок 5.5 – Решітчаста топологія ОС

У системах, де кожний процесор з’єднаний з декількома сусідніми процесорами, можна досягти компромісу між складністю системи міжпроцесорного зв’язку, її пропускною здатністю й затримкою обмінів даними. В одновимірних решітках процесорів, що часто називають лінійним масивом,

кожний процесор, крім крайніх, з'єднаний з двома сусідніми. При цьому дані, що пересилаються із процесора-джерела в процесор-приймач, послідовно проходять транзитні процесори, розташовані між джерелом і приймачем. У двовимірній решітці кожний процесор з'єднаний з північним, південним, східним і західним сусідами. Решітки процесорів характеризуються регулярністю, локальністю і простотою маршрутизації міжпроцесорних зв'язків. У тривимірному масиві кожний із процесорів з'єднаний із шістьома сусідами. Перевагою такої системи є мінімальна кількість ліній зв'язку (з кожного процесору виходить не більше двох ліній зв'язку).

У повністюзв'язаній системі кожний процесор має пряме з'єднання з будь-яким іншим процесором.

При об'єднанні паралельних процесорів дуже популярна топологія гіперкуба. Лінія, що з'єднує два вузла (рис. 5.6, а), визначає одновимірний гіперкуб. Квадрат, що утворений чотирма вузлами (рис. 5.6, б) – двовимірний гіперкуб, а куб з 8 вузлів (рис. 5.6, в) – тривимірний гіперкуб та т.ін.

Обмін повідомленнями в гіперкубі базується на двійковому поданні номерів вузлів. Нумерація вузлів робиться так, що для будь- якої пари суміжних вузлів двійкове подання номерів цих вузлів відрізняється лише в одній позиції. Вузли 0010 та 0110 – сусіди, а вузли 0110 та 0101 такими не є.

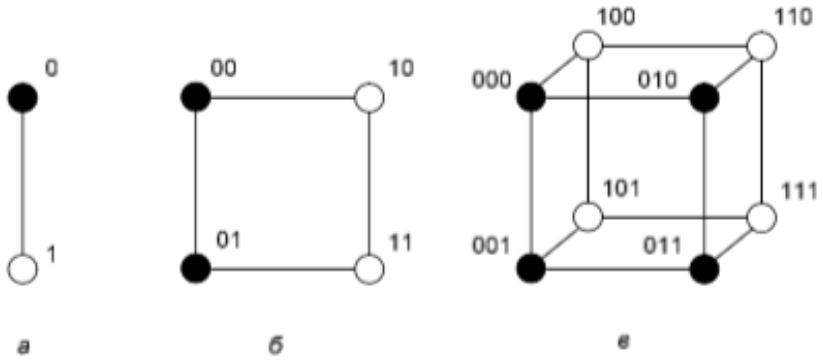


Рисунок 5.6 – Гіперкубова топологія ОС

Така архітектура дає малу кількість зв'язків між процесорами. В ОС з такою архітектурою обчислювальний процес будується таким чином: кожний процесор у вузлі має власну пам'ять та відповідно володіє потужним

обчислювальним ресурсом. Якщо обчислювальної потужності не вистачає, то до вирішення залучають процесори з сусідніх вузлів чи всього кубу. Якщо і цього недостатньо, то залучають процесори, які розташовані у вузлах зовнішнього гіперкубу відносно до даного.

Недоліки окремих типів систем мінімізуються при їх комбінуванні (гібридні системи міжпроцесорного зв'язку). Наприклад, конфігурацію піраміди одержують додаванням зв'язків між процесорами, що належать одному ярусу дерева, відповідно до конфігурації двовимірних решіток. Таким чином, у піраміді об'єднані переваги дерева й решітки.

Структури ОС із кластерами також відносять до гібридних систем зв'язку. У межах кластера процесори з'єднані відповідно до однієї з конфігурацій зв'язків, наприклад, загальною шиною, а кілька кластерів об'єднані у ОС за допомогою іншої конфігурації зв'язків, наприклад, у вигляді решітки або гіперкуба.

