МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України «Київський політехнічний інститут»

Методичні вказівки до виконання лабораторних робіт з курсу КОМП'ЮТЕРНІ СИСТЕМИ

для студентів напрямку підготовки 6.050102
"Комп`ютерна інженерія"
денної та заочної форм навчання

Рекомендовано
кафедрою обчислювальної техніки
Протокол № від 2012 р.
Завідувач кафедри
_____Луцький Г.М.
(підпис)

Київ 2012

ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ

Сучасні обчислювальні системи являють собою високопродуктивні, відмовостійкі засоби обчислювальної техніки. У зв'язку з безперервним розширенням сфери застосування обчислювальних систем і необхідністю вирішення все більш складних задач у промисловості постійно зростають вимоги до їх продуктивності та надійності.

Так, наприклад, у цивільній авіації обчислювальні системи використовують в автоматизованих системах керування повітряним рухом, експлуатаційним авіапідприємством, матеріально-технічним забезпеченням, в системах резервування і продажу авіаквитків, бортових пілотажно-навігаційних комплексах, системах автоматизованої обробки польотної інформації, авіаційних тренажерах тощо.

Для закріплення практичних навичок, отриманих при вивченні курсу "Комп'ютерні системи", передбачено виконання лабораторних робіт. Лабораторні роботи супроводжуються складанням звіту і виконуються в лабораторії кафедри обчислювальної техніки.

Перш ніж розпочати виконання лабораторної роботи, необхідно ознайомитись з теоретичними відомостями. Звіт по лабораторній роботі повинен бути оформлений на окремих аркушах, містити титульний аркуш, на якому має бути вказана тема лабораторної роботи, назва дисципліни, факультет, група, прізвище, ім'я, по батькові студента, варіант лабораторної роботи, мета, загальні теоретичні відомості, хід виконання роботи, схеми та рисунки, висновки.

Лабораторна робота 1

МОДЕЛІ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи: вивчення моделей обчислювальних процесів.

Загальні теоретичні відомості

Перед тим, як спроектувати обчислювальну систему, аналізують, які функції буде виконувати дана система, які прикладні задачі вона буде вирішувати. Після визначення кола задач, що вирішуються, оцінюють трудомісткість алгоритмів. Поняття "трудомісткість алгоритмів" використовують для оцінки потреб у часі, пов'язаних з періодом роботи сукупності пристроїв обчислювальних систем (ОС).

Трудомісткість алгоритмів – це кількість обчислювальної роботи, яка необхідна для реалізації цих алгоритмів на обчислювальній системі. Трудомісткість оцінюється кількістю операцій, що виконуються з метою обробки вводу та виводу інформації в процесі розв'язування задачі.

Інформація, що стосується задачі, поділяється на програму та дані. Останні містять в собі вихідні дані, проміжні величини та кінцеві результати. Дані можуть бути розташовані як в оперативній, так і в зовнішній пам'яті.

Оперативна та зовнішня пам'ять мають відмінності у методах доступу, швидкісних характеристиках. Затрати часу на звернення до оперативної пам'яті пропорційні кількості інформації, що читається або записується. Витрати часу на звернення до зовнішньої пам'яті в основному визначаються часом пошуку області даних та в меншому ступені залежать від кількості інформації, що передається. Тому для зменшення витрат часу на обмін інформації з зовнішньою пам'яттю за кожне звернення доцільно передавати достатньо велику кількість інформації. Операція звернення до зовнішньої пам'яті розглядається зазвичай як операція вводу-виводу.

Складність обчислень оцінюється трудомісткістю, яка, в свою чергу, визначається кількістю операцій, що виконуються з метою обробки, вводу та виводу інформації в процесі розв'язку задачі. Трудомісткість алгоритму – величина ймовірнісна, оскільки число

операторів, що виконуються при різних вхідних даних, може бути різною величиною. Саме тому повна характеристика трудомісткості передбачає опис кількості операцій, що виконуються за одну реалізацію алгоритму, випадковими величинами.

Використання статичних методів визначення кількості операцій – складний та довготривалий процес, тому трудомісткість алгоритмів звичайно описується методами математичної статистики, наприклад, за допомогою математичного опису числа операцій, що виконуються.

При вирішенні задач аналізу та синтезу ОС виникає необхідність в описі властивостей обчислювальних процесів. Обчислювальний процес подають у вигляді моделі, яка несе в собі інформацію тільки про властивості самого процесу. З урахуванням відомостей про трудомісткість алгоритму можна сформулювати такі вимоги до моделі, яка повинна:

1) визначати порядок проходження алгоритмів запитів на кожен з видів обслуговування – рахування та ввод-вивід інформації;
2) визначати трудомісткість обслуговування запитів – кількість операцій, яку повинен виконати процесор при обслуговуванні запиту на рахування, та кількість символів інформації, що вводиться або виводиться;
3) відповідати реальним процесам з точністю до збігу, як найменше, математичних очікувань їх однойменних характеристик.

На початковому етапі побудови моделі необхідно оцінити трудомісткість алгоритму [4].

При оцінці трудомісткості обчислювальний процес можна подати у вигляді послідовності операторів, при цьому розрізнюють три типи операторів: функціональні, переходу та оператори звернення до файлів. Функціональні оператори задають сукупність обчислювальних операцій. Оператори переходу задають правила вибору одного з можливих шляхів розвитку обчислювального процесу. Оператори звернення до файлів відображують процес обміну інформацією із зовнішніми пристроями. Для оцінки трудомісткості алгоритму необхідно розбити множину операторів на два класи: основні оператори, які об’єднують в собі функціональні оператори та оператори переходу (позначимо V_a) $S_O=\{V_{ai},...,V_{am}\}$; оператори звернення до файлів (позначимо V_b) $S_H=\{V_{bi},...,V_{bm}\}$.

Трудомісткість алгоритму можна охарактеризувати такою сукупністю параметрів:

Q – середня кількість процесорних операцій, які виконуються за одну реалізацію алгоритму; $N_1,...,N_h$ - середній час звернення до файлів $F_1,...,F_h$; $\theta_1,...,\theta_h$ – середня кількість байтів, що передається за одне звернення до файлів $F_1,...,F_h$ відповідно. Сукупність операторів та зв'язків між ними бачимо на графі алгоритму (рис. 1), в якому індексами *i* та *j* відмічена ймовірність переходу P_{ij} з вершини V_i до вершини V_j. Оскільки обчислювальний процес не може призупинитись в довільній, за винятком кінцевої *K*, вершині, то з імовірністю *i* відбудеться перехід до будь-якого оператора алгоритму. З урахуванням цього для будь-якого оператора V_i (*i*=1,2,…,k-1) повинна виконуватись умова P_{ij} =1. Імовірності P_{ij} залежать від ймовірності виконання умови, що перевіряється оператором *i* з метою вибору шляху переходу.

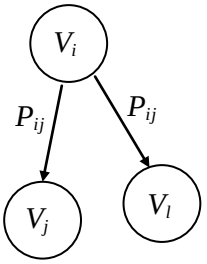


Рис. 1

Наприклад, нехай оператор *i* породжує перехід до оператора *j* при від'ємному значенні деякої змінної *X* та до оператора *l* при додатному значенні *X* (рис. 2). Якщо відомо, що величина *X* рівномірно розподілена в діапазоні (-1, 3), то з імовірністю 0,25 її знак є від'ємним та з імовірністю 0,75 – додатним. З цього витікає, що перехід до оператора *j* відбувається з ймовірністю P_{ij} =0,25 та перехід до оператора *l* – з імовірністю P_{il}=0,75.

Для операторів циклу ймовірність переходу визначається кількістю циклів. Наприклад, оператор циклу повинен виконуватися 9 разів, в цьому випадку ймовірність виходу з цикла P_{ij}=0,1, а імовірність виконання циклу P_{ij}=0,9. Якщо за оператором *i* обов'язково виконується оператор *j*, то P_{ij}=1.

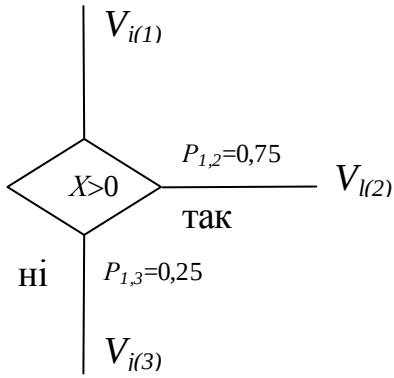


Рис. 2

Нехай $N_1,...,N_{k-1}$ – середнє число звернень до операторів $V_1,...,V_{k-1}$ за один прогін алгоритму, k_i – число операцій, які складають i-оператор. У такому випадку характеристики трудомісткості можуть бути обчислені таким чином.

Визначаємо середнє число процесорних операцій, що виконуються при одному прогоні алгоритму:

$$\theta \le \sum_{V_{oi}} n_i k_i, \qquad (1)$$

Після цього визначаємо середнє число звернень до файлу F_h:

$$N_h = \sum_{V_i} n_i (h = 1,...,H) \qquad (2)$$

та середню кількість інформації, що передається при одному зверненні до файла F_h:

$$\theta_h = \frac{1}{N_h} \sum_{V_i}^{k=1} n_i l_i (h = 1,...,H), \qquad (3)$$

де l_i – середня кількість інформації, яка передається при виконанні оператора i.

Значення $N_1,...,N_{k-1}$ визначаються розв'язком системи лінійних алгебраїчних рівнянь, канонічний запис яких має вигляд:

$$
\begin{aligned}
&(P_{11}\text{-}1)n_1+P_{21}n_2+\ldots+P_{(k-1)1}n_{k-1}= -1\\
&P_{12}n_1+(P_{22}-1)n_2+\ldots+P_{(k-1)2}n_{k-1}= 0 \qquad (4)\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&P_{1(k-1)}n_1+P_{2(k-1)}n_2+\ldots+(P_{(k-1)(k-1)}-1)n_{k-1}= 0.
\end{aligned}
$$

У виразі (1) підсумовування виконується для всіх вершин, які належать до основних операторів, у виразах (2), (3) – для всіх вершин, що відображають звернення до даного файлу.

При формуванні коефіцієнтів системи (4) необхідно звернути увагу на те, що коефіцієнти P_{ij} записуються не по рядках, а по стовпцях, тобто, наприклад, на місці другого елемента першого рядка стоїть елемент з індексом 21, а не коефіцієнт з індексом 12 матриці переходів.

Визначивши за формулами (1), (2), (3) величини θ, N_h , θ_h, можна визначити середню трудомісткість етапу рахування, на основі якої виконується оцінка оптимальної швидкодії процесора.

Середня трудомісткість етапу рахування визначається за формулою:

$$\theta_0 = \frac{\theta}{N}, \qquad (5)$$

де N – сума середнього числа N_i звернень до основних операторів(S_o), тобто:

$$N = \sum_{V_{0i}} n_i. \qquad (6)$$

Вихідні дані:

- схема алгоритму (рис. 3, 4, 5, 6).

Вихідні дані визначаються останньою цифрою залікової книжки. Вона визначає блок-схему алгоритму. При цьому цифрам 0, 1, 2 відповідає блок-схема, наведена на рис. 3, цифрам 3, 4, 5 – блок-схема, наведена на рис. 4, цифрам 6, 7, 8 – блок-схема, наведена на рис. 5, а цифрі 9 – блок-схема, наведена на рис. 6.

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Занести до звіту блок-схему, яка передбачена завданням.
5. Скласти звіт по лабораторній роботі.
6. Зробити висновки по роботі.
7. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Дати визначення поняттю “трудомісткість алгоритмів”.
2. Для чого використовують поняття “трудомісткість алгоритму”?
3. У вигляді чого можна подати обчислювальний процес?
4. Сформулювати вимоги до моделі обчислювального процесу.
5. Які розрізняють типи операторів алгоритму?
6. Якими параметрами можна охарактеризувати трудомісткість алгоритму?

