

## Розділ 6

### *Алгоритми виконання операцій обробки даних*

Операції обробки даних ініціюються відповідними командами обробки даних. До числа цих операцій входять:

- логічні операції (логічне множення, логічне додавання, інверсія і т. д.) над розрядами слів, скалярами та векторами;
- операції зсуву (праворуч, ліворуч) над скалярами та векторами;
- операції відношення: менше, більше, рівне, менше-рівне, більше-рівне;
- арифметичні операції (додавання, віднімання, множення та ділення) над одиночними даними та векторами даних;
- операції обчислення елементарних функцій типу  $\exp X$ ,  $\ln X$ ,  $\sin X$ ,  $\cos X$ ,  $\arctg y/x$ ,  $\text{Sh } X$ ,  $\text{Ch } X$ , піднесення до степеня  $A^m$ ;
- операції перетворення даних (перетворення із формату з фіксованою в формат з рухомою комою і навпаки, перетворення з двійково-десяtkового коду в двійковий і на-впаки і т. д.);
- операції реорганізації масивів і визначення їх параметрів: сортування, пошук максимуму або мінімуму, вибір заданого масиву, зсув елементів масиву, стиск масиву;
- операції обробки символів та стрічок символів: пошук символу, зсув, заміна символов у стрічці, пакування стрічок символів, порівняння стрічок символів.

В останніх комп'ютерах у зв'язку з широким використанням засобів телекомунікацій та мультимедіа до складу основних операцій добавилися складні операції типу кодування, компресії, шифрування тощо.

В даному розділі розглянемо основні алгоритми виконання вищезазначених операцій, не вникаючи в питання їх реалізації в комп'ютері.

#### **6.1. Логічні операції**

До складу команд обробки даних входить велика кількість команд, які ініціюють логічні операції. До їх числа входять операції булевої алгебри: логічне множення, додавання, додавання по модулю два, інверсія і т. д. При цьому логічні операції можуть виконуватись над окремими розрядами слова, над одиночними даними, а також над векторами даних.

Логічні операції передбачають побітову обробку даних. Коли говорять про логічну операцію над парою слів, то мається на увазі, що проводяться окремі операції над кожною парою біт, які входять до цих слів.

### 6.1.1. Операція заперечення

Операція заперечення (інверсія, НЕ, NOT) є операцією над одним операндом і означає, що біти із значенням “0” набувають значення “1” і навпаки. Для відображення дії логічної операції часто використовують так звані таблиці істинності. Табл. 6.1 є таблицею істинності для операції заперечення.

Таблиця 6.1

біт операнда	біт результату
0	1
1	0

Приклади:

$$\text{NOT} (1000\ 1010\ 0010\ 1100) = 0111\ 0101\ 1101\ 0011.$$

$$\text{NOT} (1110\ 1011\ 1010\ 0111) = 0001\ 0100\ 0101\ 1000.$$

### 6.1.2. Логічне I

Ця операція (загальноприйняте позначення I, AND) передбачає наявність як мінімум двох операндів, назовемо їх X та Y. Вона виконує порозрядну кон'юнкцію змінних, тобто отримання одиниці лише тоді, коли всі вхідні змінні рівні одиниці. Відобразимо значення функції наступною таблицею істинності (табл. 6.2.)

Таблиця 6.2

біт X	біт Y	біт результату
0	0	0
0	1	0
1	0	0
1	1	1

Приклади виконання операції логічного множення приведено на рис. 6.1.

$$\begin{array}{ll} \text{AND} & 1101\ 0011\ 0010\ 1110 \\ & 0011\ 0010\ 1111\ 0000 \\ = & 0001\ 0010\ 0010\ 0000 \end{array} \quad \begin{array}{ll} \text{AND} & 1100\ 0101\ 0010\ 1100 \\ & 1100\ 1101\ 1111\ 0010 \\ = & 1100\ 0101\ 0010\ 0000 \end{array}$$

Рис. 6.1. Приклади виконання операції логічного множення

### 6.1.3. Логічне АБО

Ця операція (загальноприйняте позначення АБО, OR) також передбачає наявність як мінімум двох операндів X та Y. Вона виконує порозрядну диз'юнкцію змінних, тобто отримання одиниці тоді, коли хоча б одна вхідна змінна рівна одиниці. Відобразимо значення функції наступною таблицею істинності (табл. 6.3).

Таблиця 6.3

біт X	біт Y	біт результату
0	0	0
0	1	1

1	0	1
1	1	1

Приклади виконання операції логічного додавання приведено на рис. 6.2.

$$\begin{array}{l} \text{OR} \\ \quad \begin{array}{l} 1101\ 0011\ 0010\ 1110 \\ 0011\ 0010\ 1111\ 0000 \\ =\ 1111\ 0011\ 1111\ 1110 \end{array} \end{array}$$

$$\begin{array}{l} \text{OR} \\ \quad \begin{array}{l} 1100\ 0101\ 0010\ 1100 \\ 1100\ 1101\ 1111\ 0010 \\ =\ 1100\ 1101\ 1111\ 1110 \end{array} \end{array}$$

Рис. 6.2. Приклади виконання операції логічного додавання

#### 6.1.4. Виключне АБО

Також цю операцію ще називають додавання за модулем два (XOR), оскільки вона виконує порозрядне додавання вхідних змінних за модулем два. Ця операція виконується як мінімум над двома операндами X та Y. Відобразимо значення функції наступною таблицею істинності (табл. 6.4).

Таблиця 6.4

біт X	біт Y	біт результату
0	0	0
0	1	1
1	0	1
1	1	0

Приклади виконання операції логічного додавання за модулем два наведено на рис. 6.3.

$$\begin{array}{l} \text{AND} \\ \quad \begin{array}{l} 1101\ 0011\ 0010\ 1110 \\ 0011\ 0010\ 1111\ 0000 \\ =\ 0001\ 0010\ 0010\ 0000 \end{array} \end{array}$$

$$\begin{array}{l} \text{AND} \\ \quad \begin{array}{l} 1100\ 0101\ 0010\ 1100 \\ 1100\ 1101\ 1111\ 0010 \\ =\ 1100\ 0101\ 0010\ 0000 \end{array} \end{array}$$

Рис. 6.3. Приклади виконання операції логічного додавання за модулем два

### 6.2. Операції зсуву

#### 6.2.1. Логічні зсуви

При виконанні логічного зсуву праворуч або ліворуч всі розряди слова зсуванняться на один розряд у відповідну сторону, в перший розряд записується нуль, а останній розряд випадає (рис. 6.4). Зсуви досить часто використовуються як складові операції при виконанні багатьох алгоритмів обробки даних. Для формату представлення даних без знаків зсув на один розряд ліворуч еквівалентний множенню на два, а на один розряд праворуч – відповідно ціличисельному діленню на два.

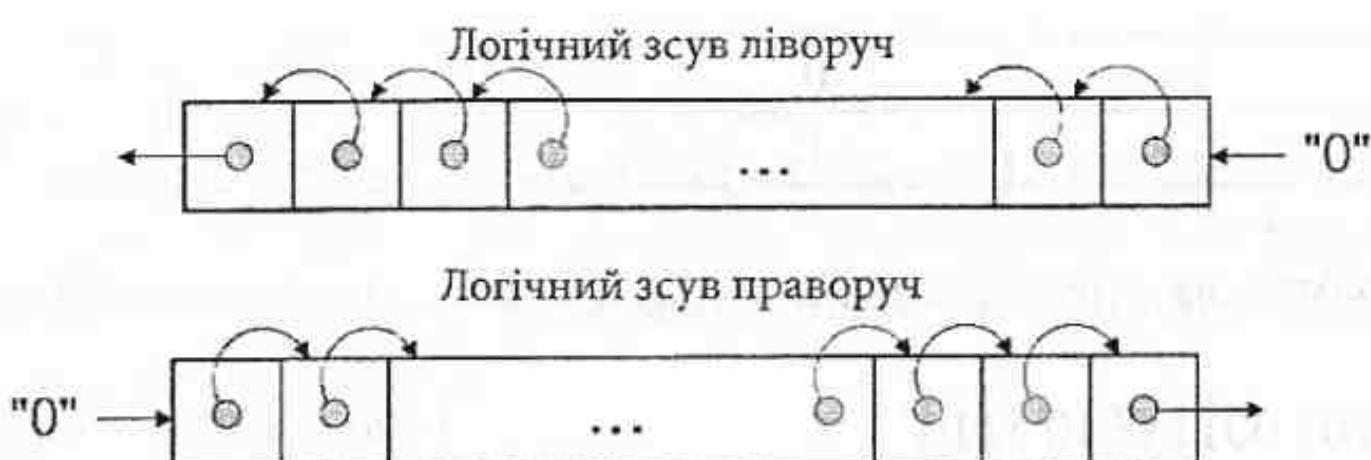


Рис. 6.4. Логічний зсув ліворуч та праворуч

Команда логічного зсуву має наступні поля: код логічної операції зсуву праворуч або ліворуч, адреса операнда та код зсуву, який вказує величину зсуву, тобто кількість розрядів, на які має бути проведений зсув. Якщо операнд позначити через  $X$ , а код зсуву через  $Y$ , то результат  $Z$  виконання операції буде рівним  $Z = X 2^{+/-Y}$ , де знак “+” відповідає зсуву ліворуч, а знак “-” – праворуч.

### 6.2.2. Арифметичні зсуви

Арифметичні зсуви дуже подібні до попередніх (логічних), але вони мають таку особливість, як розмноження знака при зсуві праворуч та збереження знака при зсуві ліворуч (рис. 6.5).



Рис. 6.5. Арифметичний зсув ліворуч та праворуч

Такі зсуви також мають зміст множення і ділення на два для формату представлення даних із знаками в оберненому та доповнільному кодах. Вони дозволяють після операції зсуву ліворуч зберегти знак представленого числа та після операції зсуву праворуч зберегти коректний результат ділення на два.

Команда арифметичного зсуву має наступні поля: код операції арифметичного зсуву праворуч або ліворуч, адреса операнда та код зсуву, який вказує величину зсуву, тобто кількість розрядів, на які має бути проведений зсув.

### 6.2.3. Циклічні зсуви

Циклічний зсув передбачає, що розряди, які витісняються з одного боку операнда, дописуються з іншого його боку (рис. 6.6).



Рис. 6.6. Циклічний зсув ліворуч та праворуч

Команда циклічного зсуву має наступні поля: код операції циклічного зсуву право- руч або ліворуч, адреса операнда та код зсуву, який вказує величину зсуву, тобто кіль- кість розрядів, на які має бути проведений зсув.

## 6.3. Операції відношення

### 6.3.1. Порівняння двійкових кодів на збіжність

Операція порівняння двійкових кодів на збіжність дає логічний результат, рівний одиниці, якщо коди збігаються. В інших випадках значення результата рівне нулю. Виконувана функція визначається наступним булевим рівнянням:

$$Z = \text{AND} \left( \left( \underset{i=0}{\overset{n}{\text{AND}}} X_i \text{ AND } Y_i \right) \text{ OR } \left( \text{NOT}(X_i) \text{ AND } \text{NOT}(Y_i) \right) \right),$$

де  $i = 0, 1 \dots n$  – номери розрядів чисел  $X$  та  $Y$ , які порівнюються,  $Z$  – розряд ре- зультату.

### 6.3.2. Визначення старшинства двійкових кодів

Операції визначення старшинства двійкових кодів, тобто визначення, яке з двох чисел є меншим, більшим, меншим-рівним, більшим-рівним, виконуються з використан- ням двійкового віднімання.