У динамічній топології ОС з'єднання вузлів забезпечується електронними ключами, варіюючи установки яких можна змінювати топологію системи. У вузлах динамічних ОС розташовуються комутуючі елементи, а пристрої, що обмінюються повідомленнями (термінали), підключаються до входів та виходів цієї мережі. Як термінали можуть виступати процесори чи процесори та модулі пам'яті. Для таких ОС частіше всього використовується одно- або багатокаскадна комутація на основі матричних комутаторів.

ОС з шинною архітектурою – найбільш простий та дешевий вид динамічних ОС. При одношинній топології (рис. 5.7), усі вузли підключені до однієї шини, що сумісно використовується. В кожний момент часу обмін повідомленнями може вести лише одна пара вузлів, тобто на період передавання повідомлення шину можна розглядати як мережу, що складається з двох вузлів.

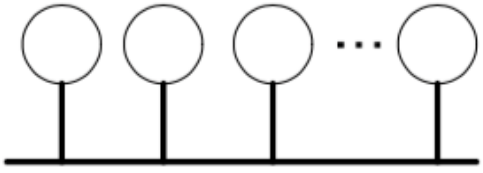


Рисунок 5.7 – Одношинна топологія ОС

Більш ефективною архітектурою динамічних ОС є система, в якій процесори зв’язані між собою за допомогою матричного комутатора. У цьому випадку в кожний момент часу обмін повідомленнями можуть вести n/2- пар вузлів, де n – кількість вузлів у системі. Недоліком такої архітектури є висока вартість.

Приклад: Побудувати часову діаграму роботи ОС для трипроцесорної ОС топології “Зірка” для заданного на рис. 5.8. дерева обчислювальної задачі:

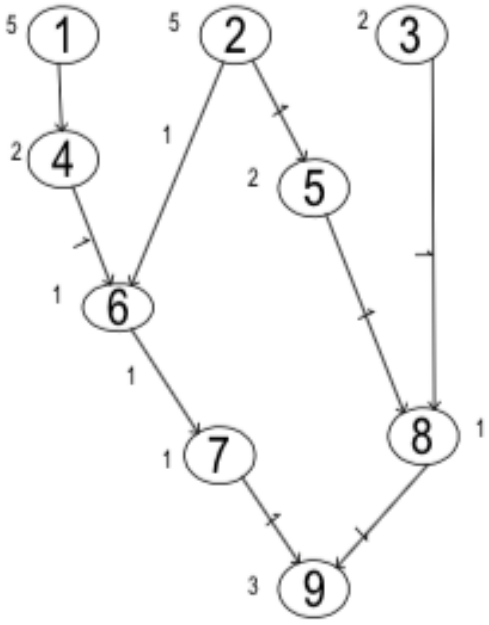


Рисунок 5.8 – Дерево обчислювальної задачі

ОС для трьох процесорів топології “Зірка” подана на рис. 5.9.

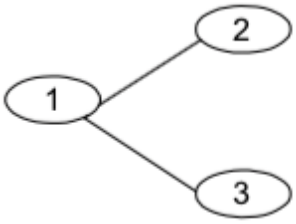


Рисунок 5.9 – Топологія “Зірка”

Часову діаграму роботи ОС для трипроцесорної ОС топології “Зірка” наведено на рис. 5.10.