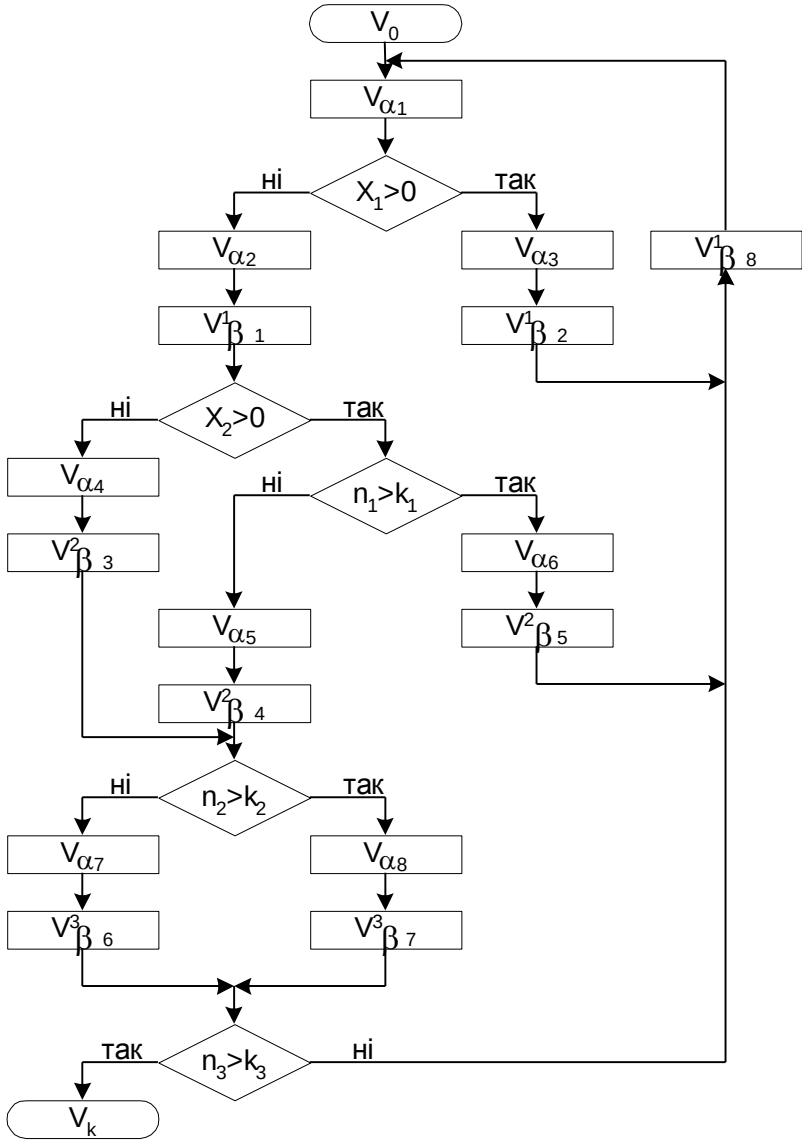

V0
Vα1
ні
X1>0
так
Vα2
Vα3
V1β 8
V1β 1
V1β 2
ні
X2>0
так
Vα4
ні
n1>k1
так
V2β 3
Vα6
Vα5
V2β 5
V2β 4
ні
n2>k2
так
Vα7
Vα8
V3β 6
V3β 7
так
n3>k3
ні
Vk

Рис. 3

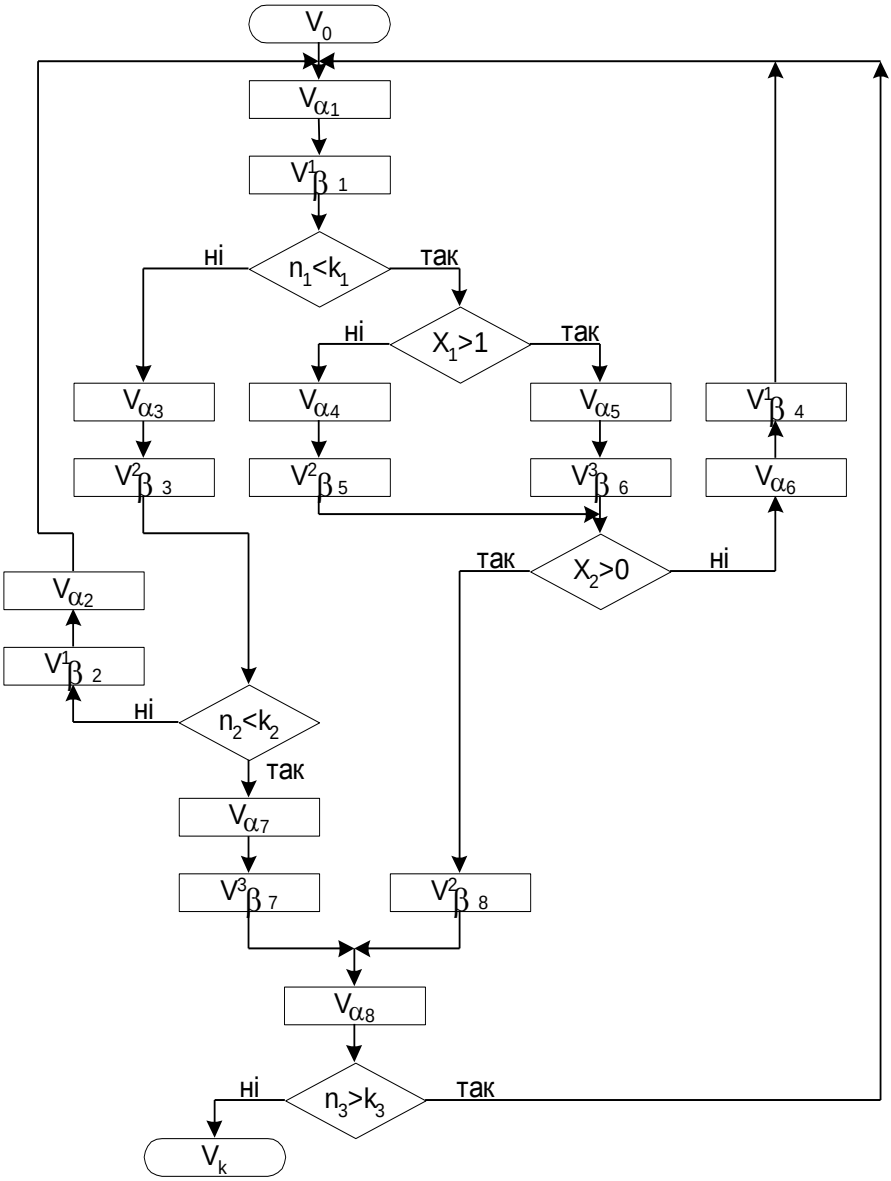

V_0
$V_{\alpha 1}$
$V^1_{\beta 1}$
$n_1<k_1$
ні
так
$X_1>1$
ні
так
$V_{\alpha 3}$
$V_{\alpha 4}$
$V_{\alpha 5}$
$V^1_{\beta 4}$
$V^2_{\beta 3}$
$V^2_{\beta 5}$
$V^3_{\beta 6}$
$V_{\alpha 6}$
$X_2>0$
так
ні
$V_{\alpha 2}$
$V^1_{\beta 2}$
$n_2<k_2$
ні
так
$V_{\alpha 7}$
$V^3_{\beta 7}$
$V^2_{\beta 8}$
$V_{\alpha 8}$
$n_3>k_3$
ні
так
V_k

Рис. 4

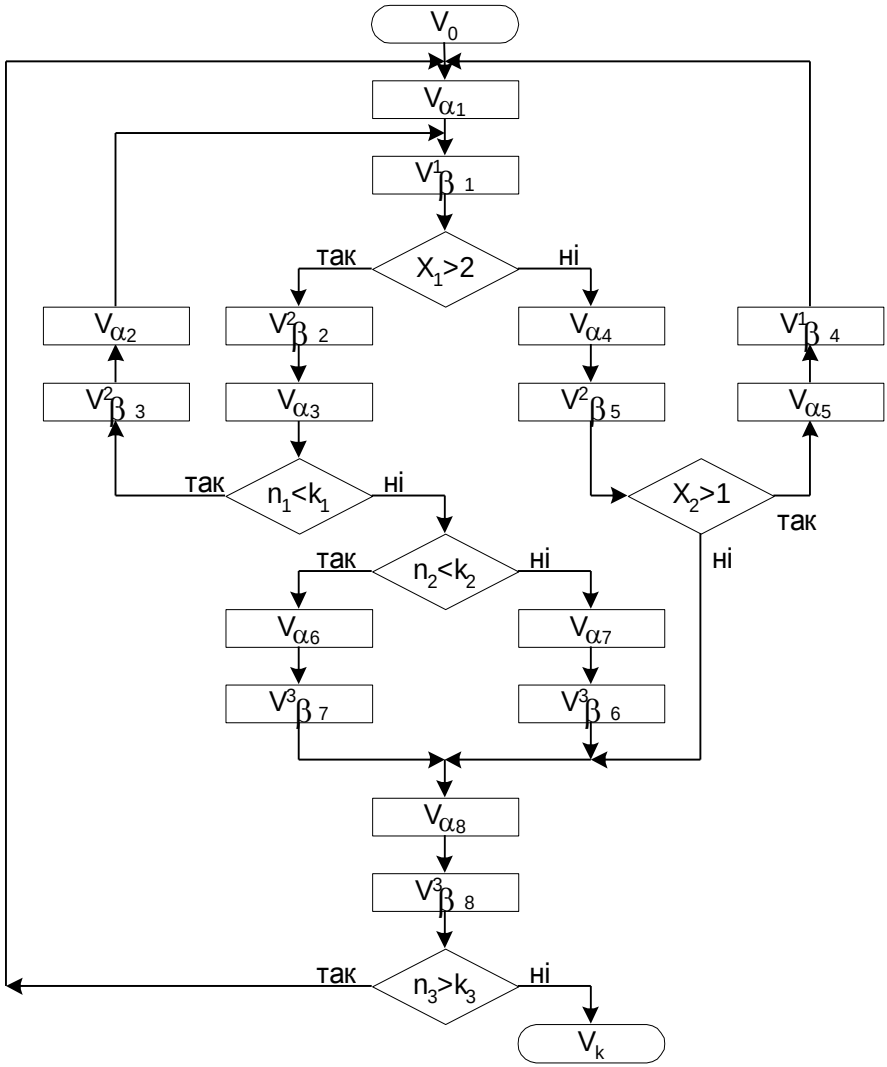

V_0
$V_{\alpha 1}$
$V^1\beta_1$
так
$X_1>2$
ні
$V_{\alpha 2}$
$V^2\beta_2$
$V_{\alpha 4}$
$V^1\beta_4$
$V^2\beta_3$
$V_{\alpha 3}$
$V^2\beta_5$
$V_{\alpha 5}$
так
$n_1<k_1$
ні
$X_2>1$
так
так
$n_2<k_2$
ні
ні
$V_{\alpha 6}$
$V_{\alpha 7}$
$V^3\beta_7$
$V^3\beta_6$
$V_{\alpha 8}$
$V^3\beta_8$
так
$n_3>k_3$
ні
V_k