Якщо це числа без знаків, то при відніманні першого числа від другого отриманий результат може бути більшим нуля, рівний нулю, або меншим нуля. Тоді, якщо отриманий результат є більшим нуля, то перше число є більшим від другого, якщо отриманий результат рівний нулю, то числа рівні, а якщо отриманий результат є меншим нуля, то друге число є більшим від першого.

Якщо це числа із однаковими знаками, то при відніманні першого числа від другого отриманий результат може бути додатнім, рівним нулю, або від'ємним. Тоді якщо отриманий результат рівний нулю, то числа рівні, при від'ємному результаті перше число є меншим другого, а друге відповідно більшим, а при додатному результаті перше число є більшим, а друге відповідно меншим від першого. Якщо ж це числа із різними знаками, то більшим є додатне число.

## 6.4. Арифметичні операції

Команди обробки даних також ініціюють виконання арифметичних операцій. Це операції додавання, віднімання, множення та ділення над числами, представленими в форматах з фіксованою та рухомою комою. При цьому арифметичні операції також можуть виконуватись як над скалярними даними, так і над векторами.

До арифметичних належать також наступні операції над одиночними операндами:

- взяття абсолютної величини від операнда;
- інверсія знака операнда;
- прирощення операнда на одиницю;
- зменшення операнда на одиницю.

Методи реалізації арифметичних операцій в комп'ютерах вибирають з врахуванням потрібної швидкодії, формату представлення чисел, системи числення та інших техніко-економічних і системних факторів. Критерієм вибору методу зазвичай є досягнення мінімальних затрат обладнання, максимальної швидкодії або оптимального співвідношення затрат обладнання і швидкодії. Існує велика кількість алгоритмів виконання арифметичних операцій в комп'ютері. В даному розділі розглянемо лише основні алгоритми, а питання їх модифікації та реалізації будуть розглянуті в наступних розділах.

### 6.4.1. Додавання двійкових чисел без знаків

Додавання в двійковій системі числення відбувається подібно до звичайного додавання в десятковій системі. Додаються два розряди, які розміщені у числі на одній позиції. При виникненні переносу він передається в старший розряд і там додається. В загальному додавання можна описати наступними формулами:

$$\begin{aligned} S_i &= x_i \text{ XOR } y_i \text{ XOR } c_i \\ c_{i+1} &= (x_i \text{ AND } y_i) \text{ OR } (x_i \text{ AND } c_i) \text{ OR } (y_i \text{ AND } c_i) \\ c_0 &= 0 \end{aligned}$$

де:  $S_i$  – i-тий розряд суми,  $x_i$ ,  $y_i$  – i-ті розряди першого та другого доданків відповідно,  $c_i$  – розряд переносу.

Ці формули отримані з наступної таблиці істинності (табл. 6.5).

Таблиця 6.5

$c_i$	$x_i$	$y_i$	$S_i$	$c_{i+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Якщо результат, який мав би бути отриманий після додавання, є більшим, ніж може вміститися в даній розрядній сітці, то старший розряд втрачається і така ситуація називається втратою результату через переповнення.

Приклади виконання операції додавання двійкових чисел без знаків приведено на рис. 6.7.

$$\begin{array}{r}
 101101011 \\
 + 01000001 \\
 \hline
 10101100
 \end{array}$$
  

$$\begin{array}{r}
 11011001 \\
 + 11010110 \\
 \hline
 110001001
 \end{array}$$

Переповнення

Рис. 6.7. Приклади виконання операції додавання двійкових чисел без знаків

Позначимо оператор виконання операції однорозрядного двійкового додавання відповідно до табл. 6.5 знаком суми, як це показано на рис. 6.8.

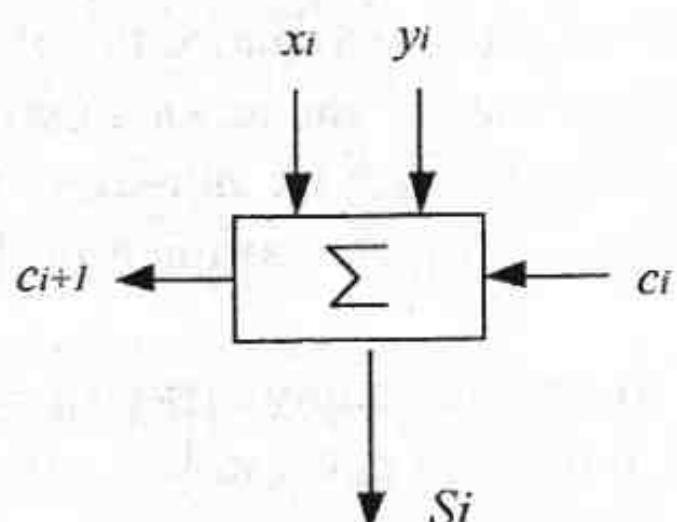


Рис. 6.8. Оператор однорозрядного двійкового додавання

Тут  $s_i$  –  $i$ -й розряд суми,  $x_i$ ,  $y_i$  –  $i$ -ті розряди першого та другого доданків відповідно,  $c_i$  та  $c_{i+1}$  – відповідно  $i$ -й та  $(i+1)$ -й розряди переносу. Тоді граф алгоритму  $n$ -розрядного додавання буде мати вигляд, показаний на рис. 6.9. Показаний на рис. 6.9. алгоритм називається алгоритмом додавання двійкових чисел з послідовним переносом, оскільки перенос проходить через всі оператори однорозрядного двійкового додавання.

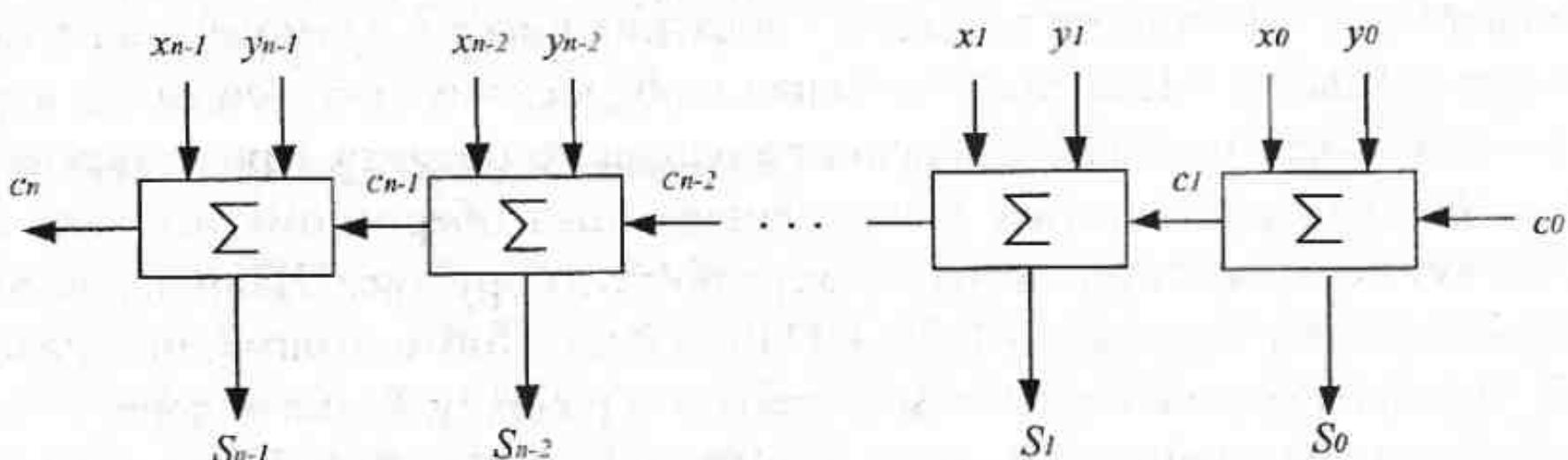


Рис. 6.9. Граф алгоритму  $n$ -розрядного додавання двійкових чисел з послідовним переносом

Вхідними даними тут є двійкові числа  $x_{n-1}, x_{n-2}, \dots, x_0$  та  $y_{n-1}, y_{n-2}, \dots, y_0$ , а вихідним – двійкове число  $s_{n-1}, s_{n-2}, \dots, s_0$ , яке є сумою вхідних чисел. Показаний алгоритм додавання є доволі простим. Недоліком представленого алгоритму є його суттєва часова складність, тобто велика кількість операцій, які знаходяться на критичному шляху, оскільки

для отримання останнього розряду суми перенос повинен бути сформованим всіма операторами однорозрядного двійкового додавання. Для зменшення кількості операцій, які знаходяться на критичному шляху, створено ряд алгоритмів додавання, в яких скорочено кількість послідовних операцій при формуванні переносу. Це здійснюється шляхом паралельного формування переносів для всіх розрядів (так званий прискорений перенос), або паралельного формування переносів для груп розрядів (так званий частково-прискорений перенос), або використання алгоритму за методом вибору переносу.

#### 6.4.2. Додавання двійкових чисел із знаками

Як ми вже бачили в попередньому розділі, існує чотири методи представлення  $n$ -розрядних чисел із знаками: в прямому, оберненому та доповняльному кодах, а також із зміщенням. В прямому коді старший розряд представляє знак числа, а наступні  $n-1$  розряди представляють модуль числа. В оберненому та доповняльному кодах додатні числа мають те ж саме представлення, що і в прямому. Представлення ж від'ємних чисел тут є іншим. В оберненому коді від'ємні числа представляються шляхом інверсії їх розрядів, а в доповняльному коді крім того до молодшого розряду оберненого коду додається одиниця. В представленні із зміщенням всі числа, як додатні так і від'ємні, додаються до зміщення і отримані суми представляються як звичайні числа без знаків. Так від'ємне число  $k$  буде представлене як  $k + b = 0$ , де  $b$  – зміщення. Типовим значенням зміщення вибирається число  $2^n - 1$ .

Приклад: використовуючи чотирирозрядну сітку ( $n = 4$ ), представимо в описаних вище кодах число  $k = -3$  (або в двійковій системі  $k = -011_2$ ). В прямому коді  $k = 1011_2$ , причому старший розряд є знаковий. В оберненому коді  $k = 1100_2$ , а в доповняльному коді  $k = 1101_2$ . Для системи із зміщенням, коли зміщення  $b = 2^n - 1 = 8$ , маємо  $k = 0101_2$ .

Додавання чисел, представлених в прямому коді, вимагає проведення початкового аналізу знаків чисел. Якщо знаки одинакові, то модулі чисел додаються, а результату присвоюється їх знак до проведення додавання. Якщо ж їх знаки різні, то модулі чисел віднімаються, а результату присвоюється знак більшого за модулем числа, або знак "+", якщо модулі чисел є рівними.

Додавання чисел, представлених в оберненому та прямому кодах, не залежить від їх знаків і проводиться так само як додавання додатних чисел в прямому коді з тією різницею, що при додаванні чисел, представлених в оберненому коді, необхідно перенос з старшого розряду подавати на вхід переносу молодшого розряду. Представлення в доповняльному коді використовується значно ширше, ніж в оберненому, оскільки при додаванні чисел тут перенос із старшого розряду просто ігнорується. Наприклад, додавши  $5 + (-2)$  в доповняльному коді маємо  $0101_2 + 1110_2 = 0011_2$ . Тобто отриманий правильний результат  $3_{10}$  при ігноруванні переносом із старшого розряду. Якщо ж додати ті ж самі числа в оберненому коді отримаємо  $0101_2 + 1101_2 + 1 = 0011_2$ . Тобто також отриманий правильний результат  $3_{10}$  при врахуванні переносу із старшого розряду. Цей перенос називається циклічним. Додавання до отриманої суми одиниці циклічного переносу не викликає повторного циклічного переносу. Наведений приклад наглядно ілюструє рис. 6.10.

$$\begin{array}{r}
 + \begin{array}{r} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array} \\
 \hline 0 & 0 & 1 & 1
 \end{array}
 \quad
 \begin{array}{r}
 + \begin{array}{r} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{array} \\
 \hline 0 & 0 & 1 & 0 \\ 1
 \end{array}$$

Рис. 6.10. Додавання чисел 5 та -2 в доповняльному (зліва) та в оберненому (справа) кодах

Крім того, потрібно зауважити, що в доповняльному коді для представлення нуля існує лише один код – всі нулі, тоді як в оберненому коді два – всі нулі та всі одиниці, що призводить до неоднозначностей.