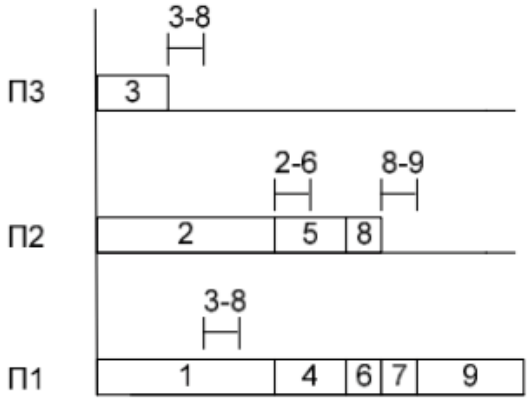


Рисунок 5.10 – Часова діаграма роботи ОС для трьохпроцесорної ОС топології “Зірка”

Таблиця 5.1 - Час виконання операції в ОС

Час / Номер варіанта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	4	2	6	1	3	2	7	1	1	3	6	3	2	3	6	5	1	1	4
1	1	3	3	5	2	1	3	4	3	2	2	2	8	2	1	3	2	5	4
2	1	4	6	5	5	3	2	2	3	3	1	7	5	1	2	6	1	2	3
3	1	4	4	3	2	5	6	2	1	1	3	4	1	4	2	5	7	3	2
4	2	2	4	2	5	3	2	1	5	3	2	7	3	6	2	7	3	1	1
5	3	4	3	8	4	2	6	4	5	3	3	7	4	5	2	4	2	3	1
6	4	6	1	4	3	5	4	3	6	3	2	4	3	4	6	2	5	7	8
7	1	5	2	3	2	7	2	1	6	3	4	5	6	1	2	6	4	4	2
8	7	2	5	2	1	3	2	2	5	3	4	3	4	2	3	7	2	5	2
9	5	2	6	1	5	3	4	4	3	2	1	2	8	7	5	2	1	3	3

Вихідні дані:

а) час виконання операцій (табл. 5.1.);

б) топологія ОС та кількість процесорів (табл. 5.2).

Час виконання операцій визначають останньою цифрою залікової книжки. Типи топологій та кількість процесорів для аналізу ОС визначають за передостанньою цифрою залікової книжки.

Хід роботи

1. Вивчити теоретичні відомості.

2. Отримати допуск до роботи.

3. Вибрати за методичними вказівками варіант завдання.

4. Використовуючи дані з практичної роботи 4 (рис. 4.6. – 4.10), побудувати часові діаграми роботи ОС заданих топологій та кількості процесорів (табл. 5.2) з заданим часом виконання операцій (табл. 5.1).

5. Визначити час виконання обчислювальної задачі для кожної топології, що аналізується.

6. Скласти звіт по лабораторній роботі

7. Зробити висновки по роботі, визначивши, яка з топологій найбільш оптимальна для вирішення конкретної обчиислювальної задачі.

Таблиця 5.2 – Топологія ОС та кількість процесорів

Тип топології / Номер варіанта	Лінійка			Кільце			Решітка		Зірка			Дерево		Гіперкуб	
	2	3	4	2	3	4	4	9	2	3	4	3	7	4	8
0							*	*	*	*	*			*	*
1	*	*	*	*	*	*						*	*		
2									*	*	*	*	*	*	*
3	*	*	*						*	*	*	*	*		
4	*	*	*	*	*	*	*	*							
5				*	*	*	*	*				*	*		
6	*	*	*				*	*						*	*
7				*	*	*			*	*	*			*	*
8	*	*	*						*	*	*			*	*
9				*	*	*	*	*				*	*		

Практична робота №6

Тема: Моделювання часових характеристик комп'ютерних систем

Мета: Вивчення методів оцінки трудомісткості алгоритмів.

Теоретичні відомості

Приклад: Реалізацію алгоритму можна подати у вигляді направленого графа, вершини якого відповідають операторам. Ребра графа відмічаються імовірностями переходів від і-ї вершини до j-ї вершини. Граф наведено на рис.6.1.