Рис. 5

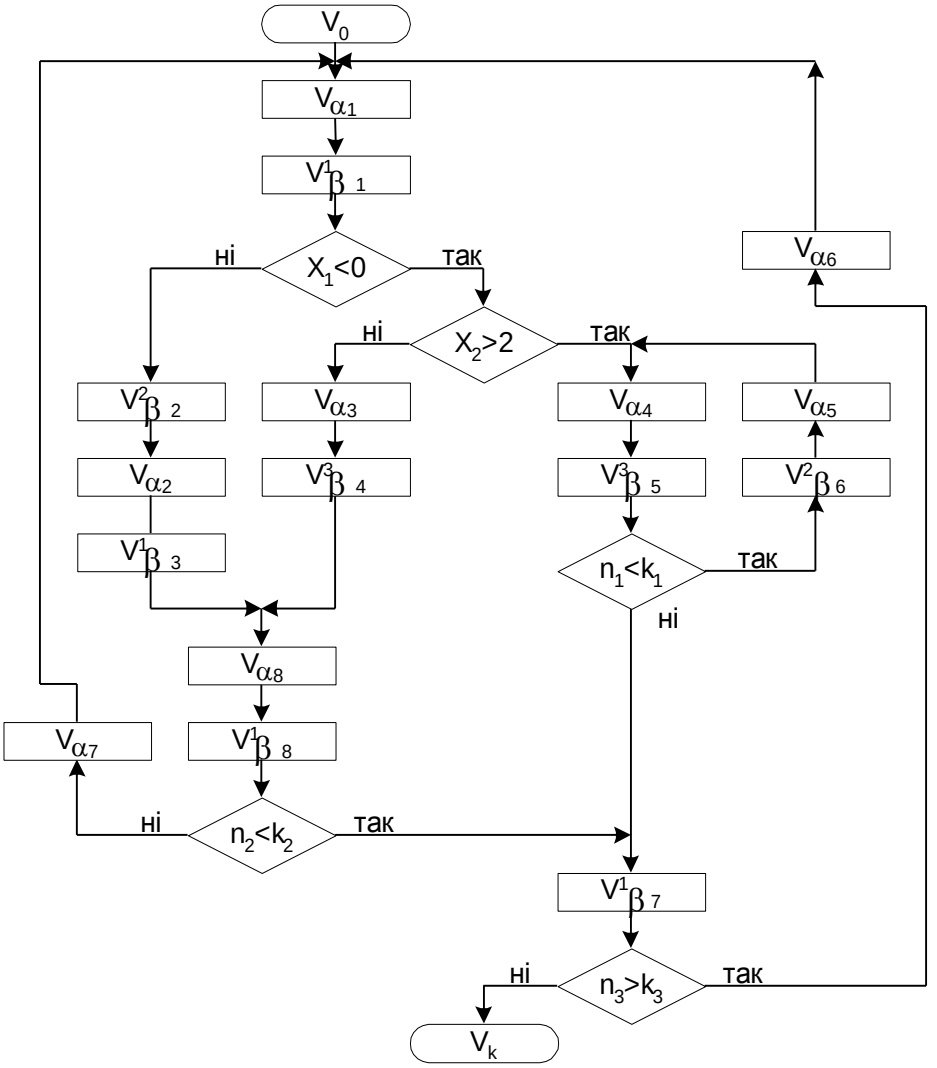

V_0
$V_{\alpha 1}$
$V^1_{\beta 1}$
$X_1<0$
ні
так
$X_2>2$
ні
так
$V^2_{\beta 2}$
$V_{\alpha 3}$
$V_{\alpha 4}$
$V_{\alpha 5}$
$V_{\alpha 6}$
$V_{\alpha 2}$
$V^3_{\beta 4}$
$V^3_{\beta 5}$
$V^2_{\beta 6}$
$V^1_{\beta 3}$
$n_1<k_1$
так
ні
$V_{\alpha 8}$
$V_{\alpha 7}$
$V^1_{\beta 8}$
ні
$n_2<k_2$
так
$V^1_{\beta 7}$
ні
$n_3>k_3$
так
V_k

Рис. 6

Лабораторна робота 2

МОДЕЛЮВАННЯ ЧАСОВИХ ХАРАКТЕРИСТИК ОБЧИСЛЮВАЛЬНИХ СИСТЕМ ТА МЕРЕЖ

Мета роботи: вивчення методів оцінки трудомісткості алгоритмів.

Загальні теоретичні відомості

Приклад: Реалізацію алгоритму можна подати у вигляді направленого графа, вершини якого відповідають операторам [3]. Ребра графа відмічаються імовірностями переходів від *i*-ї вершини до *j*-ї вершини. Граф наведено на рис. 7.

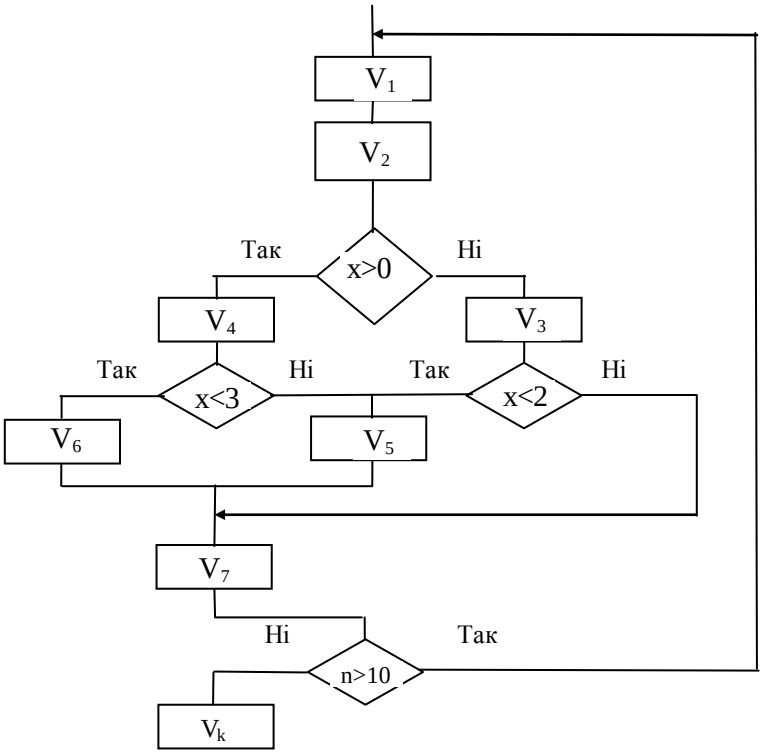


Рис. 7

Для цього графа матрицю ймовірностей переходу задано в табл. 1, елемент P_{ij} якої визначає ймовірність переходу із стану i в стан j.

Матриця ймовірнісних переходів

Таблиця 1

	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 k_i =1 для всіх i=1,…,k. На підставі табл. 1 та формули (4) складаємо систему з семи лінійних алгебраїчних рівнянь

$$\left|\begin{array}{r} -N_1 + 0{,}9\,N_7 = -1, \\ N_1 - N_2 = 0, \\ 0{,}25\,N_2 - N_3 = 0, \\ 0{,}75\,N_2 - N_4 = 0, \\ 0{,}5\,N_3 + 0{,}2\,N_4 - N_5 = 0, \\ 0{,}8\,N_4 - N_6 = 0, \\ 0{,}5\,N_3 + N_5 + N_6 - N_7 = 0. \end{array}\right. \qquad (7)$$

Розв’язуючи систему (7) знаходимо значення N_1 ,…, N_{k-1} . Підставляючи отримані значення у формулу (1), 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 (з лабораторної роботи 1); б) k_i – кількість операцій, що складають V_{ai} оператор (табл. 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. 3); г) області зміни параметрів X_i та N_i (табл. 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.

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

Таблиця 2

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

Середня кількість інформації, що передається при виконанні V_{bi} оператора звернення (L_i)

Таблиця 3

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

Області зміни параметрів X_i та N_i

Таблиця 4

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

N_i – середнє число попадань обчислювального процесу у стан S_i $(1,\ldots, k_i)$.

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Використовуючи дані з лабораторної роботи 1,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.

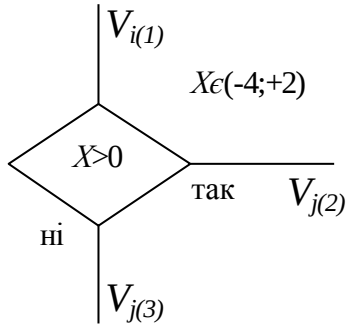
Примітка. Для зменшення розмірності системи лінійних рівнянь доцільно об'єднувати послідовні ланцюжки операторів в один узагальнений оператор.

Запитання для самоперевірки

1. Як розраховується імовірність переходу?
2. За якою формулою визначається середня кількість процесорних операцій, які виконуються за один прогін алгоритму?
3. Як визначити середню кількість звернень до алгоритму?
4. Як визначити середню кількість інформації, яка передається при одному зверненні до файлу?
5. Як визначити середню трудомісткість етапу розрахунку?

Приклади тестових завдань

1. Розрахувати імовірності переходу *Pij*:

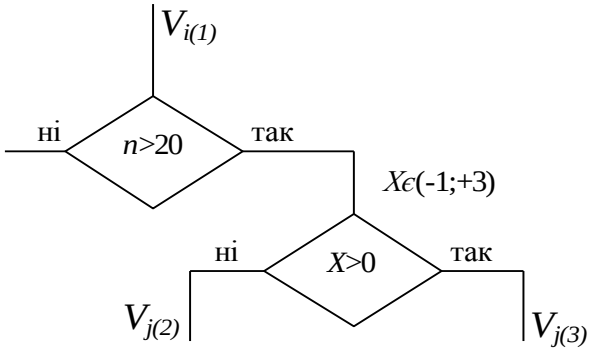


а) $P_{12}=2/3$; $P_{13}=1/3$;

б) $P_{12}=1/3$; $P_{13}=2/3$;

в) $P_{12}=1/6$; $P_{13}=5/6$.

2. Розрахувати імовірність переходу *Vij*:



а) P_{12}=0.019; P_{13}=0.0357;
б) P_{12}=0.25; P_{13}=0.75;
в) P_{12}=0.238; P_{13}=0.714.

Лабораторна робота 3

ВИВЧЕННЯ РОБОТИ БАГАТОПРОЦЕСОРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Мета роботи: закріплення теоретичних знань із структури багатопроцесорних ОС.

Загальні теоретичні відомості

Передумовою для створення обчислювальної системи є мікромініатюризація компонентів схем (інтегральна технологія), підвищення їх надійності, зниження вартості та енергоспоживання. Все це дозволяє об'єднати в одному апаратурному комплексі більше елементів, ніж у звичайних ЕОМ, і одержати збільшення швидкодії не тільки за рахунок підвищення швидкостей окремих елементів і скорочення довжин з'єднань, але й за рахунок ускладнення структури системи.

Багатопроцесорні обчислювальні системи можна розділити на багатопроцесорні обчислювальні системи з роздільним керуванням

(тип М1), з загальним керуванням (тип М2), та матричні обчислювальні системи (тип М3).

На рис. 8 зображена багатопроцесорна обчислювальна система типу М1 із роздільним керуванням. Система вміщує N процесорів, кожний з яких має свій пристрій керування, арифметико-логічний пристрій та пристрій пам'яті. Всі процесори зв'язані між собою через загальну пам'ять і з нею ж зв'язані спільні для всіх процесорів канали зовнішнього обміну. Загальна пам'ять – це частина внутрішньої пам'яті системи. Можливо, що пам'ять в складі кожного процесора відсутня або являє собою надоперативну пам'ять. Тоді відповідно загальне поле пам'яті містить або всі ступені внутрішньої пам'яті, або головну пам'ять і велику пам'ять, або тільки велику пам'ять.

Можливі і такі побудови обчислювальних систем типу М1, в яких загальної внутрішньої пам'яті взагалі немає, а обмін інформації йде через окремі ланцюги. При цьому загальні канали адресуються до всієї внутрішньої пам'яті системи, указуючи як адресу звернення, номер процесора та адресу пам'яті.

Обчислювальні системи типу М1 називають системами з множинним потоком команд і множинним потоком даних (МКМД). Такі системи краще всього пристосовані до використання паралелізму незалежних гілок. Суть цього виду паралелізму полягає в тому, що в програмі розв'язання великої задачі на окремих етапах можна виділити незалежні частини гілки, які можуть виконуватися паралельно (одночасно одна з одною).

Для системи М1 виконання гілки доручається відповідному процесору і можливості обчислювальної системи будуть використані найкращим чином, якщо число незалежних гілок задачі (програми) буде точно дорівнювати або буде кратним числу процесорів у системі.

Окрім того, важливо, щоб усі гілки програми, які входять до одного й того ж ярусу, були б по можливості однакової довжини (трудомісткості) та щоб ця довжина була б достатньо великою.

Кратність ширини ярусів графа кількості процесорів може бути отримана тільки в разі збігу майже неймовірних обставин.

На таку ситуацію можна розраховувати або у спеціалізованій системі, яка побудована для розв'язання певної задачі (класу задач),

або як на результат спеціального перетворення програми розв'язання задачі.

Тому ОС типу М1 з великою кількістю процесорів використовуються для досягнення високої системної продуктивності, тобто продуктивності, яка досягається при розв'язанні сукупності задач користувача.

Основною позитивною якістю даних ОС є потенційно дуже велика взаємодія процесорів. Великий недолік полягає у тому, що загальна пам'ять може буги потрібна одночасно різним процесорам. У таких випадках виникають затримки з доступом до пам'яті: тривалість подібних затримок, яка називається часом конфліктів пам`яті, може зростати зі збільшенням кількості процесорів.

На рис. 9 наведено можливий варіант обчислювальної системи типу М2 із загальним керуванням, яка орієнтована на паралелізм суміжних операцій.