При виконанні додавання двійкових чисел можливе переповнення, коли отримана сума перевищує діапазон представлення чисел, тобто коли вона виходить за межі розрядної сітки. Для інформування програміста про отримання неправильного результату переповнення повинно бути зафікованим. Для фіксації переповнення аналізуються знакові розряди чисел. Переповнення виникає тільки при додаванні чисел з однаковими знаками і виявити його можна порівнюючи знаки суми і доданків. При переповненні знак суми S не дорівнює знакам доданків x та y, тобто  $\text{Sign}_x = \text{Sign}_y \neq \text{Sign}_S$ .

Для спрощення фіксування наявності переповнення використовуються так звані модифіковані коди з двома знаковими розрядами (тобто 0 представляється як 00, а 1 представляється як 11). Неоднаковість цих розрядів після виконання операції означає наявність переповнення, як це показано на прикладах на рис. 6.11.

$$\begin{array}{r}
 + \begin{array}{r} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \\
 \hline 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0
 \end{array}
 \quad
 \begin{array}{r}
 + \begin{array}{r} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{array} \\
 \hline 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1
 \end{array}$$

Рис. 6.11. Приклади виконання додавання двійкових чисел з використанням модифікованих кодів з двома знаковими розрядами

В першому прикладі відбулося переповнення, оскільки два перші розряди, які представляють знак, не є однаковими.

### 6.4.3. Віднімання двійкових чисел

Віднімання можна проводити двома способами: проведенням безпосередньо віднімання розрядів чисел або додаванням двійкового коду від'ємника з протилежним знаком. Перший спосіб, як правило, використовується тоді, коли числа представлені в прямому коді. Віднімання проводиться подібно до віднімання в десятковій системі: віднімаються відповідні розряди, а при виникненні одиниці запозичення вона вираховується з старшого розряду. Віднімання можна описати такими формулами:

$$\begin{aligned}
 S_i &= x_i \text{XOR } y_i \text{XOR } b_i \\
 b_{i+1} &= (x_i \text{ AND } y_i) \text{ OR } (x_i \text{ AND } b_i) \text{ OR } (y_i \text{ AND } b_i) \\
 b_0 &= 0
 \end{aligned}$$

Ці формули отримані з наступної таблиці істинності (табл. 6.6).

Таблиця 6.6

$b_i$	$x_i$	$y_i$	$s_i$	$c_{i+1}$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Де:  $s_i$  –  $i$ -й розряд різниці,  $x_i, y_i$  –  $i$ -ті розряди зменшуваного та від'ємника відповідно,  $b_i$  – розряд запозичення.

Приклади виконання операції віднімання двійкових чисел без знаків приведено на рис. 6.12.

Рис. 6.12. Приклади виконання операції додавання двійкових чисел без знаків

Коли числа представлені в оберненому або доповняльному кодах, то можна використати інший метод: потрібно змінити знак від'ємника, всі його розряди потрібно інвертувати, а для доповняльного коду, крім того, збільшити від'ємник на одиницю молодшого розряду, і тоді просто додати до зменшуваного отримане число  $S = x + (\text{NOT}(y) + 1 \text{ м.р.})$ .

#### 6.4.4. Множення двійкових чисел

Множення може проводитись в прямому, оберненому та доповняльному кодах. Знак результата операції множення можна визначати окремо. Для цього використовується операція XOR над знаковими розрядами співмножників відповідно до табл. 6.7.

Таблиця 6.7

знак X	знак Y	Знак результата
0	0	0
0	1	1
1	0	1
1	1	0

При виконанні множення двох операндів однакової розрядності розрядність результата збільшується вдвічі, порівняно з розрядністю множників.

При виконанні множення операндів, представлених в прямому коді, їх модулі множаться як цілі двійкові числа без знаків, або як дробові числа без знаків, оскільки про-

цедура множення в обох випадках та ж сама. При виконанні множення операндів, представлені в оберненому коді, всі розряди від'ємних чисел потрібно інвертувати, а далі проводити множення так само, як над даними, представленими в прямому коді. Разом з тим, потрібно зауважити, що існують методи прямого множення операндів, представлені в обернених кодах.

#### 6.4.4.1. Множення цілих двійкових чисел без знаків

Якщо позначити множене буквою X, а множник буквою Y, причому представити Y у вигляді суми його двійкових розрядів

$$Y = \sum_{i=0}^{n-1} Y_i 2^i,$$

то результат Z множення двох цілих двійкових чисел без знаків визначається з виразу:

$$Z = XY = X \sum_{i=0}^{n-1} Y_i 2^i = XY_0 2^0 + XY_1 2^1 + \dots + XY_{n-1} 2^{(n-1)}.$$

З наведеного виразу видно, що операція множення двійкових чисел зводиться до операції логічного множення множеного (множене – перший множник) на розряди множника та підсумовування отриманих результатів з їх зсувом на кількість розрядів, рівну відповідному показнику ступеня у виразі. Граф алгоритму множення має вигляд, показаний на рис. 6.13.

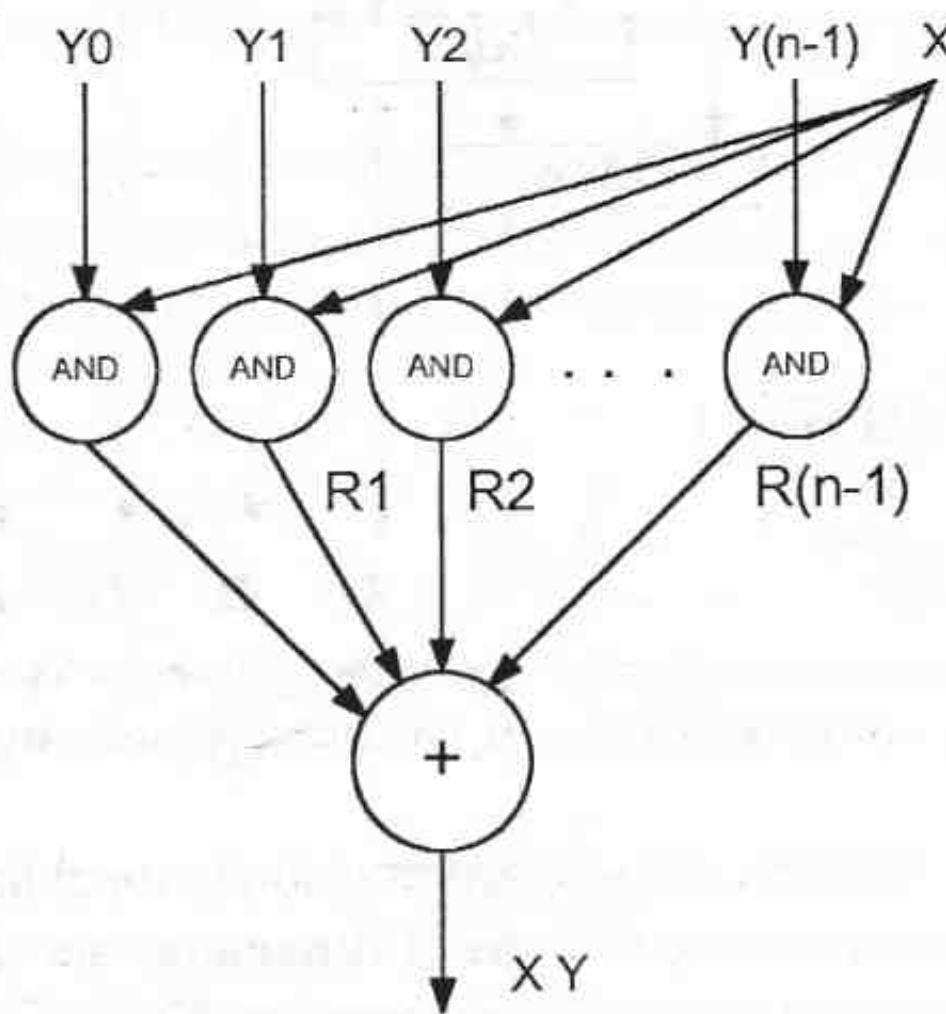


Рис. 6.13. Граф алгоритму множення

Лінійка операторів AND формує n n-розрядних результатів логічного множення множеного на розряди множника, які зсуваніся праворуч на  $R_i$  ( $i = 1, 2, \dots, n-1$ ) розрядів. Ці результати називаються частковими добутками. Оператор із знаком додавання тут означає багатомісну операцію додавання часткових добутків.

#### 6.4.4.2. Багатомісна операція додавання часткових добутків

Алгоритм виконання багатомісної операції додавання часткових добутків залежить від типу використовуваних операторів додавання. Це можуть бути зокрема оператори попарного  $n$ -розрядного додавання двох чисел, оператори попарного однорозрядного додавання двох чисел, оператори багатомісного паралельного додавання чисел і т. д. Якщо використовувати оператори попарного  $n$ -розрядного додавання двох чисел, тобто двох часткових добутків, то можуть бути запропоновані наступні алгоритми:

- алгоритм послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу молодших розрядів множника;
- алгоритм послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу старших розрядів множника;
- алгоритм паралельного попарного додавання часткових добутків з використанням структури бінарного дерева.

Граф алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу молодших розрядів множника, показаний на рис. 6.14.