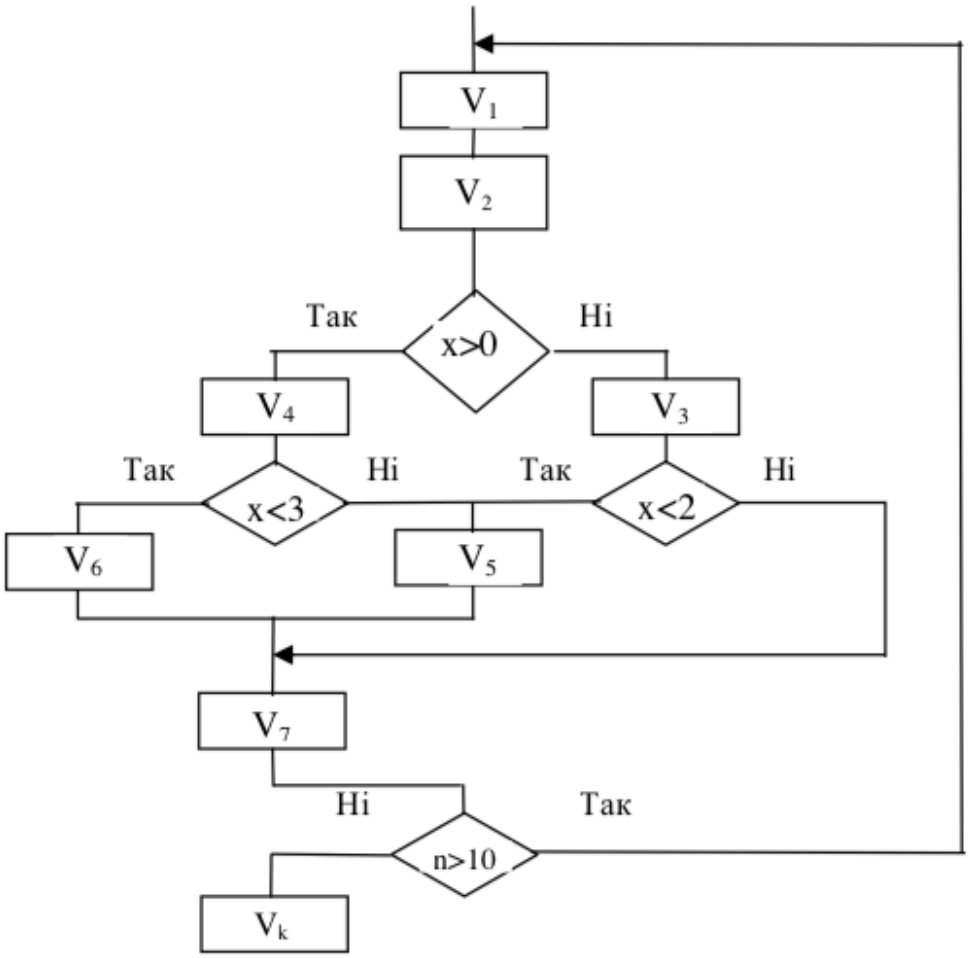


Рисунок 6.1 – Направлений граф алгоритму

Для цього графа матрицю ймовірностей переходу задано в табл. 1, елемент Pij якої визначає ймовірність переходу із стану і в стан j.

Таблиця 6.1 - Матриця ймовірнісних переходів

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_k
V_1		1						
V_2			0,25	0,75				
V_3					0,5		0,5	
V_4					0,2	0,3		
V_5							1	
V_6							1	
V_7	0,9							0,1

Для простоти припустимо, що всі оператори алгоритму – основні та ki =1 для всіх i=1,...,k. На підставі табл. 1 та формули (4) складаємо систему з семи лінійних алгебраїчних рівнянь

$$\left|\begin{array}{r} -N_1 + 0{,}9\,N_7 = -1, \\ N_1 - N_2 = 0, \\ 0{,}25\,N_2 - N_3 = 0, \\ 0{,}75\,N_2 - N_4 = 0, \\ 0{,}5\,N_3 + 0{,}2\,N_4 - N_5 = 0, \\ 0{,}8\,N_4 - N_6 = 0, \\ 0{,}5\,N_3 + N_5 + N_6 - N_7 = 0. \end{array}\right.$$

Розв’язуючи систему знаходимо значення N_1,..., N_{k-1}. Підставляючи отримані значення у формулу, отримаємо:

$$\theta = \sum_{i=1}^{l} k_i N_i = 39{,}75.$$

Якщо при виконанні будь-яких операторів відбувається звернення до файлів, то необхідно визначити ще величини N та θ. Визначивши таким чином ці величини, можна визначити середню трудомісткість етапу рахування.