Під загальним керуванням зібрано: пристрій управління, N арифметико-логічних процесорів, кожний з них може мати свою невелику пам`ять, але може і не мати її. В основному процесори працюють над загальною пам`яттю, пристроєм керування якої вважаються інструкції.

Є два принципових варіанти організації роботи такої системи. Перший варіант (тип М2.1.) полягає в тому, що пристрій керування зчитує з пам`яті інструкції "вперед" зі швидкістю, яка набагато перевищує швидкість виконання однієї інструкції одним процесором; прочитавши інструкцію, аналізує, чи є умови для того, щоб почати її виконання, а якщо є, то доручає її виконання будь-якому вільному процесору (але першому, який вивільниться, якщо всі вони були зайняті) та аналізує наступну інструкцію і т. ін.

Перегляд програми "вперед" та виконання окремих її інструкцій може продовжуватись і у тому випадку, коли з`ясовується, що виконання чергової інструкції потрібно відкласти. Інструкції, виконання яких відкладено, накопичуються у буфері пристрою керування.

В другому варіанті (тип М2.2.) пристрій керування зчитує з пам`яті безпосередньо вектори-інструкції (VLIW- структури ОС), один вектор-інструкція містить N компонентів – за числом процесорів у системі. Кожна з них вказує, яку операцію повинен виконувати відповідний процесор.

Для ОС типу М2.1. аналіз програми і організація паралельних обчислень проводиться пристроєм керування безпосередньо на етапі виконання програми (методом інтерпретації), для ОС типу М2.2. – на етапі програмування під час трансляції програми. Зрозуміло, що оптимізація програми методом трансляції може бути виконана більш ефективно, ніж інтерпретація. Окрім того, при виконанні програми в ОС типу М2.1. додатковий час буде витрачатись на аналіз програми.

Недоліком обчислювальної системи типу М2.2. є залежність часу виконання будь-якого вектора-інструкції від тривалості виконання самої тривалої операції в ньому.

Обчислювальні системи типу М2 орієнтовані на паралелізм суміжних операцій. Його суть полягає в наступному. Під час виконання програми зустрічаються ситуації, коли вихідні дані для деякої *i*-ї операції формуються не під час виконання попередньої (*i*-1)-ї операції, а раніше – під час виконання, наприклад (*i*-2)-ї чи (*i*-3)-ї операції. Тоді можна сполучити час виконання *i*-ї операції з виконанням (*i*-2)-ї чи (*i*-3)-ї операції.

Кількість процесорів у системах типу М2 не робиться більше ніж 2 – 6, оскільки глибина паралелізму суміжних операцій звичайно невелика.

Характерною особливістю систем типу М2 є той факт, що, незважаючи на наявність загального для усіх *N* процесорів пристрою керування, усі вони управляються різними інструкціями чи різними компонентами вектора-інструкції.

Системи типу М2 висувають певні вимоги до пам’яті. У першому варіанті вимагається висока швидкодія пам’яті (щоб читати інструкції швидше, ніж йде їх виконання), у другому вимагається широкий формат звернень до пам’яті (щоб за одно звернення прочитати один вектор-інструкцію).

“Висока швидкодія” пам’яті – це час звернення до пам’яті, приблизно в *N* разів менше, ніж час виконання операції у процесорі; “широкий формат” пам’яті – це можливість вибірки орієнтовно *N* слів за одно звертання.

Принцип побудови обчислювальних систем типу М3 наведено на рис 10. Загальний пристрій керування роздає всім *N* процесорам однакові команди для кожного кроку. Всі процесори виконують одну і ту ж операцію – кожен над своїми даними. Системи типу М3 називають матричними системами (або з матричним процесором), а

самі процесори – процесорними елементами матричного процесора.

Число процесорних елементів у матричному процесорі може досягати значної кількості.

Матричні обчислювальні системи орієнтовані на розв'язання задач з природним паралелізмом. Такі задачі зводяться до операцій над багатомірними векторами, матрицями та іншими аналогічними об'єктами. Кожен з цих об'єктів може бути поданий сукупністю чисел: вектора – своїми компонентами, матриць – своїми елементами. Більшість операцій при цьому – сукупності однакових операцій над відповідними парами чисел (елементами) двох аналогічних об'єктів. Наприклад, складання двох векторів полягає в додаванні відповідних компонент векторів. Очевидно, що всі ці операції складання можуть бути виконані паралельно і незалежно одна від одної.

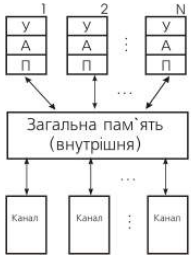


Рис. 8

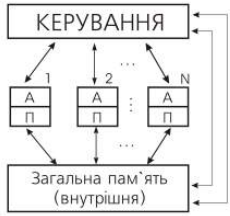


Рис. 9

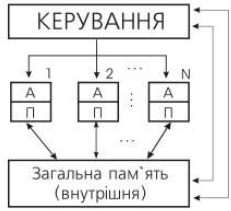


Рис. 10

Вихідні дані:

а) арифметичний вираз (табл. 5);

в) кількість процесорів ОС (табл. 5);

г) значення коефіцієнтів α і β (табл. 5).

Номер варіанта визначається останньою цифрою номера залікової книжки.

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Накреслити дерево заданого арифметичного виразу.
5. Скласти звіт по лабораторній роботі.
6. Зробити висновки по роботі.

Арифметичний вираз, кількість процесорів ОС (N), коефіцієнти α і β

Таблиця 5

Остання цифра номера залікової книжки	N	α	β	Вираз
0	3	2	4	A+B*C+D/E+F(G+H)+K/L
1	3	3	4	A*B*C+D*E+F(G+H)+K+L
2	4	2	3	(A+B)/(C+D)*E*F+G/H+K/L
3	3	2	4	(A+B)/(C+D*E)+F+(G+H)/K*L
4	2	3	4	A/B*(C+D)+E*F*G*H+K/L
5	3	3	5	A+B*C(D+E/F)+G*H/(K+L)
6	4	2	4	(A+B*C+D*E)/(F+G*H+K*L)
7	3	3	4	A*B+C*D+E*F+G*H+K/L
8	3	2	3	A+B+C+D+E/(F+G/(H+K*L))
9	4	2	4	A(B+C(D+E(F+G(H+K/L))))

Запитання для самоперевірки

1. На які типи можна розділити багатопроцесорні обчислювальні системи?
2. Що являє собою багатопроцесорна ОС із роздільним керуванням? Навести особливості та недоліки системи типу М1.
3. Які є два принципових варіанти організації роботи системи з загальним керуванням? Навести особливості та недоліки систем типу М2.1. та М2.2.
4. У чому полягає принцип побудови систем матричного типу?
5. За якими критеріями можна визначити ефективність ОС?

Лабораторна робота 4

ЗНАХОДЖЕННЯ ОПТИМАЛЬНОЇ МОДИФІКАЦІЇ ОБЧИСЛЮВАЛЬНОЇ СИСТЕМИ ДЛЯ ОБЧИСЛЕННЯ АРИФМЕТИЧНИХ ВИРАЗІВ

Мета роботи: розробка ефективної структури ОС для обчислення арифметичних виразів.

Загальні теоретичні відомості

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

- коефіцієнт прискорення

$$K_n = T_0 / T_N, \qquad (8)$$

де T_0 – час розв'язання задачі в 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)

$$K_з = T_0 / (N * T_n), \qquad (9)$$

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

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

$$\alpha = t_m / t_c; \quad \beta = t_g / t_c, \qquad (10)$$

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. 11 зображено дерево арифметичного виразу:

$$a+bc+d/(e+f)+gh. \qquad (11)$$

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

$$T_0=4t_c+2t_m+t_g. \qquad (12)$$

Розглянемо можливість скорочення часу обчислення арифметичного виразу (11) за рахунок організації паралельного виконання операцій і використання обчислювальних систем різних структур.

На рис. 12 дерево арифметичного виразу (11) у вигляді, зручному для паралельної обробки. Обчислення виразу (11) 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 (11), який визначається кількістю виконуваних паралельно на кожному етапі операцій, змінюється від трьох на першому етапі до однієї на четвертому.

Розглянемо організацію обчислення арифметичного виразу (11) 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 (12) $T_0=11t_c$..

Для варіанта М2.1 є можливість організації паралельного виконання всіх трьох операцій на етапі 1 (рис. 4.2.).

На рис. 4.2. зображені часові діаграми завантаженості процесорів системи.

При такому розподілі операцій по процесорах час розв'язання задачі

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

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.

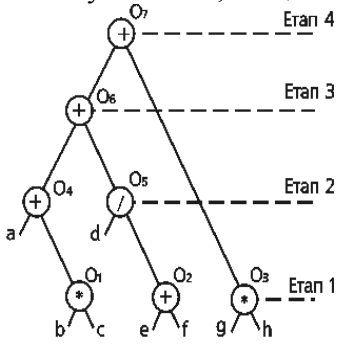


Рис.2.4

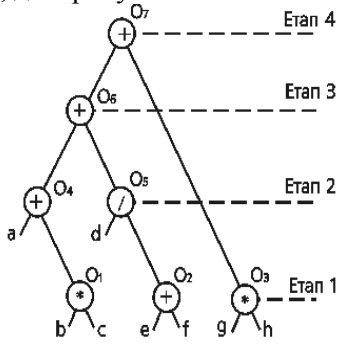


Рис.2.5

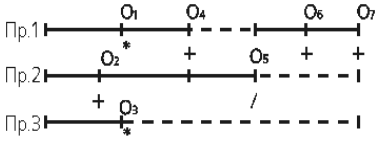


Рис.2.6

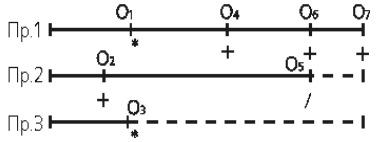


Рис.2.7

На рис. 12 показано змінене дерево арифметичного виразу (11).

Як видно із часових діаграм роботи процесорів (рис. 14), у цьому випадку можна скоротити час розв'язання задачі за рахунок ліквідації простою першого процесора:

$$T_{M2.1}=t_M+3t_c=5t_c.$$

Використовуючи співвідношення (11) і (12) для прийнятих N, α і β, визначимо коефіцієнти прискорення і завантаження:

$$K_П=T_0/T_{M2.1}=2{,}2,$$
$$K_З=T_0/(NT_{M2.1})=0{,}73.$$

Для другого варіанта організації роботи системи (М2.2.) 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 обчислювальних системах типу М2.2. арифметичного виразу, зображеного деревом на рис. 12 подані в табл. 6.

"Вектор-інструкції" для реалізації арифметичного виразу в ОС типу М2.2.

Таблиця 6

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

Часові діаграми завантаження процесорів системи показано на рис. 15.

$$T_{M2.2}=t_M+t_Д+2t_c=7t_c.$$

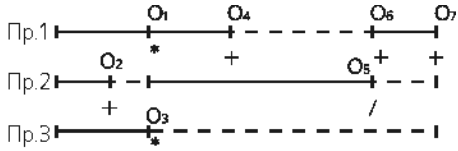


Рис.2.8

Значення коефіцієнтів прискорення і завантаження:

$K_П=T_0/T_{M2.2}=1,6;$
$K_З=T_0/(NT_{M2.2})=0,54.$

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. 16). Обчислення арифметичного виразу, який розглядається в обчислювальній системі типу М3, здійснюється в п'ять етапів.

При цьому

$$T_{M3}= t_m +t_g+3t_c=8t_c.$$

На рис. 17 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.

В табл. 7 наведено значення коефіцієнтів прискорення і завантаження під час розв'язання задачі обчислення арифметичного виразу (11) в різних обчислювальних системах.

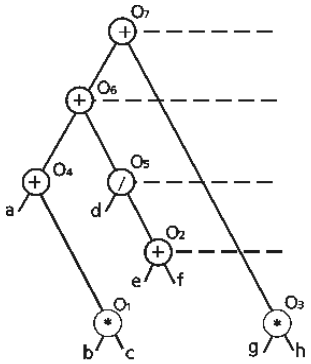


Рис.2.4

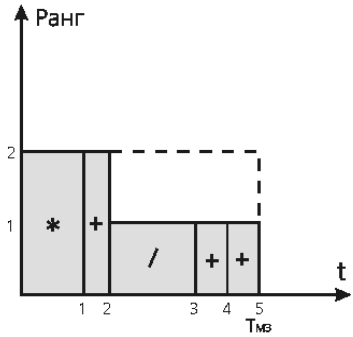


Рис.2.5

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

Таблиця 7

Тип ОС	$K_П$	$K_З$
М2.1	2,2	0,73
М2.2	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 М2.1. дозволяє розв'язати задачу за мінімальний час;
- за ступенем використання обладнання (завантаження процесорів) перевагу слід віддати системі типу М2.1.