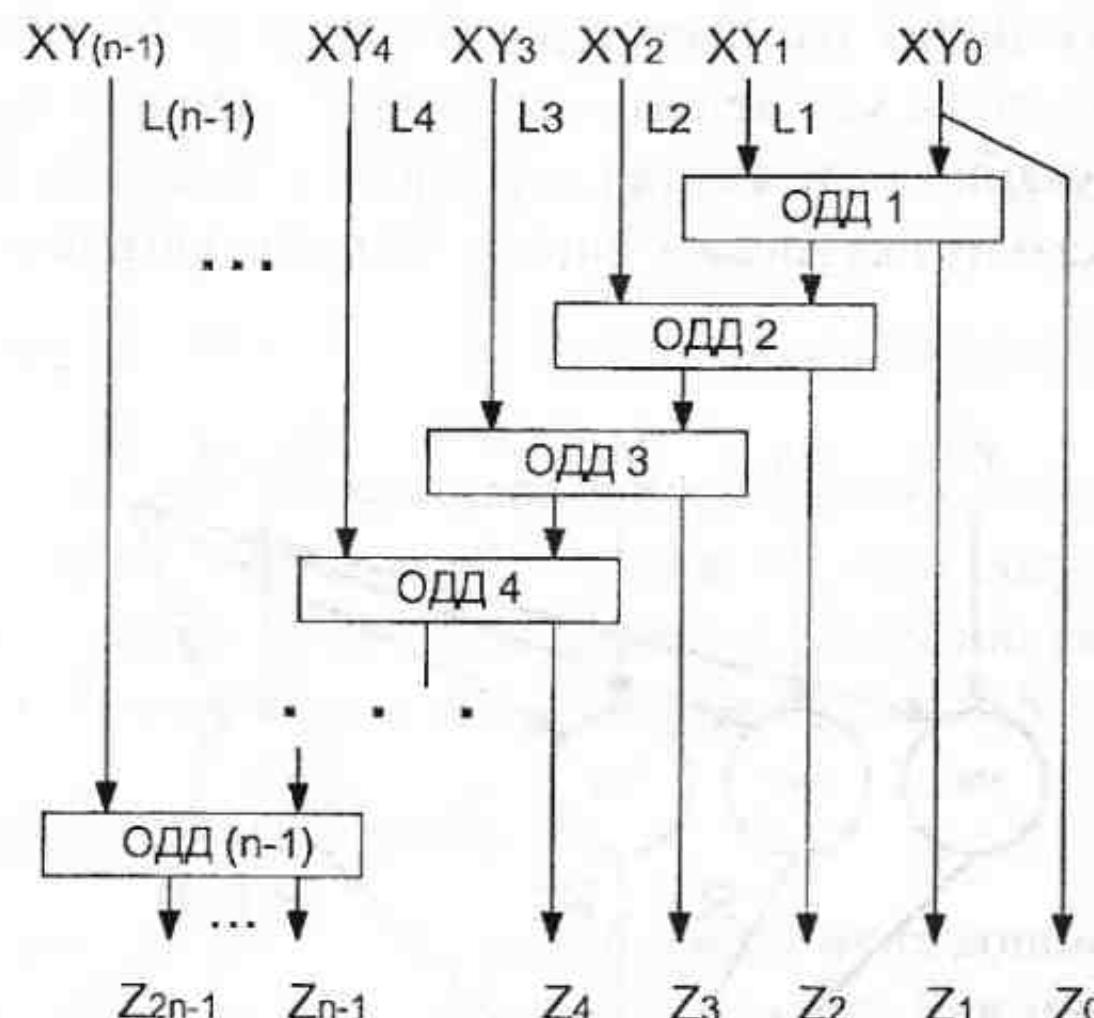


Рис. 6.14. Граф алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу молодших розрядів множника

Тут послідовно з'єднані оператори двомісного (попарного) додавання (ОДД), причому часткові добутки подаються для додавання із зміщенням на  $i$  ( $L_1, L_2, \dots, L_i, L_{(n-1)}$ ) розрядів ліворуч. Молодший розряд результату кожного  $i$ -го ОДД ( $\text{ОДД}_1, \text{ОДД}_2, \dots, \text{ОДД}_{(n-1)}$ ) є відповідним розрядом добутку  $Z_i$ , а нульовий розряд добутку є рівним молодшому розряду першого часткового добутку. Розряди добутку від  $Z_{2n-1}$ -го до  $Z_{n-1}$ -го отримуються з виходів  $(n-1)$ -го оператора ОДД, починаючи з другого виходу.

Подібним до розглянутого є алгоритм послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу старших розрядів множника, граф якого показано на рис. 6.15.

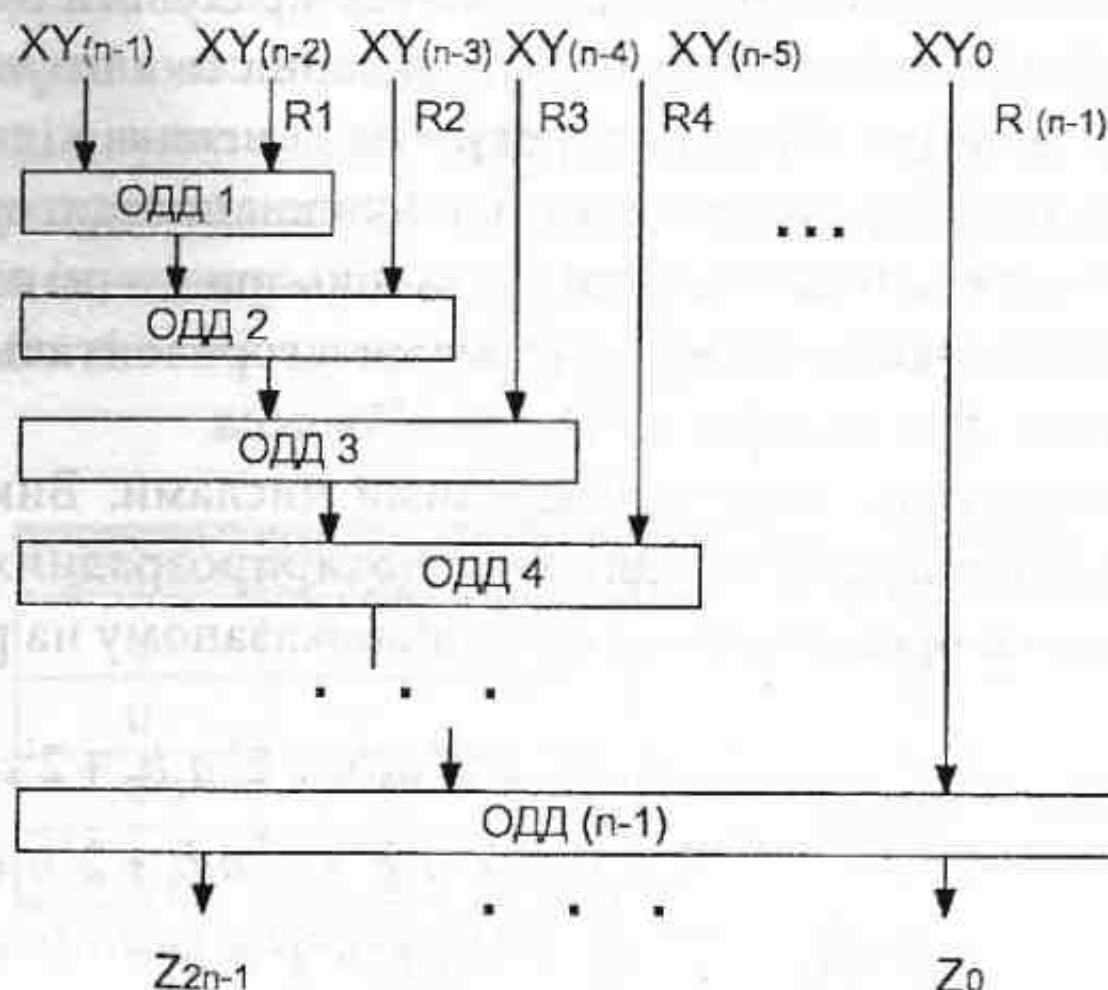


Рис. 6.15. Граф алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу старших розрядів множника

Різниця в тому, що тут часткові добутки зміщуються на  $i$  ( $R_1, R_2, \dots, R_i, R_{(n-1)}$ ) розрядів праворуч, а це вимагає розширення на один біт розрядності кожного  $i$ -го ОДД в порівнянні з  $(i-1)$ -м.

Обидва розглянуті алгоритми вимагають виконання послідовно  $n-1$  додавання, тобто послідовного виконання ОДД, причому в другому алгоритмі ОДД є обчислювально складнішими.

Алгоритм паралельного попарного додавання часткових добутків з використанням структури бінарного дерева вимагає виконання лише  $\log_2 n$  послідовних додавань за рахунок розпаралелення обчислень. Граф цього алгоритму показано на рис. 6.16.

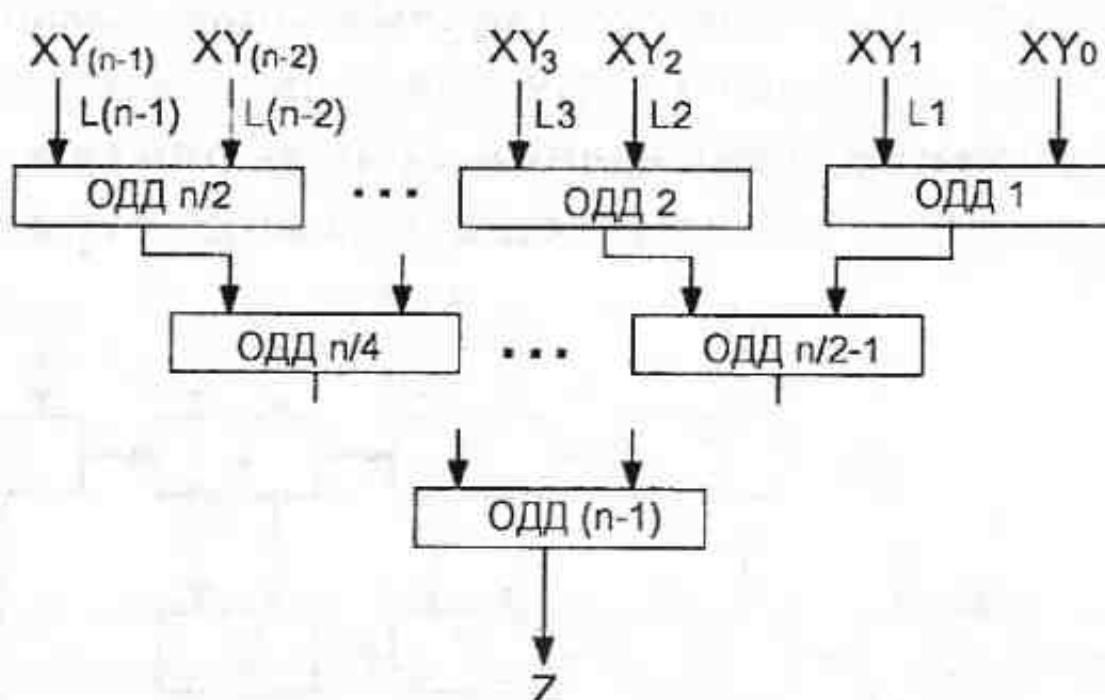


Рис. 6.16. Граф алгоритму паралельного попарного додавання часткових добутків з використанням структури бінарного дерева

При цьому на рис. 6.16 часткові множники зміщуються ліворуч, як на рис. 6.14, хоча їх можна змістити і праворуч, як на рис. 6.15.

Найчастіше в якості операторів додавання в алгоритмах виконання багатомісної операції додавання часткових добутків використовується оператор двомісного однороз-

рядного двійкового додавання. Це дозволяє повніше врахувати особливості порозрядної обробки даних та зменшити обчислювальну складність алгоритмів, тобто кількість виконуваних операцій. Для цього випадку розроблена велика кількість алгоритмів додавання часткових добутків. Нижче розглянуті найуживаніші алгоритми. Найвідоміши ми серед матричних алгоритмів, які базуються на використанні операторів двомісного однорозрядного двійкового додавання, є алгоритми з горизонтальним та діагональним розповсюдженням переносу та алгоритми Дадда і Уоллеса.

Для наочності обмежимося чотирирозрядними числами. Виконання багатомісної операції додавання часткових добутків для двох чотирирозрядних чисел  $A$  та  $B$ , де  $A = a_3a_2a_1a_0$  і  $B = b_3b_2b_1b_0$  можна представити у вигляді, показаному на рис. 6.17.

$$\begin{array}{r}
 2^3 a_3 b_0 + 2^2 a_2 b_0 + 2^1 a_1 b_0 + 2^0 a_0 b_0 \\
 2^4 a_3 b_1 + 2^3 a_2 b_1 + 2^2 a_1 b_1 + 2^1 a_0 b_1 \\
 2^5 a_3 b_2 + 2^4 a_2 b_2 + 2^3 a_1 b_2 + 2^2 a_0 b_2 \\
 \hline
 2^6 a_3 b_3 + 2^5 a_2 b_3 + 2^4 a_1 b_3 + 2^3 a_0 b_3 \\
 \hline
 2^7 Z_7 + 2^6 Z_6 + 2^5 Z_5 + 2^4 Z_4 + 2^3 Z_3 + 2^2 Z_2 + 2^1 Z_1 + 2^0 Z_0
 \end{array}$$

Рис. 6.17. Виконання багатомісної операції додавання часткових добутків для двох чотирирозрядних чисел

Перший рядок даного виразу представляє перший частковий добуток, другий – другий, і так далі. Окремі часткові добутки  $a_i b_j$ , як це було показано раніше на рис. 6.13, отримані за допомогою елементів AND. Додавання часткових добутків зводиться до додавання розрядів часткових добутків з одинаковими вагами (по стовпцях) з врахуванням переносів з молодших розрядів.