Вихідні дані:

а) схема алгоритму; б) k_i – кількість операцій, що складають V_{ai} оператор (табл. 6.2); в) L_i – середня кількість інформації, що передається при виконанні V_i оператора звернення до файлу, де m – номер файлу, до якого відбувається звертання (табл. 6.3); г) області зміни параметрів X_i та N_i (табл. 6.4).

Вихідні дані визначають за двома останніми цифрами залікової книжки. Остання цифра залікової книжки визначає область зміни параметрів. Передостання цифра залікової книжки визначає значення k_i та L_i.

Таблиця 6.2 – Число операцій, що складають V_{ai} оператор (k_i)

Кількість операторів V_a	Номер варіанта									
	0	1	2	3	4	5	6	7	8	9
V_{a1}	20	20	50	30	60	20	40	80	30	10
V_{a2}	30	30	40	10	60	100	20	40	60	80
V_{a3}	50	30	20	30	40	60	30	20	100	200
V_{a4}	20	30	50	20	30	30	10	80	90	35
V_{a5}	50	50	30	20	10	50	30	70	20	20
V_{a6}	30	20	10	30	100	30	20	60	70	45
V_{a7}	100	10	20	50	40	20	100	30	40	50
V_{a8}	20	40	100	100	20	40	50	300	200	100

Таблиця 6.3 – Середня кількість інформації, що передається при виконанні Vbi оператора звернення (Li)

Кількість інформації V_b	Номер варіанта									
	0	1	2	3	4	5	6	7	8	9
V_{b1}	500	700	800	250	900	250	800	300	500	400
V_{b2}	250	800	100	500	250	1000	100	200	400	300
V_{b3}	120	500	250	150	100	700	500	250	800	500
V_{b4}	800	100	150	1000	700	250	900	200	100	200
V_{b5}	100	600	800	200	500	1000	250	800	200	700
V_{b6}	600	900	700	100	400	400	400	500	900	300
V_{b7}	900	600	900	400	800	900	100	100	600	900
V_{b8}	400	700	600	200	900	400	600	400	400	100

Таблиця 4 – Області зміни параметрів Xi та Ni

Пара-метри	Номер варіанта									
	0	1	2	3	4	5	6	7	8	9
X_1	-1,+3	-2,+3	-2,+3	0,+4	-2,+2	0,+5	0,+6	-1,+4	1,+7	-3,+1
X_2	-1,+1	-1,+4	-2,+2	-3,+1	-3,+2	-2,+4	0,+5	-1,+3	-2,+4	-2,+3
K_1	10	20	30	10	50	10	20	10	20	25
K_2	20	10	15	20	40	20	10	20	10	10
K_3	10	30	10	30	10	30	10	20	10	20

Ni – середнє число попадань обчислювального процесу у стан Si

Хід роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Використовуючи дані, визначити:
   а) середню кількість операцій, яка виконується за один прогін алгоритму;
   б) середню кількість звернень до кожного з файлів;
   в) середню кількість інформації, яка передається при одному звертанні до файлу;
   г) середню трудомісткість етапу рахування.
5. Скласти звіт по лабораторній роботі.
6. Зробити висновки по роботі.
7. Відповісти на запитання для самоперевірки.

Примітка. Для зменшення розмірності системи лінійних рівнянь доцільно об'єднувати послідовні ланцюжки операторів в один узагальнений оператор.