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.

Вихідні дані:

а) тип структури (табл. 8);
б) арифметичний вираз (з лабораторної роботи 3);

в) кількість процесорів ОС (з лабораторної роботи 3);

г) значення коефіцієнтів α і β (з лабораторної роботи 3).

Номер варіанта визначається передостанньою цифрою номера залікової книжки. За нею обирається тип структури.

Тип структури ОС

Таблиця 8

Тип структури	Передостання цифра номера варіанта									
	0	1	2	3	4	5	6	7	8	9
М1	*	-	*	-	*	*	-	-	*	*
М2.1	-	*	-	*	-	*	-	*	-	*
М2.2	-	*	*	-	-	-	*	-	*	-
М3	*	-	-	*	*	-	*	*	-	-

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Накреслити дерево заданого арифметичного виразу.
5. Визначити час T_0 обчислення виразу в традиційній ЕОМ.
6. Для багатопроцесорної ОС конкретного типу обчислити значення T_N, K_y, $K_З$, та звести їх у таблицю.
7. На основі одержаних результатів побудувати структуру обчислювальної системи, яка дозволяє найбільш ефективно розв'язати задачу обчислення арифметичного виразу.
8. Зробити аналіз на основі значення рангу задачі, доцільність збільшення (зменшення) числа процесорів у системі. У випадку необхідності збільшення N визначити мінімальний час розв'язання задачі і структуру, яка дозволяє це зробити.
9. Скласти звіт по лабораторній роботі.
10. Зробити висновки по роботі.

Запитання для самоперевірки

1. За якими критеріями можна визначити ефективність ОС?
2. За якою формулою визначається коефіцієнт прискорення?
3. За якою формулою визначається коефіцієнт завантаження процесорів?

Приклади тестових завдань

1. Чому дорівнює час розв'язання задачі (A+B+C)*(D+G)/E+L*K/F, що розглядається в традиційній ЕОМ для випадку α=3; β=3:

а) $11t_c$;

б) $16t_c$;

в) $14t_c$.

2. Зробити аналіз ефективності різних структур, використовуючи таблицю, де наведено значення коефіцієнтів прискорення і завантаження під час розв'язання задачі обчислення арифметичного виразу в різних обчислювальних системах:

Тип ОС	$K_П$	$K_З$
М3	1,7	0,85
М1	2,1	0,7

а) - використання обчислювальної системи типу М3 дозволяє розв'язати задачу за мінімальний час.

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

б) - використання обчислювальної системи типу М3 дозволяє розв'язати задачу за мінімальний час;

- за ступенем використання обладнання (завантаження процесорів) перевагу слід віддати системі типу М1;

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

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

Лабораторна робота 5

АРХІТЕКТУРА КОНВЕЄРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Мета роботи: аналіз структур конвеєрних обчислювальних систем.

Загальні 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 $j=2,\ldots,P$).

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. Можливо, що в кожного ПК є свої індексні та базові регістри, регістри ключів захисту та інші індивідуальні ланцюги. Частина ланцюгів може бути загальної для різних ПК (наприклад, формувач адреси операндів, дешифратор коду операцій), які використовуються усіма ПК по черзі. Ці загальні ланцюги на рисунку включені до складу шарів процесора.

Пристрій керування приєднаний до шарів конвеєрного процесора через кільцевий комутатор. Комутатор улаштований 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 *(p+1)*-му такті 1-й пристрій знову підключається до 1-го шару процесора, де починається виконання 2-ї задачі (операції) і т.д.

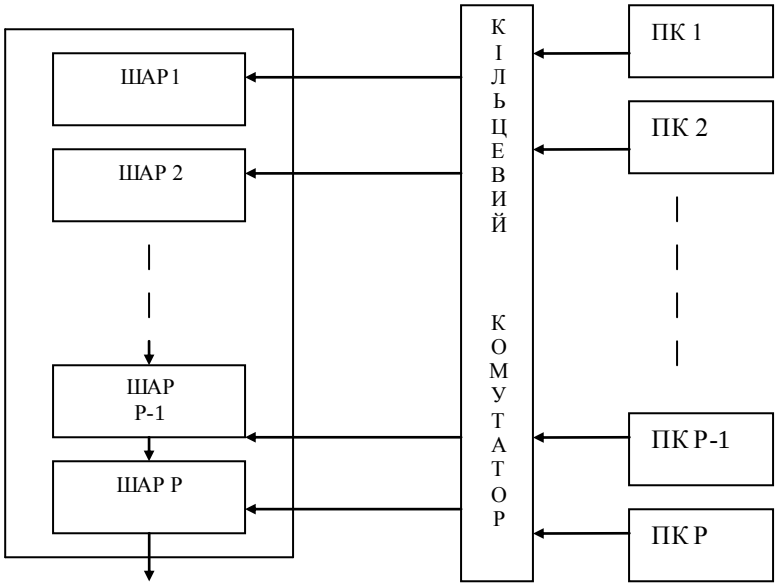


Рис. 5.1

Система, побудована зазначеним чином, аналогічна за своїми можливостями *P*-процесорній системі типу М1 (з роздільним керуванням) і орієнтована на використання паралелізму незалежних гілок.

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.

Конвеєрна система, побудована зазначеним чином, аналогічна багатопроцесорній системі М2 (із загальним керуванням). Будемо називати її ОС типу К2.

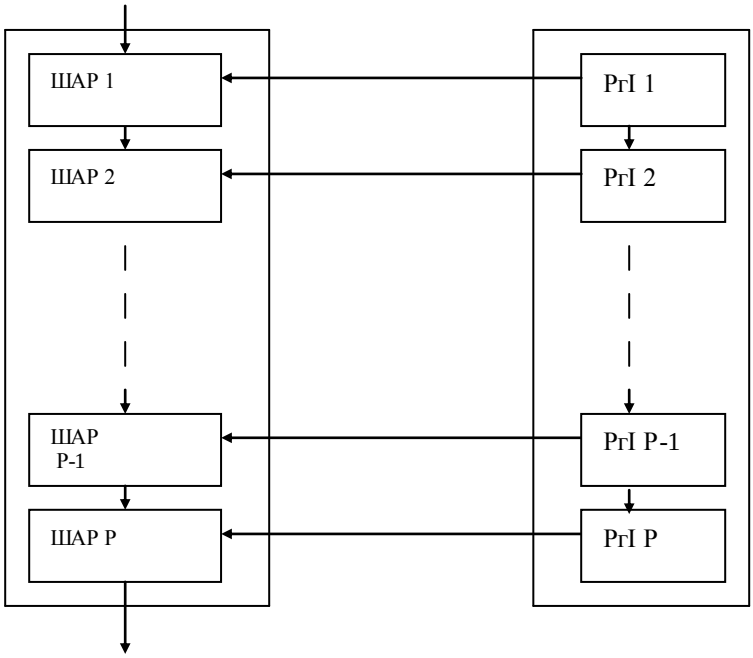


Рис. 5.2

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.

Вихідні дані:

- арифметичний вираз (табл. 5.1).

Номер варіанта визначається викладачем.

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Накреслити дерево арифметичного виразу.
5. Скласти звіт по лабораторній роботі.
6. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Що являє собою конвеєр?
2. Як за можливостями можна класифікувати конвеєр?
3. Що являють собою конвеєрні системи з роздільним керуванням? Навести особливості та недоліки системи К1.

4. На які типи поділяються багатофункціональні конвеєри, виходячи зі швидкості зміни функцій?

Таблиця 5.1

Арифметичний вираз, кількість шарів в конвеєрі *N*,
коефіцієнти τ_* та $\tau_/$

Номер варіанта	N	τ_*	$\tau_/$	Арифметичний 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))))

Лабораторна робота 6

АНАЛІЗ ФУНКЦІОНУВАННЯ КОНВЕЄРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Мета роботи: аналіз функціонування та ефективності конвеєрних обчислювальних систем.

Загальні теоретичні відомості

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 часів 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.

На рис. 6.1 зображено дерево арифметичного виразу (3):

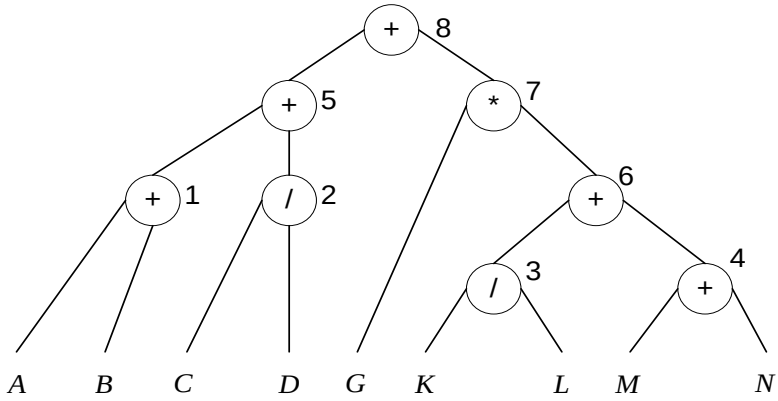


Рис. 6.1

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. 6.1, для випадку з N=4 (рис. 6.2).

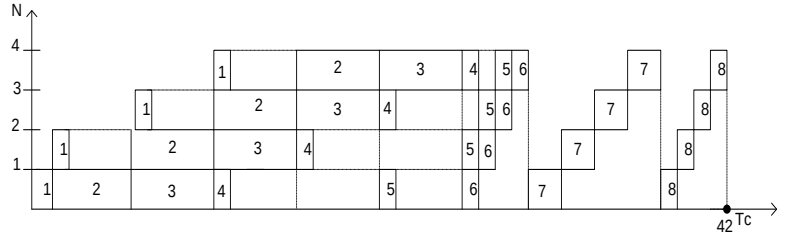


Рис. 6.2

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

$$K_n = T_0 / T_{дин} = \frac{68\tau_c}{42\tau_c} \approx 1{,}62;$$

$$K_З = T_0 / (N * T_{ДИН}) = \frac{68\tau_c}{4 * 42\tau_c} \approx 0{,}405.$$

2) Розглянемо діаграму роботи конвеєра зі статичною перебудовою (рис. 6.3).

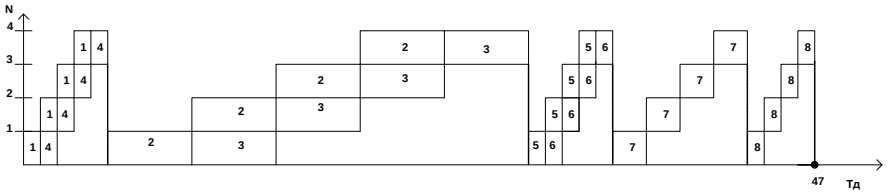


Рис. 6.3

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

$$K_n = T_0 / T_{ст} = \frac{68\tau_c}{47\tau_c} \approx 1{,}45;$$

$$K_З = T_0 / (N * T_{ст}) = \frac{68\tau_c}{4 * 47\tau_c} \approx 0{,}362.$$

3) Розглянемо діаграму роботи конвеєра з постійним тактом (рис. 6.4).

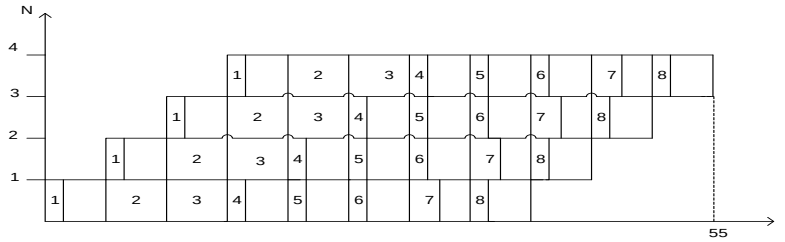


Рис. 6.4

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

$$K_n = T_0 / T_{пост} = \frac{68\tau_c}{55\tau_c} \approx 1{,}24;$$

$$K_З = T_0 / (N * T_{пост}) = \frac{68\tau_c}{4 * 55\tau_c} \approx 0{,}309.$$

В табл. 6.1 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	n	З
з динамічною перебудовою К2.1	,62	,405
зі статичною перебудовою К2.2	,45	,362
з постійним тактом К1	,24	,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 К2.1.