На рис. 6.18 зображено граф алгоритму виконання багатомісної операції додавання часткових добутків з горизонтальним розповсюдженням переносу при використанні операторів однорозрядного двійкового додавання, який синтезований на основі графа алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу молодших розрядів множника, який показаний на рис. 6.14.

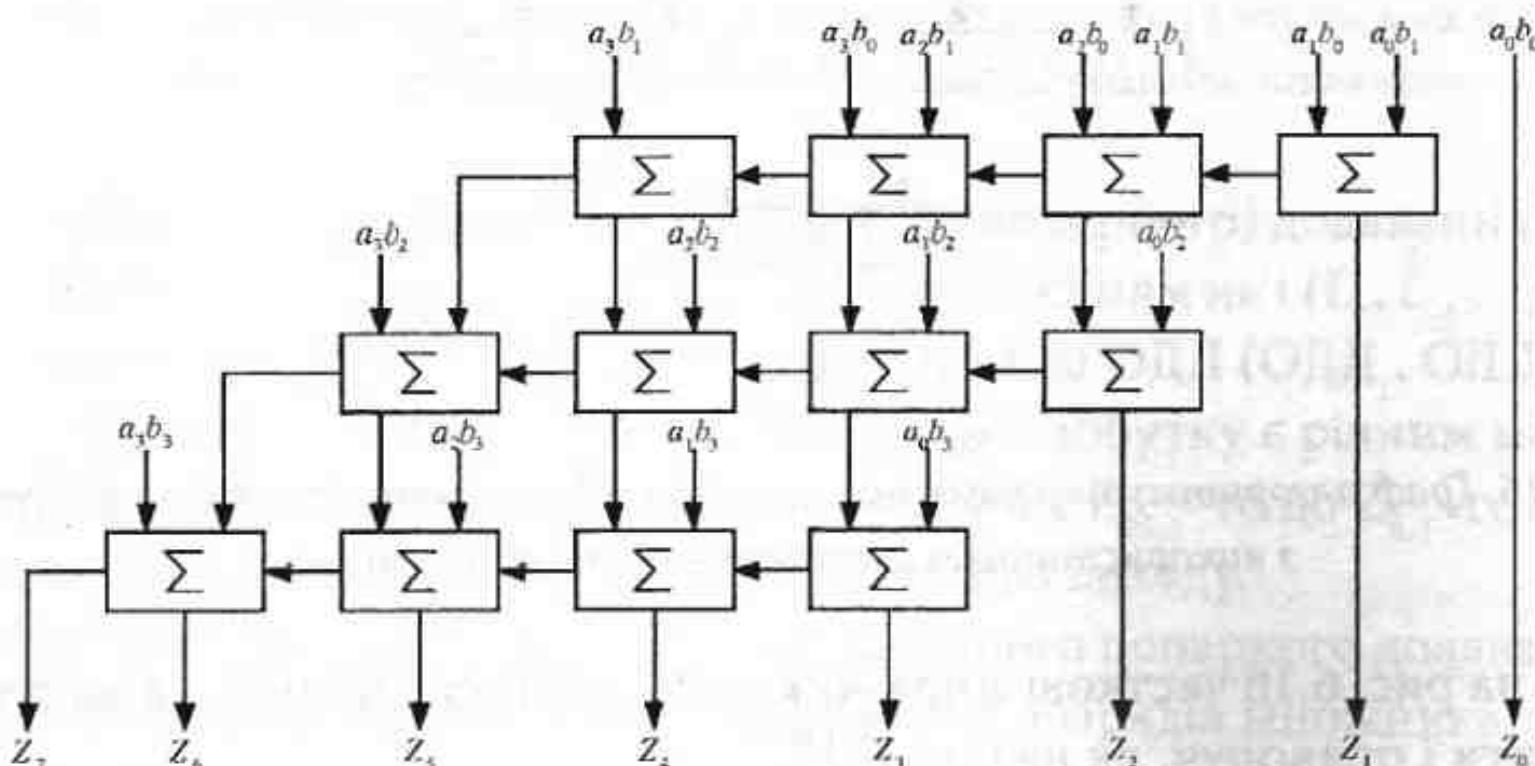


Рис. 6.18. Граф алгоритму паралельного матричного виконання багатомісної операції додавання часткових добутків з горизонтальним розповсюдженням переносу

В наведеному алгоритмі знаком суми з трьома входами позначено оператор виконання повного однорозрядного двійкового додавання, представлений на рис. 6.8, де  $a_i, b_i$  – вхідні дані;  $Z_i$  – сума, а знаком суми з двома входами (три крайні зліва оператори) позначено оператор виконання неповного однорозрядного двійкового додавання, коли відсутній вхідний перенос, що спрощує алгоритм додавання. Цей оператор виконує операції відповідно до табл. 6.8. На рис. 6.18 переноси не позначено з метою спрощення сприйняття.

Таблиця 6.8

$x_i$	$y_i$	$Z_i$	$c_{i+1}$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Його роботу можна описати наступними формулами:

$$Z_i = x_i \text{XOR } y_i$$

$$c_{i+1} = x_i \text{AND } y_i$$

Обчислювальна складність даного алгоритму рівна:  $n^2 - n$  операцій двомісного однорозрядного двійкового додавання ( $n^2 - 2n$  операцій повного та  $n$  операцій неповного двомісного однорозрядного двійкового додавання). Кількість операторів двомісного однорозрядного двійкового додавання на критичному шляху рівна  $3n - 4$ .

Виконання багатомісної операції додавання часткових добутків з діагональним розповсюдженням переносу показано на рис. 6.19. Цей алгоритм також синтезований на основі графа алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу молодших розрядів множника, який показаний на рис. 6.14.

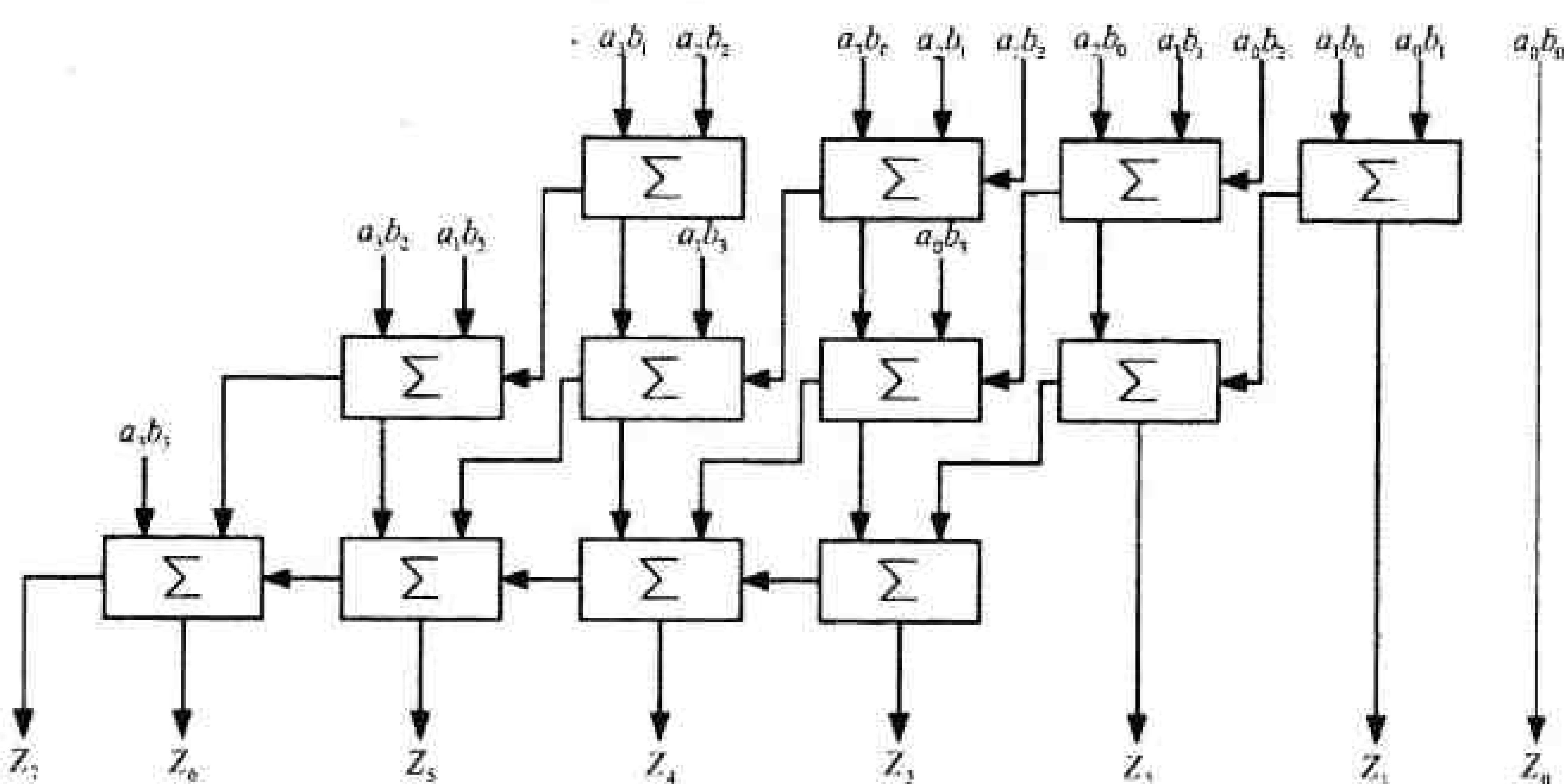


Рис. 6.19. Граф алгоритму паралельного матричного виконання багатомісної операції додавання часткових добутків з діагональним розповсюдженням переносу

Обчислювальна складність даного алгоритму рівна:  $n^2 - n$  операцій двомісного однорозрядного двійкового додавання ( $n^2 - 2n$  операцій повного та  $n$  операцій неповного двомісного однорозрядного двійкового додавання). Кількість операторів двомісного однорозрядного двійкового додавання на критичному шляху рівна  $2n - 2$ .

Подібним чином можна побудувати алгоритми виконання багатомісної операції додавання часткових добутків з послідовним та діагональним розповсюдженням переносу

відповідно до графа алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу старших розрядів множника, показаного на рис. 6.15.

На рис. 6.20 показано граф алгоритму Дадда виконання багатомісної операції додавання часткових добутків, побудований на основі графа алгоритму паралельного попарного додавання часткових добутків з використанням структури бінарного дерева (рис. 6.16).