Вихідні дані:

- арифметичний вираз (табл. 5.1);
- кількість шарів конвеєру (табл. 5.1);
- значення коефіцієнтів τ_* та $\tau_/$ (табл. 5.1).

Номер варіанта визначається викладачем.

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Використовуючи дані з лабораторної роботи 1, визначити час обчислення виразу в традиційній ЕОМ.
5. Виконати задачу оптимальної завантаженості (задачу планування обчислень) для кожного типу конвеєрних ОС, що розглядаються.
6. Обчислити значення для типів конвеєру, що розглядаються, та звести їх у таблицю.
7. Зробити аналіз функціонування конвеєрів різних типів на підставі коефіцієнтів.
8. Скласти звіт по лабораторній роботі.
9. Відповісти на запитання для самоперевірки.

Примітка: задачу планування обчислень виконати у вигляді графічної залежності, де по осі абсцис відкладається час, а по осі ординат – кількість шарів в конвеєрі.

Запитання для самоперевірки

1. За якими критеріями можна визначити ефективність конвеєрної системи?
2. Навести особливості та недоліки систем К2.1 та К 2.2.
3. За якою формулою визначається коефіцієнт прискорення?
4. За якою формулою визначається коефіцієнт завантаження процесорів?

Приклади тестових завдань

Нехай задано арифметичний вираз вигляду: *P*O+A+B+C/D+G*(K+Z/M)*.

На рис. 6.5 зображено дерево виразу.

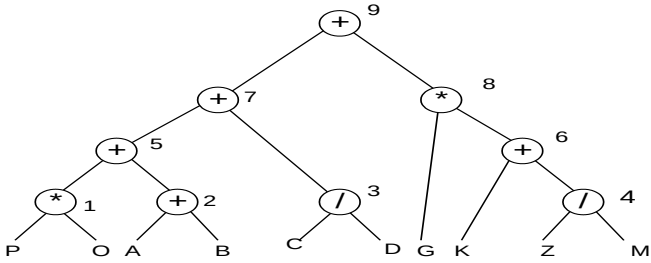


Рис. 6.5

Відомо, що N=3; $\tau_g = 2\tau_c$; $\tau_m = 4\tau_c$.

Завдання. Визначити, діаграми яких типів конвеєрів зображені на рис. 6.6, 6.7, 6.8.

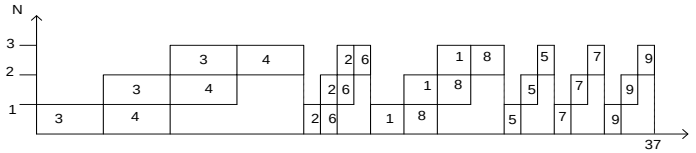


Рис. 6.6

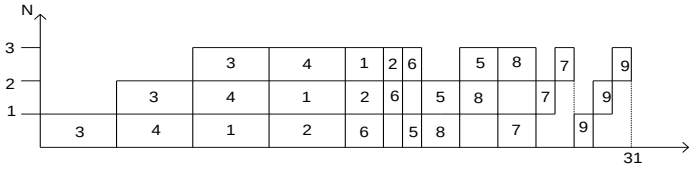


Рис. 6.7

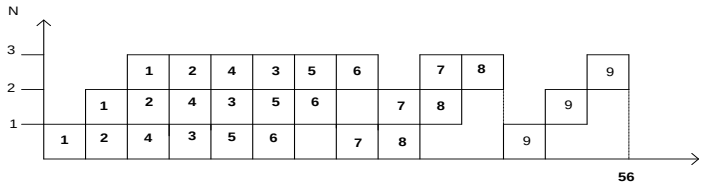


Рис. 6.8

Лабораторна робота 7

АРХІТЕКТУРА ПАРАЛЕЛЬНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Мета роботи: аналіз паралельних обчислювальних систем.

Загальні теоретичні відомості

В основі паралельних обчислювальних систем (ОС) полягає ідея, що вирішення задачі на паралельних процесорах забере в *p*-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 *p* 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. 7.1).

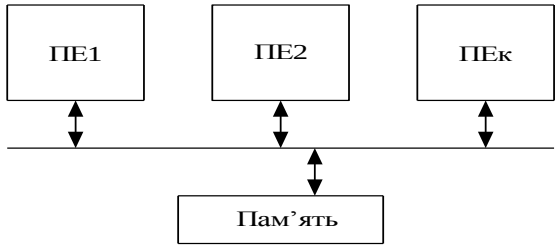


Рис. 7.1. Структура паралельної ОС із загальною пам'яттю

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. 7.2.).

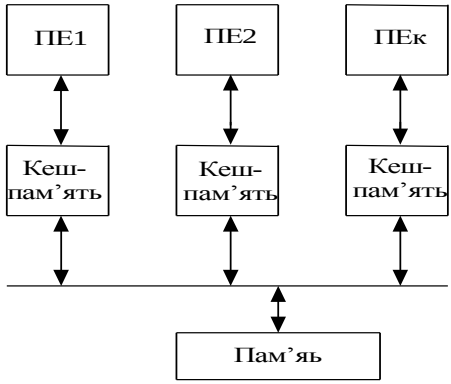


Рис. 7.2. Структура паралельної ОС із кеш-пам'яттю

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. 7.3).

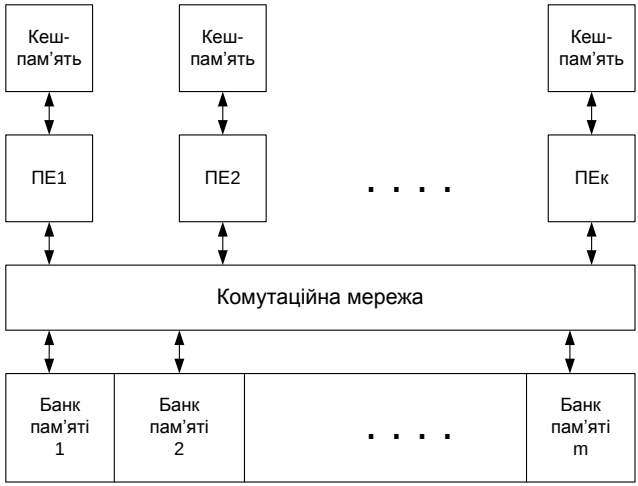


Рис. 7.3. Структура паралельної ОС із розподіленою пам'яттю банків з локально-адресованими просторами

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.

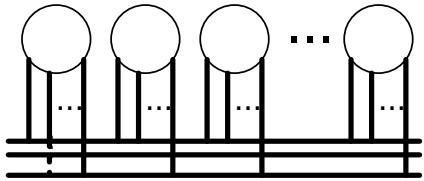


Рис. 7.4. Шинна топологія ОС

Багатошинна топологія передбачає наявність m незалежних шин для підключення до m банків пам'яті. (рис. 7.4). 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.

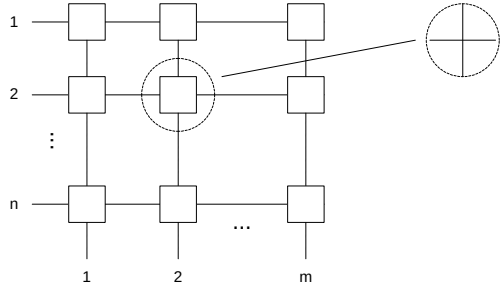


Рис. 7.5. Топологія перехресної комутації

Перехресний комутатор (рис. 7.5) 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.

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. 7.3 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.

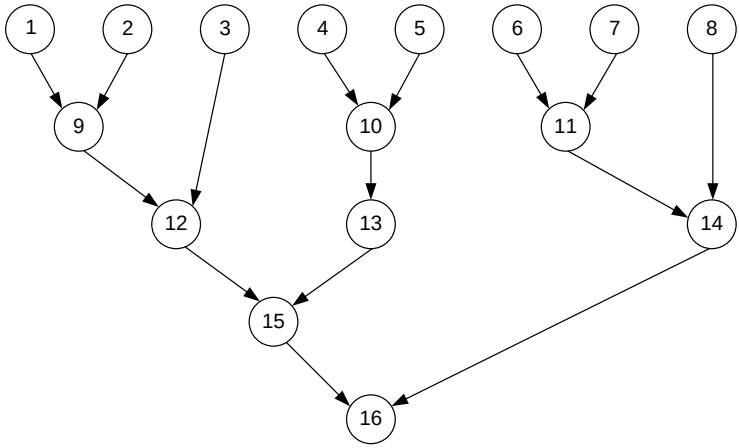
3.Паралельні ОС дозволяють нарощувати продуктивність в результаті модульної конструкції.

4.Паралельні ОС є відмовостійкими.

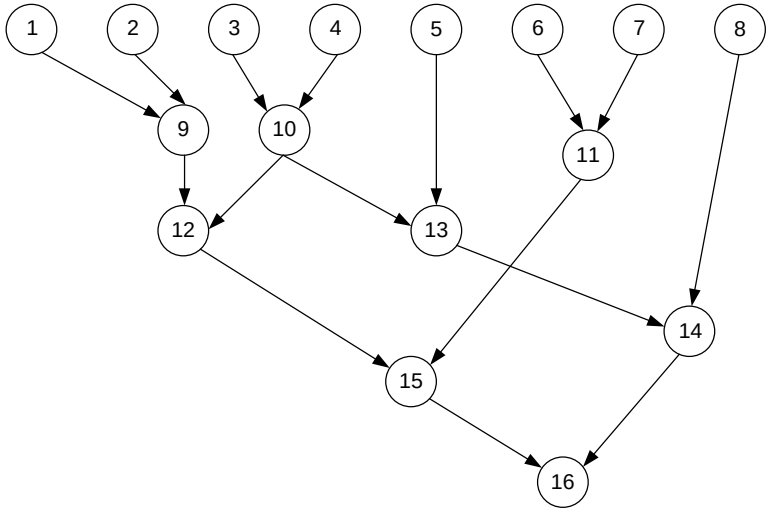
Вихідні дані:

а) граф обчислювальної задачі (рис. 7.6, 7.7).

Вихідні дані визначають за останньою цифрою залікової книжки. Остання цифра залікової книжки визначає граф обчислювальної задачі. Цифрам 0, 1 відповідає граф на рис. 7.6, цифрам 2, 3 відповідає граф на рис. 7.7, цифрам 4, 5 відповідає граф на рис. 7.8, цифрам 6, 7 відповідає граф на рис. 7.9, цифрам 8, 9 відповідає граф на рис. 7.10.

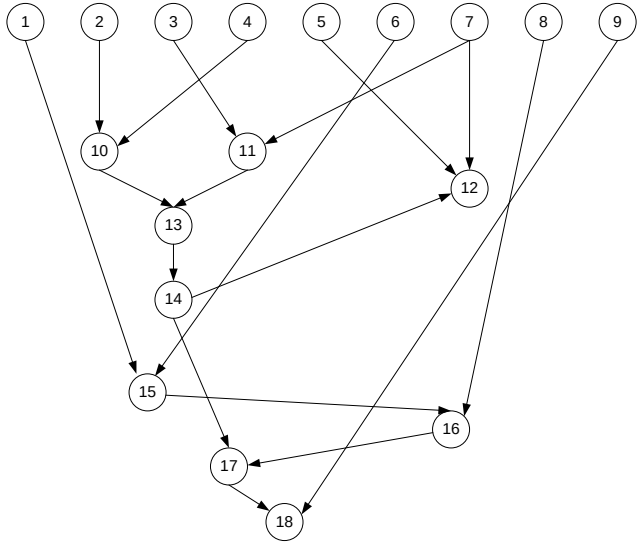


а

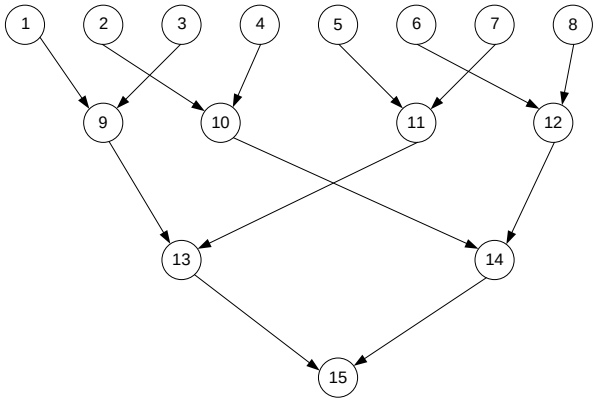


б

Рис. 7.6. Граф обчислювальної задачі: *а* – остання цифра залікової книжки 0; *б* – остання цифра залікової книжки 1

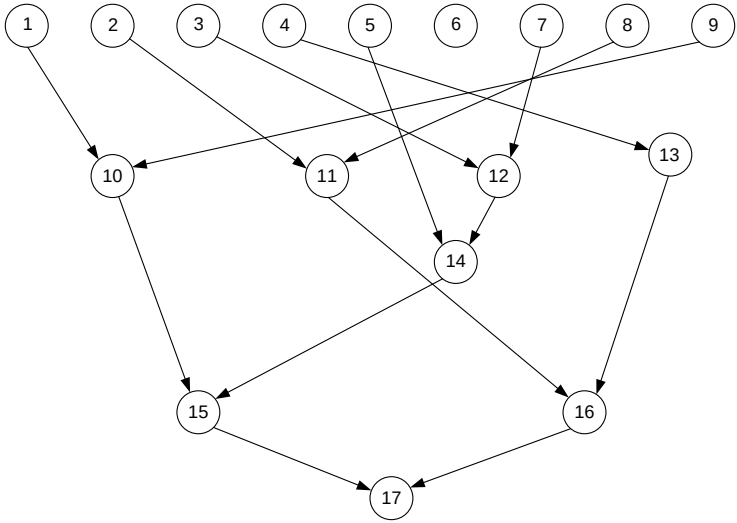


а

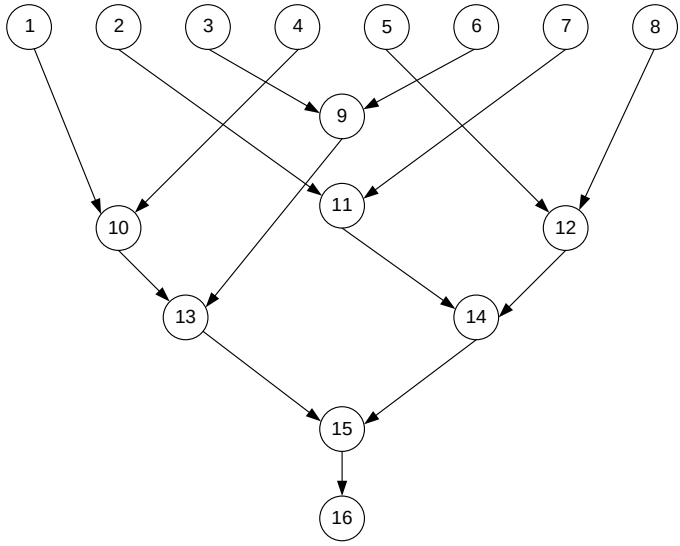


б

Рис. 7.7. Граф обчислювальної задачі: *а* – остання цифра залікової книжки 2; *б* – остання цифра залікової книжки 3

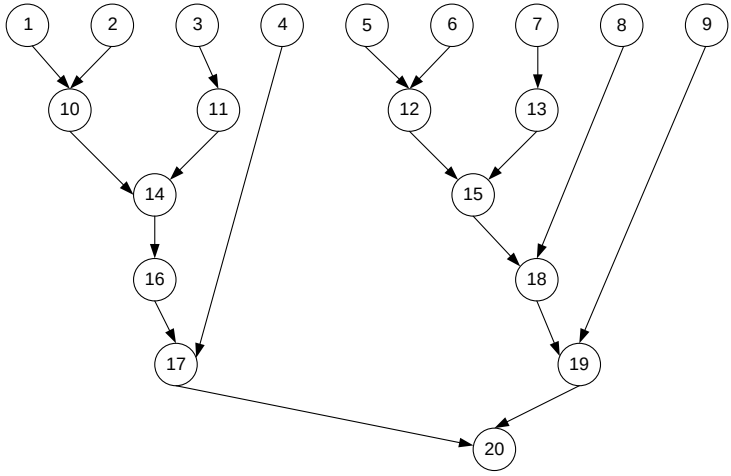


а

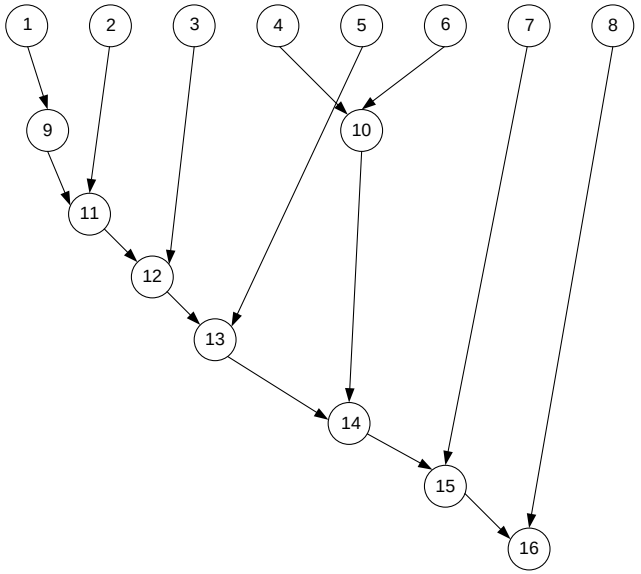


б

Рис. 7.8. Граф обчислювальної задачі: *а* – остання цифра залікової книжки 4; *б* – остання цифра залікової книжки 5

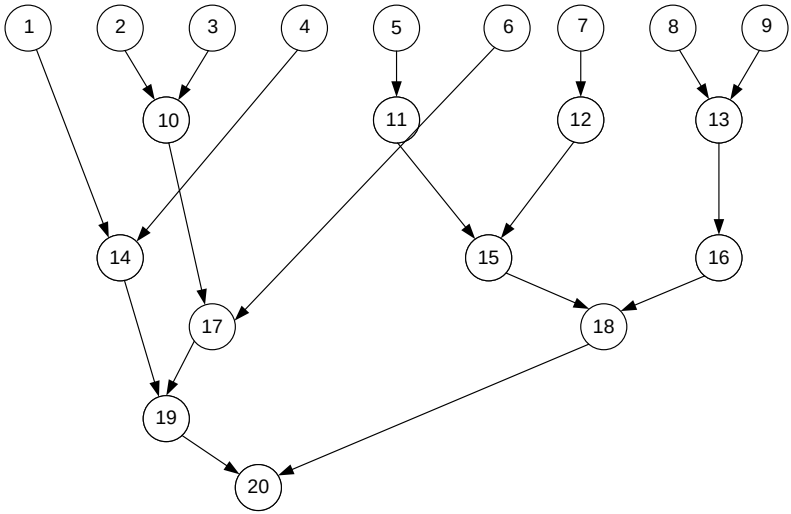


а

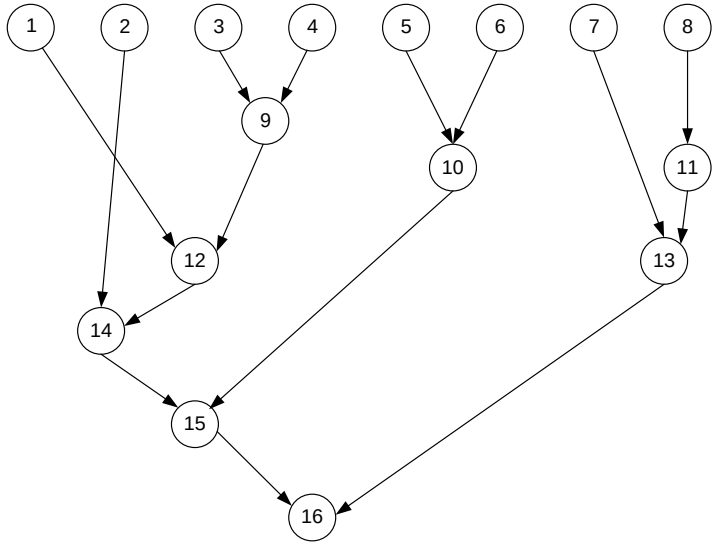


б

Рис. 7.9. Граф обчислювальної задачі: *а* – остання цифра залікової книжки 6; *б* – остання цифра залікової книжки 7



а



б

Рис. 7.10. Граф обчислювальної задачі: *а* – остання цифра залікової книжки 8; *б* – остання цифра залікової книжки 9

Хід виконання 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.
5. Скласти звіт по лабораторній роботі.
6. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Яка ідея полягає в основі паралельних ОС?
2. Назвіть важливі класифікаційні ознаки паралельних ОС.
3. Які існують способи керування сукупністю процесорів в паралельних ОС?

4. Проілюструйте доступ до пам'яті в паралельних ОС із розподіленою пам'яттю.

5. Що являють собою архітектури із локальною пам'яттю?

6.Назвіть переваги паралельних ОС у порівнянні із послідовними.

Лабораторна робота 8

АНАЛІЗ ФУНКЦІОНУВАННЯ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ З ЛОКАЛЬНОЮ ПАМ'ЯТТЮ РІЗНИХ ТОПОЛОГІЙ

Мета роботи: аналіз функціонування та ефективності обчислювальних систем різних топологій.

Загальні 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. 8.1).

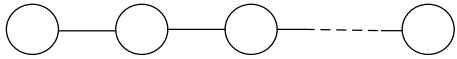


Рис. 8.1. Лінійна топологія ОС

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.

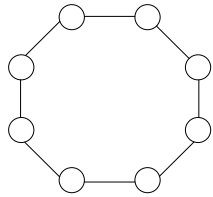


Рис. 8.2. Кільцева топологія ОС

Стандартна кільцева топологія (рис. 8.2) являє собою лінійний ланцюг, кінці якого з'єднані між собою. Недолік: додавання чи вилучення вузла потребує демонтажу мережі.

Зіркоподібна організація вузлів (рис. 8.3) 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.

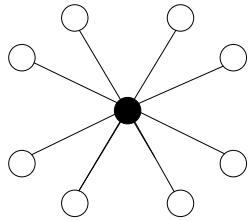


Рис. 8.3. Зіркоподібна топологія ОС

У деревоподібній топології (рис. 8.4) 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.

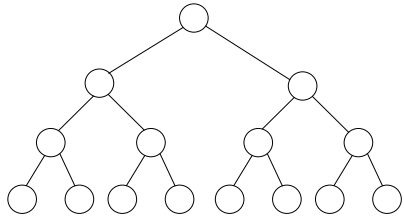


Рис. 8.4. Деревоподібна топологія ОС

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. 8.5) орієнтовані на задачі, пов'язані з обробкою масивів. Їх конфігурація визначається видом та розмірністю масиву.

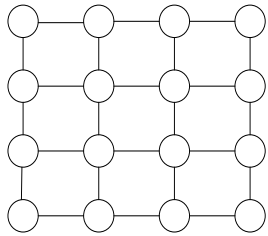


Рис. 8.5. Решітчаста топологія ОС

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. 8.6, *а*), визначає одновимірний гіперкуб. Квадрат, що утворений чотирма вузлами (рис. 8.6, *б*) – двовимірний гіперкуб, а куб з 8 вузлів (рис. 8.6, *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.

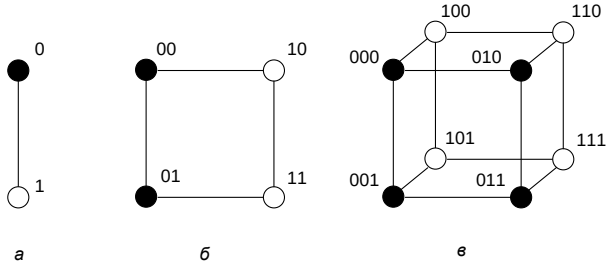


Рис. 8.6. Гіперкубова топологія ОС

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. 8.7), 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.