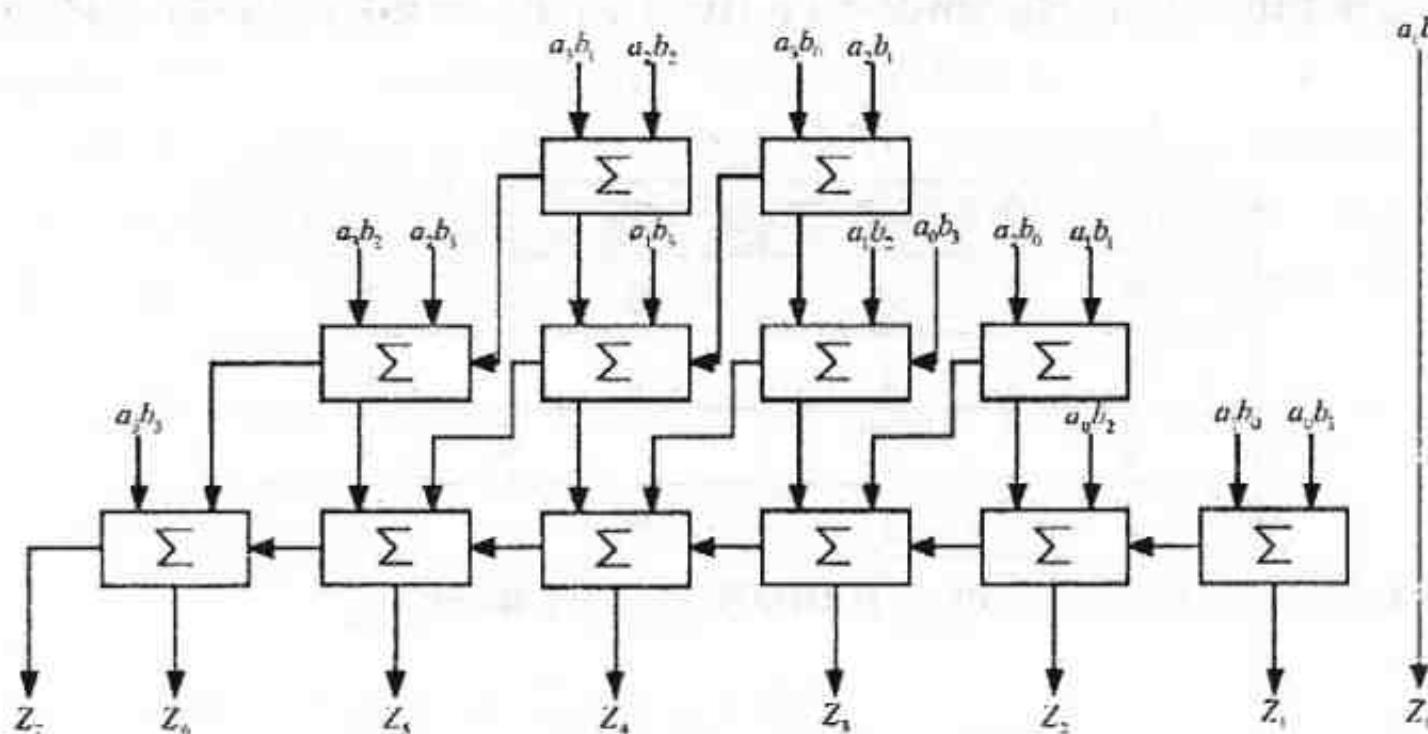


Рис. 6.20. Граф алгоритму Дадда виконання багатомісної операції додавання часткових добутків

Обчислювальна складність даного алгоритму рівна:  $n^2 - n$  операцій двомісного однорозрядного двійкового додавання ( $n^2 - 2n$  операцій повного та  $n$  операцій неповного двомісного однорозрядного двійкового додавання). Кількість операторів двомісного однорозрядного двійкового додавання на критичному шляху рівна  $2n - 2$ .

#### 6.4.4.3. Множення двійкових чисел із знаками

При множенні чисел, представлених в оберненому коді, найпростіше перевести множене і множник в прямий код та виконати операції в цьому представленні, а при від'ємному результаті провести інвертування його значення.

Існує метод множення двійкових чисел, представлених в доповняльному коді, без перетворення в прямий код. В цьому випадку множення виконується за формулою:

$$Z = XY = X \sum_{i=0}^{n-1} (Y_{n-i+1} - Y_{n-i}) 2^i = X(Y_1 - Y_0)2^0 + X(Y_2 - Y_1)2^1 + \dots + X(Y_{n-1} - Y_{n-2})2^{(n-1)}$$

Значення розряду множника, на який здійснюється множення на  $i$ -му кроці, визначається з виразу:

$$Y_{i+1} - Y_i = \begin{cases} 0, & \text{якщо } Y_i = Y_{i+1}, \\ 1, & \text{якщо } Y_i = 0, Y_{i+1} = 1, \\ -1, & \text{якщо } Y_i = 1, Y_{i+1} = 0. \end{cases}$$

При цьому добуток формується відповідно до виразу:

$$\begin{aligned} Z = XY &= X(-Y_0 + Y_1(2^0 - 2^1) + Y_2(2^1 - 2^2) + \dots + Y_i(2^{-(i-1)} - 2^{-i}) + \dots + Y_{n-1}(2^{-(n-2)} - 2^{-(n-1)})) \\ &= (-Y_0 2^0 + \sum_{i=1}^{n-1} Y_i 2^{-i}) X. \end{aligned}$$

Результат множення також отримується в доповняльному коді. На основі даного виразу та використання підходу, застосованого в п. 6.4.4.2, можна побудувати графи відповідних алгоритмів множення двійкових чисел в доповняльному коді.

#### 6.4.4.4. Прискорене множення двійкових чисел за методом Бута

Особливістю цього методу є те, що аналізуються відразу два розряди множника. Цей метод широко застосовується. Пришвидшення досягається тим, що при обрахуванні добутку за цим методом зменшується кількість додавань. Суть методу можна виразити наступними формулами для обчислення часткових добутків:

$$\begin{cases} Z_{i+1} = \frac{Z_i + X}{2}, & \text{якщо } Y_{(i,i-1)} = "01"; \\ Z_{i+1} = \frac{Z_i - X}{2}, & \text{якщо } Y_{(i,i-1)} = "10"; \\ Z_{i+1} = \frac{Z_i}{2}, & \text{якщо } Y_{(i,i-1)} = "00", "11"; \end{cases}$$

$$Z_0 = 0; \quad i = \overline{0, n-1}; \quad Y_{(-1)} = 0; \quad Z_n = Z = X \cdot Y,$$

де: X, Y, Z – множене, множник і добуток відповідно, Zi – сума часткових добутків на i-му етапі, Y(i, i-1) – i-й та i-1 розряди множника, n – кількість розрядів операндів X та Y без врахування знакового розряду.

Можна проілюструвати цей алгоритм за допомогою блок-схеми, показаної на рис. 6.21.

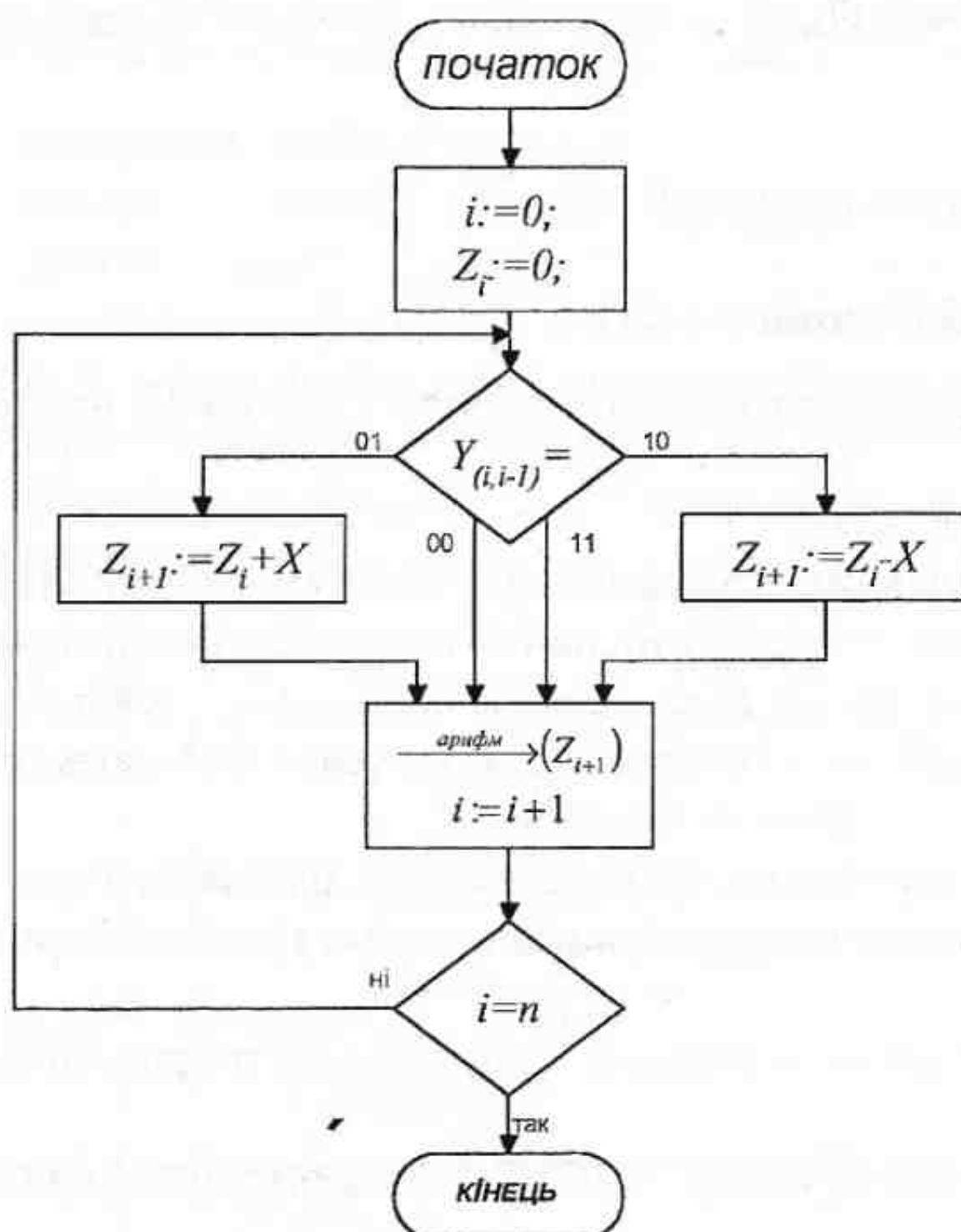


Рис. 6.21. Блок-схема множення двійкових чисел за методом Бута

Тут знаком  $\xrightarrow{\text{арифм}}$  позначена операція арифметичного зсуву праворуч.

Тепер розглянемо приклад:

$$X=0101\ 0101; Y=01101011.$$

Результати проміжних обчислень наведені в табл. 6.9.

Таблиця 6.9

i	$Z_i$	$Y(Y_{(n-i)}) Y_{(-1)}$	операція	$Z_{i+1}$
0	0000 0000 0000 0000	0110 1011  0	$Z_{i+1} = \frac{Z_i - X}{2}$	1101 0101 1000 0000
1	1010 1011 0000 0000	0110 10 11 0	$Z_{i+1} = \frac{Z_i}{2}$	1110 1010 1100 0000
2	1110 1010 1100 0000	0110  1011 0	$Z_{i+1} = \frac{Z_i + X}{2}$	0001 1111 1110 0000
3	0001 1111 1110 0000	01 10 1011 0	$Z_{i+1} = \frac{Z_i - X}{2}$	1110 0101 0111 0000
4	1110 0101 0111 0000	0110 1011  0	$Z_{i+1} = \frac{Z_i + X}{2}$	0001 1101 0011 1000
5	0001 1101 0011 1000	0110 1011  0	$Z_{i+1} = \frac{Z_i - X}{2}$	1110 0100 0001 1100
6	1110 0100 0001 1100	0110 1011  0	$Z_{i+1} = \frac{Z_i}{2}$	1111 0010 0000 1110
7	1111 0010 0000 1110	0110 1011  0	$Z_{i+1} = \frac{Z_i + X}{2}$	0010 0011 1000 0111

Таким чином

$$0101\ 0101 \cdot 01101011 = 0010\ 0011\ 1000\ 0111.$$

#### 6.4.5. Ділення двійкових чисел

Ділення  $X:Y$  передбачає визначення частки  $Q$  і залишку  $R$  відповідно до рівності  $X = Q * Y + R$ ,

де  $X$  – ділене,  $Y$  – дільник.

Алгоритм ділення в комп'ютерах виконується як послідовність операцій віднімання дільника  $Y$  від часткового залишку  $R_i$ , початкове значення якого рівне значенню діленого  $X$ . Знаковий розряд частки визначається за знаками діленого і дільника аналогічно операції множення (табл. 6.7). Приймемо, що значення  $X$  і  $Y$  завжди відповідають вимозі  $X < Y$ .

Існує два варіанти виконання операції ділення двійкових чисел:

- зі зсувом залишків ліворуч відповідно до рекурентної формули

$$R_i = 2(R_{i-1} - p_i Y),$$

де  $p_i = \text{Sign}(R_{i-1} - Y)$ ,  $R_0 = X$ ,  $i=1,2,\dots,n$ . При цьому  $i$ -й розряд частки визначається з виразу  $q_i = \text{NOT}(p_i)$ ;

- зі зсувом дільника праворуч відповідно до рекурентної формули

$$R_i = R_{i-1} - q_{i-1} Y 2^{-1},$$

де  $R_0 = X$ ,  $i=1,2,\dots,n$ ,  $q_i$  –  $i$ -й розряд частки, причому  $q_i = \text{Sign } R_i$ .

На  $n$ -му кроці ділення в обох випадках визначається значення  $n$ -го розряду частки та залишок.

Блок-схема алгоритму ділення за першим варіантом, який має ширше використання, наведена на рис. 6.22.

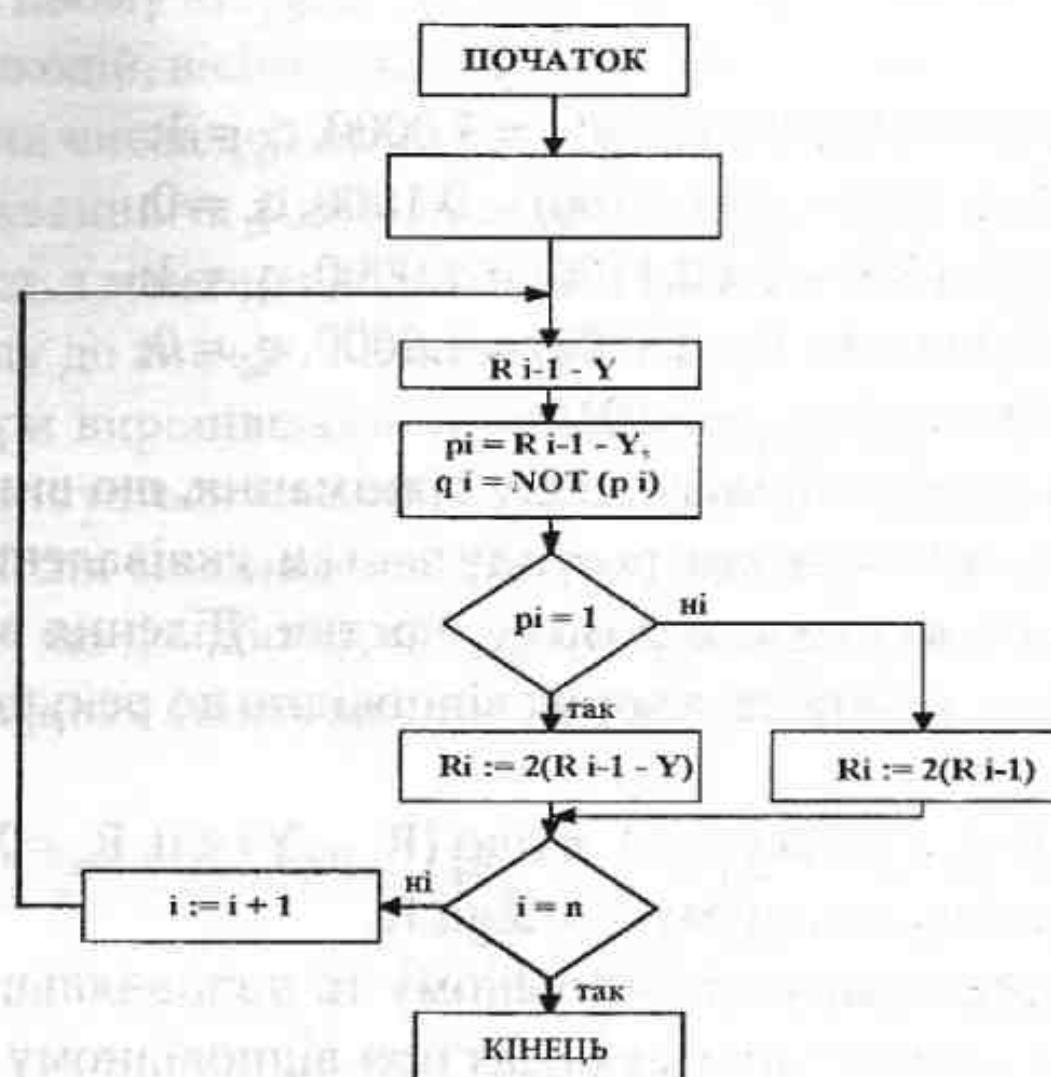
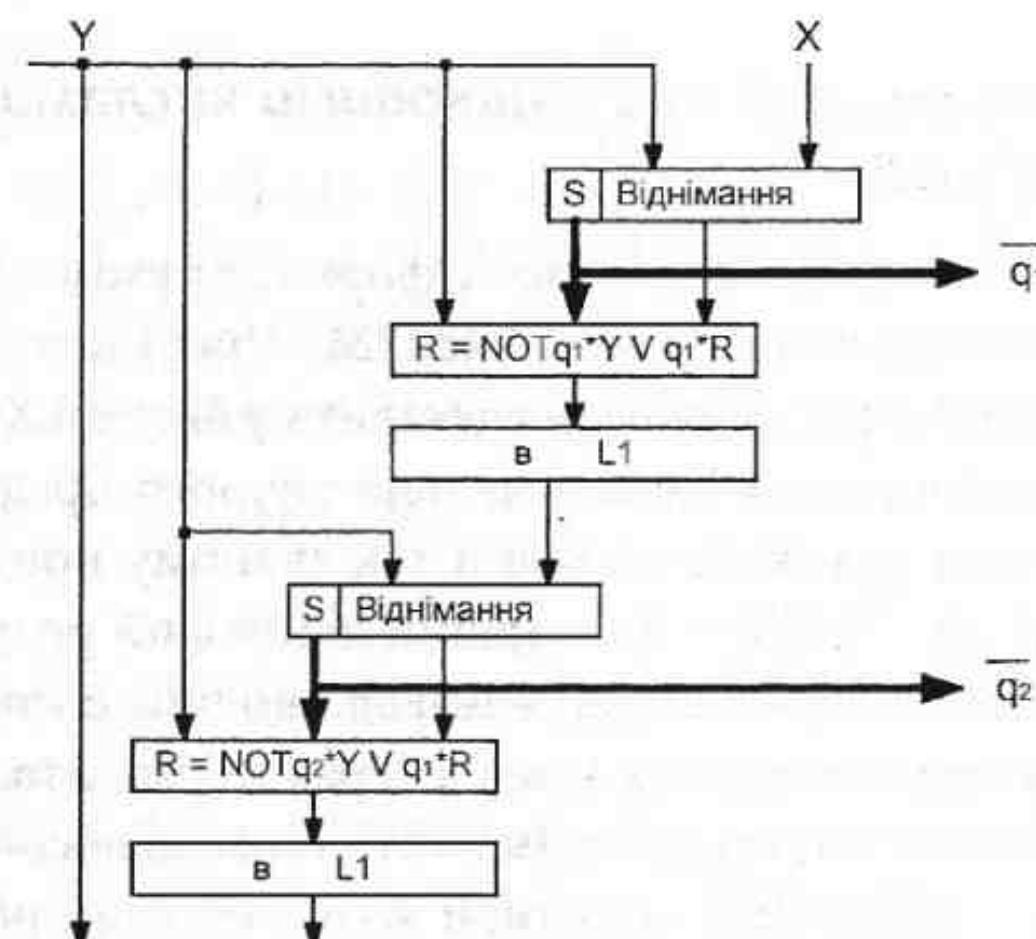


Рис. 6.22. Блок-схема алгоритму ділення

Ділення може бути виконано двома способами: з відновленням і без відновлення залишку.

При діленні з відновленням залишку послідовно віднімається дільник від діленого і проводиться аналіз значення поточного залишку. Якщо після чергового віднімання залишок позитивний, то відповідний розряд частки рівний одиниці. При від'ємному залишку розряд частки рівний нулю. В цьому випадку виконується коригуюче збільшення дільника до поточного залишку (відновлення залишку), він зсувається на один розряд ліворуч і процес повторюється. Обчислення проводяться за формулою першого варіанту, де  $q_i = 1$ , якщо  $R_i - Y >= 0$  і  $q_i = 0$ , якщо  $R_i - Y < 0$ . На рис. 6.23 наведено два перші яруси графа, описаного алгоритму ділення.



6.23

Розглянемо приклад. Нехай  $X = 0.1000_2 = 0.5_{10}$  і  $Y = 0.1100_2 = 0.75_{10}$ . Тут виконана умова  $X < Y$ . Поетапні результати виконання ділення відповідно до блок-схеми алгоритму (рис. 6.22) наступні:

$$R_0 = X = 0.1000;$$

$$R_1 = 2(R_0 - p_1 \times Y) = 2(0.1000 - 0 \times 0.1100) = 1.0000, q_1 = 1;$$

$$R_2 = 2(R_1 - p_2 \times Y) = 2(1.0000 - 1 \times 0.1100) = 0.1000, q_2 = 0;$$

$$R_3 = 2(R_2 - p_3 \times Y) = 2(0.1000 - 0 \times 0.1100) = 1.0000, q_3 = 1;$$

$$R_4 = 2(R_3 - p_4 \times Y) = 2(1.0000 - 0 \times 0.1100) = 1.0000, q_4 = 0;$$

Таким чином маємо  $Q = q_1 q_2 q_3 q_4 = 0.1010$ .

У методі ділення без відновлення залишку віднімання, що викликало появу від'ємного залишку при визначенні значення розряду частки, еквівалентне двом відніманням при визначенні сусіднього молодшого розряду частки. Ділення виконується за тим же співвідношенням зі зсувом залишків ліворуч відповідно до рекурентної формули

$$R_i = 2(R_{i-1} - p_i Y),$$

де  $p_i = 1$ , якщо  $(R_{i-1} - Y) \geq 0$ , та  $p_i = -1$ , якщо  $(R_{i-1} - Y) < 0$ ,  $R_0 = X$ ,  $i=1,2,\dots,n$ . При цьому  $i$ -й розряд частки визначається з виразу  $q_i = \text{Sign } R_i$ .