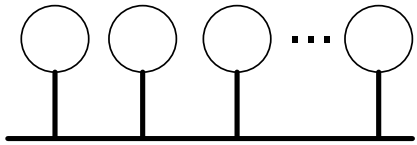


Рис. 8.7. Одношинна топологія ОС

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. 8.8. дерева обчислювальної задачі:

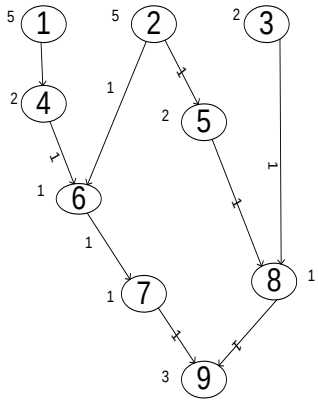


Рис. 8.8. Дерево обчислювальної задачі

ОС для трьох процесорів топології "Зірка" подана на рис. 8.9.

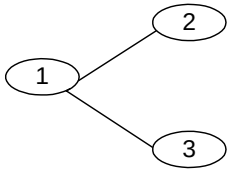


Рис. 8.9. Топологія "Зірка"

Часову діаграму роботи ОС для трипроцесорної ОС топології "Зірка" наведено на рис. 8.10.

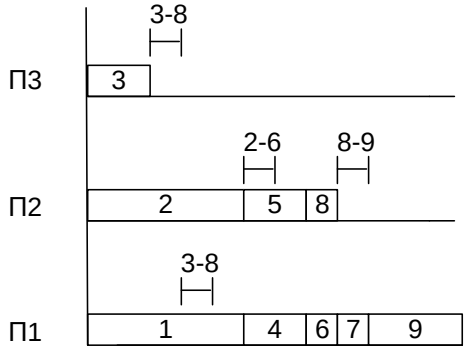


Рис. 8.10. Часова діаграма роботи ОС для трьохпроцесорної ОС топології "Зірка"

Таблиця 8.1

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	20
0	4	2	6	1	3	2	7	1	1	3	6	3	2	3	6	5	1	1	4	5
1	1	3	3	5	2	1	3	4	3	2	2	2	8	2	1	3	2	5	4	3
2	1	4	6	5	5	3	2	2	3	3	1	7	5	1	2	6	1	2	3	4
3	1	4	4	3	2	5	6	2	1	1	3	4	1	4	2	5	7	3	2	1
4	2	2	4	2	5	3	2	1	5	3	2	7	3	6	2	7	3	1	1	4
5	3	4	3	8	4	2	6	4	5	3	3	7	4	5	2	4	2	3	1	3
6	4	6	1	4	3	5	4	3	6	3	2	4	3	4	6	2	5	7	8	2
7	1	5	2	3	2	7	2	1	6	3	4	5	6	1	2	6	4	4	2	3
8	7	2	5	2	1	3	2	2	5	3	4	3	4	2	3	7	2	5	2	3
9	5	2	6	1	5	3	4	4	3	2	1	2	8	7	5	2	1	3	3	1

Вихідні дані:

а) час виконання операцій (табл. 8.1.);

б) топологія ОС та кількість процесорів (табл. 8.2).

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

Хід виконання роботи

10. Вивчити теоретичні відомості.
11. Отримати допуск до роботи.
3. Вибрати за методичними вказівками варіант завдання.
4. Використовуючи дані з лабораторної роботи 7 (рис. 7.6. – 7.10), побудувати часові діаграми роботи ОС заданих топологій та кількості процесорів (табл. 8.2) з заданим часом виконання операцій (табл. 8.1).
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.
8. Відповісти на запитання для самоперевірки.

Примітка: Кодова відстань між операціями постійна і дорівнює 1.

Таблиця 8.2

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				*	*	*	*	*				*	*		

Запитання для самоперевірки

5. Що називається топологією?
6. Які розрізняють ОС залежно від того, чи залишається конфігурація взаємозв'язків незмінною?
7. Які розрізняють типи статичних топологій?
8. Які розрізняють типи динамічних топологій?
9. Проаналізувати основні переваги та недоліки різних топологій мережі зв'язку процесорів.
10. Як мінімізуються недоліки окремих типів систем?

Приклади тестових завдань

Для заданого графа обчислювальної задачі (рис. 8.11) побудувати часові діаграми роботи ОС для топологій "лінійка" та "дерево" з різною кількістю процесорів.

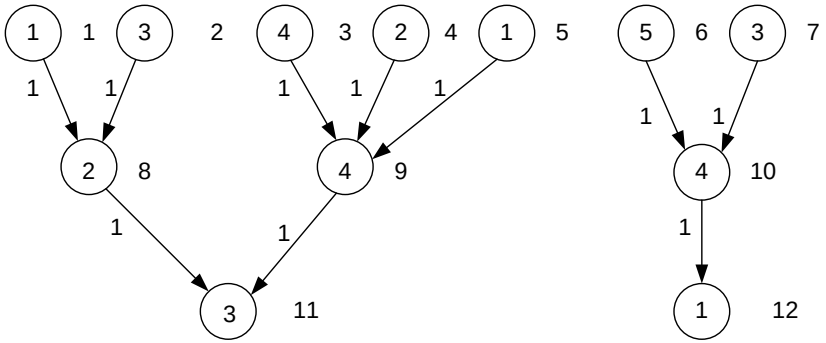


Рис. 8.11. Граф обчислювальної задачі

Топологія “лінійка” з двома процесорами подана на рис.8.12.

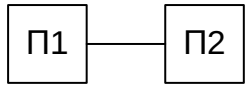


Рис. 8.12. Топологія “лінійка” з двома процесорами

Побудуємо часову діаграму роботи ОС для топології “лінійка” з двома процесорами та визначимо час обчислення заданої задачі (рис. 8.13).

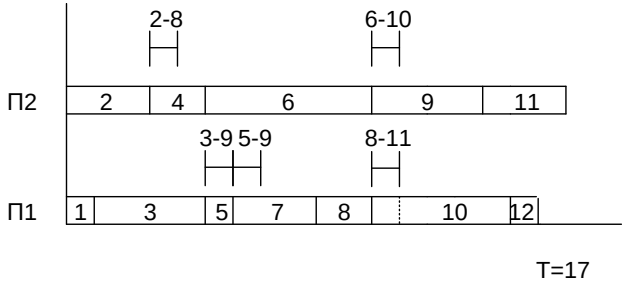


Рис. 8.13. Часова діаграма роботи ОС топології “лінійка” з двома процесорами

Топологія “лінійка” з чотирма процесорами подана на рис.8.14.

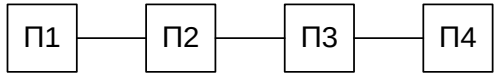


Рис. 8.14. Топологія “лінійка” з чотирма ПЕ

Побудуємо часову діаграму роботи ОС для топології “лінійка” з чотирма процесорами та визначимо час обчислення заданої задачі (рис. 8.15).

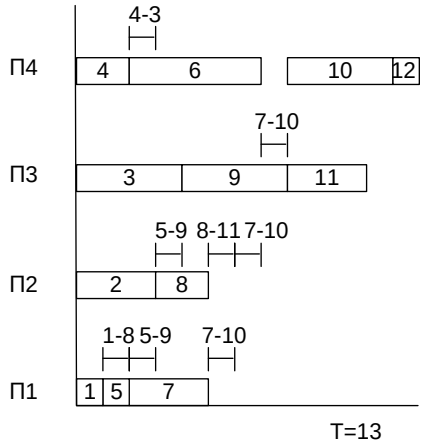


Рис. 8.15. Часова діаграма роботи ОС топології “лінійка” з чотирма процесорами

Топологія “дерево” з трьома процесорами подана на рис.8.16.

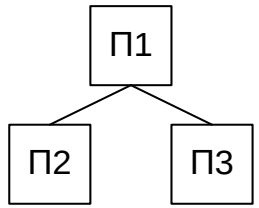


Рис. 8.16. Топологія “дерево” з трьома ПЕ

Побудуємо часову діаграму роботи ОС для топології “дерево” з трьома процесорами та визначимо час обчислення заданої задачі (рис. 8.17).

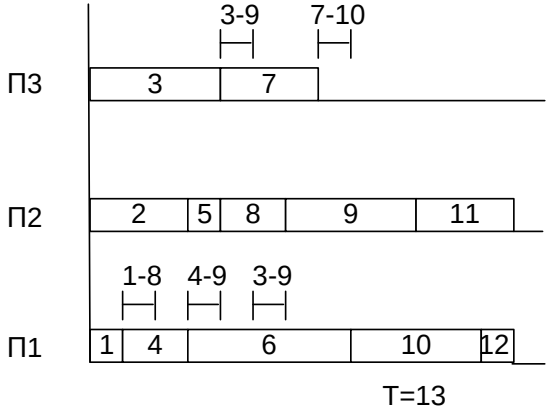


Рис. 8.17. Часова діаграма роботи ОС топології “дерево” з трьома процесорами

Топологія “дерево” з сьома процесорами подана на рис. 8.18.

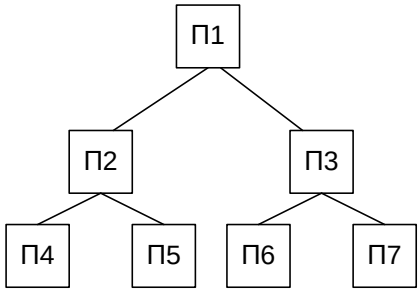


Рис. 8.18. Топологія “дерево” з сьома процесорами

Побудуємо часову діаграму роботи ОС для топології “дерево” з сьома процесорами та визначимо час обчислення заданої задачі (рис. 8.19).

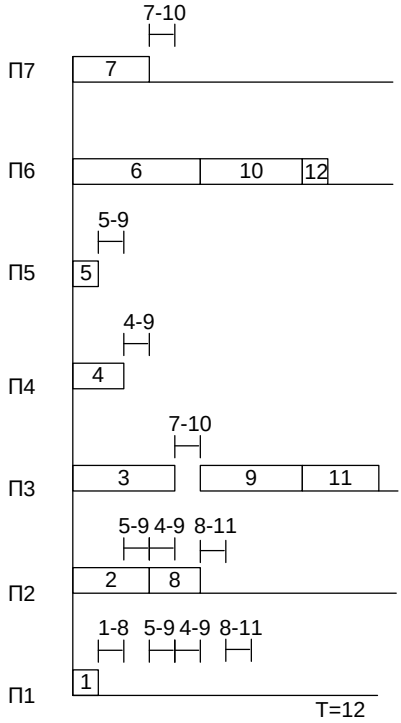


Рис. 8.19. Часова діаграма роботи ОС топології “дерево” з сьома процесорами

СПИСОК ЛІТЕРАТУРИ

1.БОГДАНОВ А.В., КОРХОВ В.В., МАРЕЕВ В.В., СТАНКОВА Е.Н. Архитектуры и топологии многопроцессорных вычислительных систем. Интернет-университет информациогнных технологий – ИНТУИТ.ру, 2004.

2.БРОЙДО В. Вычислительные системы, сети и телекоммуникации. – СПб.: Питер, 2002. – 688 с.

3.ГОЛОВКИН Б.А. Вычислительные системы с большим числом процессоров.-М.: Радио и связь, 1995. – 320 с.

4.КОУГИ П.М. Архитектура конвейерных ЭОМ: Пер. с англ. – М.: Радио и связь, 1985.

5.ОБЧИСЛЮВАЛЬНІ системи: Лабораторні роботи 1-4/ Уклад.: І.А. Жуков, Є.В. Красовська. – К .: НАУ, 2003.–34 с.

6.ОБЧИСЛЮВАЛЬНІ системи: Лабораторні роботи 5,6/ Уклад.: І.А. Жуков, Є.В. Красовська, О.В. Русанова. – К .: НАУ, 2005. – 16 с.

7.ОСНОВЫ теории вычислительных систем: Учебное пособие/ Под ред. С.А.Майорова. – М.: Высш.шк., 1978. – 260 с.

8.САМОФАЛОВ К.Г., ЛУЦКИЙ Г.М. Основы теории многоуровневых конвейерных вычислительных систем. – М.: Радио и связь, 1989. – 272 с.

9. ЦИЛЬКЕР Б. Я., ОРЛОВ С. А. Организация ЭВМ и систем: Учебник для вузов. – СПб.: Питер, 2006. – 668 с.