Ділення чисел, які представлені в оберненому та доповняльному кодах, можна виконувати за правилами ділення в прямих кодах при відповідному перетворенні операндів в цей код. Розглянемо виконання операції ділення без відновлення залишків чисел, представлених доповняльними кодами. У цьому випадку ділення зводиться до послідовності додавань і віднімань дільника від діленого, а потім часток і часткових залишків і їх зсуву на розряд ліворуч за наступними виразами:

$$R_i = 2R_{i-1} - Y, \text{ при } \text{Sign } R_i = \text{Sign } Y;$$

$$R_i = 2R_{i-1} + Y, \text{ при } \text{Sign } R_i \neq \text{Sign } Y,$$

для  $i=1,2,\dots,n+1$  при  $2R_0 = X$  та  $\text{Sign } R_0 = \text{Sign } X$ . Тут  $\text{Sign } R_i$  – знак  $i$ -го часткового залишку,  $\text{Sign } Y$ ,  $\text{Sign } X$  – відповідно знаки  $Y$  та  $X$ .

При цьому розряди частки визначаються наступним чином:

$$q_i = 1, \text{ якщо } \text{Sign } R_i = \text{Sign } Y,$$

$$q_i = 0, \text{ якщо } \text{Sign } R_i \neq \text{Sign } Y.$$

Операція виконується за  $n+1$  кроків і дає значення частки в доповняльному коді.

#### 6.4.6. Арифметичні операції над двійковими числами у форматі з рухомою комою

Як вже було показано в попередньому розділі, формат з рухомою комою передбачає наявність двох частин числа – порядку ( $P$ ) та мантиси ( $M$ ). Числа  $X$  та  $Y$ , які мають відповідно мантиси  $M_x$  та  $M_y$  і порядки  $P_x$  та  $P_y$ , можна представити у вигляді  $X = M_x 2^{P_x}$ ,  $Y = M_y 2^{P_y}$ .

Для забезпечення однозначного і максимально точного представлення чисел прийнято представляти число з рухомою комою в так званому нормалізованому вигляді. Якщо виконується нерівність  $1 \leq |M| < 2$  (старший двійковий розряд мантиси дорівнює 1), то відповідно до стандарту IEEE-754 вважається, що число представлене в нормалізованому вигляді (в багатьох попередніх комп’ютерах нормалізованим вважалося число, коли  $0.5 \leq |M| < 1$ ). Таким чином, у двійкового нормалізованого числа у форматі з рухомою комою мантиса є двійковим числом, в якому перший розряд рівний одиниці, після якої розміщено дробове число (або лише дробове число, в старшому розряді якого

завжди стоять 1). Потрібно зауважити, що ці твердження відносяться до двійкових чисел у форматі з рухомою комою, основою порядку яких є число 2. Якщо основою порядку є числа 4, 8, 16 і т. д., то в цьому випадку число є нормалізованим, коли ненульовим є його перший розряд в четвірковій, вісімковій або шістнадцятковій системі числення відповідно. Операція приведення числа до нормалізованого вигляду називається нормалізацією.

При виконанні додавання та віднімання двійкових чисел з рухомою комою порядки операндів вирівнюються, а мантиси додаються. Порядки вирівнюються збільшенням порядку меншого операнду до значення порядку більшого операнду. Цей порядок є порядком результату. Щоб при вирівнюванні порядків величина операнда не змінилася, його мантиса одночасно зменшується шляхом зсуву вправо на відповідну збільшенням порядку кількість розрядів. Після виконання вирівнювання порядків виникають додаткові молодші розряди мантиси, які до, або після, додавання відкидаються чи заокруглюються.

Додавання та віднімання двійкових чисел з рухомою комою можна представити виразом:

$$S = X \pm Y = M_X \cdot 2^{P_X} \pm M_Y \cdot 2^{P_Y} = (M_X \cdot 2^t \pm M_Y) \cdot 2^{P_Y} = (M_X \pm M_Y \cdot 2^{-t}) \cdot 2^{P_X},$$

$$t = P_X - P_Y$$

Наведемо приклад:

$$M_X = 1.000\ 1000; P_X = 0110; M_Y = 1.110\ 1001; P_Y = 0100.$$

Оскільки перший операнд більший, порядок числа Y приводиться до порядку числа X:  $M_Y = 0.011\ 1010; P_Y = 0110$ .

Далі мантиси додаються:

$$M_S = M_X + M_Y = 1.000\ 1000 + 0.011\ 1010 = 10.0000010; P_S = 0110.$$

$$\text{Тепер проводиться нормалізація: } M_S = 1.000\ 0001; P_S = 0111.$$

Множення чисел у форматі з рухомою комою описується наступним співвідношенням:

$$D = X \cdot Y = M_X \cdot 2^{P_X} \cdot M_Y \cdot 2^{P_Y} = M_X \cdot M_Y \cdot 2^{P_X + P_Y}$$

При виконанні множення порядки операндів додаються, а мантиси перемножуються. Після перемноження мантис виникають додаткові молодші розряди мантиси, які відкидаються або заокруглюються.

Розглянемо приклад:

$$M_X = 1.000\ 1000; P_X = 0110; M_Y = 1.110\ 1001; P_Y = 0100.$$

$$\text{Тоді } M_D = 1.000\ 1000 \cdot 1.110\ 1001 = 0.111\ 1011\ 1100\ 1000; P_D = 0110 + 0100 = 1010.$$

$$\text{Після нормалізації: } M_D = 1.111\ 0111; P_D = 1001.$$

Ділення чисел у форматі з рухомою комою описується наступним співвідношенням:

$$D = \frac{X}{Y} = \frac{M_X \cdot 2^{P_X}}{M_Y \cdot 2^{P_Y}} = \frac{M_X}{M_Y} \cdot 2^{P_X - P_Y}$$

При виконанні ділення порядки операндів віднімаються, а мантиси діляться.

Розглянемо приклад:

$$M_X = 1.111\ 1100; P_X = 0111; M_Y = 1.100\ 0000; P_Y = 0011.$$

$$\text{Тоді } M_D = 1.111\ 1100 / 1.100\ 0000 = 0.10101; P_D = 0111 - 0011 = 0100.$$

$$\text{Після нормалізації: } M_D = 1.010\ 1000; P_D = 0101.$$

## 6.5. Операції обчислення елементарних функцій

Значна частина задач обробки даних розв'язується з використанням операцій обчислення тригонометричних і гіперболічних функцій, функцій типу  $\exp x$ ,  $\ln x$ ,  $x^2$  та інших, операцій повороту вектора і перетворення координат з однієї системи в іншу, наприклад, з декартової в полярну. При цьому в деяких областях, наприклад, в радіонавігації, сейсморозвідці, гідроакустіці, вказані операції займають основну частину обчислень. Вирішення таких задач вимагає значних витрат часу Тому спочатку в спеціалізованих, а останнім часом і в універсальних комп'ютерах, до складу системи команд вводять операції обчислення елементарних функцій

В процесі розробки комп'ютерів та математичних методів обчислень було створено велику кількість методів обчислення елементарних функцій. Найчастіше в універсальних комп'ютерах використовується обчислення елементарних функцій за допомогою різного виду апроксимуючих виразів та аналітичних ітераційних методів

### 6.5.1. Розклад функції в ряд та використання ітеративних обчислень

Для обчислення елементарних функцій часто використовується їх розклад в нескінчений ряд, який містить лише елементарні арифметичні та логічні операції, або використовуються ітеративні співвідношення. Кількість членів ряду чи ітерацій вибирається залежно від необхідної точності, яка до того ж обмежується розрядною сіткою.

Для прикладу можна навести обчислення квадратного кореня за допомогою ітеративної формули Ньютона

$$A_{i+1} = (N/A_i + A_i)/2,$$

де:  $A_i$  –  $i$ -те наближення.  $A_0$  можна вибрати довільним,  $N$  – число, з якого треба добути корінь.

Щоб перевірити правильність попробуємо добути корінь з 36:

$$\begin{aligned} A_0 &= 1 \text{ (приймаємо випадкове число)} \\ A_1 &= (36/1 + 1)/2 = 18.5 \\ A_2 &= (36/18.5 + 18.5)/2 = 10.2229 \\ A_3 &= (36/10.2229 + 10.2229)/2 = 6.8722 \\ A_4 &= (36/6.8722 + 6.8722)/2 = 6.0553 \\ A_5 &= (36/6.0553 + 6.0553)/2 = 6.0002 \\ A_6 &= (36/6.0002 + 6.0002)/2 = 6.0000 \end{aligned}$$

### 6.5.2. Обчислення елементарних функцій методом "цифра за цифрою"

Протягом останніх десятиліть багато робіт було присвячено ефективному ітераційному алгоритму обчислення елементарних функцій, який найчастіше називають методом "цифра за цифрою". Обчислення елементарних функцій методом "цифра за цифрою" зводиться до виконання двох етапів. На першому етапі аргумент представляється або у вигляді суми п доданків  $\ln(1+f_i a^{-i})$ , або  $f_i \arctg a^{-i}$ , або  $f_i \operatorname{arth} a^{-i}$ , або в вигляді добутку п спів множників  $(1-f_i a^{-i})$ . Тут  $i$  – номер ітерації. На цьому етапі визначаються значення  $f_i$ ,

які можуть бути рівні 0,1 або +1, -1. У першому випадку говорять про ітераційний процес із знакопостійними приростами, в другому – із знакозмінними. На другому етапі, на підставі знайдених на першому етапі значень, визначається величина елементарної функції шляхом додавання або множення констант. Одночасне виконання двох описаних етапів методу “цифра за цифрою” називають методом Волдера, а послідовне – методом Меджіта. Відміна методів Волдера і Меджіта, що полягає у величинах ітераційних кроків, значеннях констант і послідовності виконання етапів, істотним чином впливає на точність, швидкодію і структуру операційного пристроя. Детальний аналіз показав, що метод Меджіта має вищу точність, ніж метод Волдера, проте швидкодія його реалізації нижча.

З урахуванням отриманих Вальтером результатів по уніфікації методу і шляхом об'єднання одержаних для різних функцій алгоритмів, єдиний обчислювальний алгоритм “цифра за цифрою” в двійковій позиційній системі числення можна представити в наступному вигляді:

$$\begin{aligned} X_{i+1} &= X_i - p f_i Y_i 2^{-i-1} \\ Y_{i+1} &= Y_i + f_i X_i 2^{-i-1} \\ Z_{i+1} &= Z_i - f_i C_i \\ f_i &= -R \operatorname{Sign} Y_i + (1-R) \operatorname{Sign} Z_i \\ C_i &= [(1+p)/2 \operatorname{arctg} 2^{-i-1} + (1+|p|) \operatorname{arth} 2^{-i-1}] (1-p) / 2 [2^{-i-1}]; \\ i &= 0, 1, 2, 3, 3, 4, \dots, n-1. \end{aligned}$$

Тут  $\lfloor h \rfloor$  – ціла частина від  $h$ ;  $|p|$  – модуль  $p$ ;  $f_i$  – двійкові оператори, що приймають значення +1 або -1 залежно від знаку  $Y_i$  або  $Z_i$ ,  $i$  – номер ітерації, причому, ітерації 3, 12, ...,  $k$ , ...,  $3(k-1)$ , ... повторюються двічі. Параметр  $p$  визначає тип обчислюваних функцій:  $p = 0$  – лінійні,  $p = 1$  – тригонометричні,  $p = -1$  – гіперболічні;  $R$  – параметр, що визначає від чого залежить значення  $f_i$ : якщо  $R = 0$ , то  $f_i = \operatorname{sign} Z_i$ , якщо  $R = 1$ , то  $f_i = \operatorname{sign} Y_i$ ;  $C_i$  – константи. В табл. 6.10 представлені функціональні можливості розглянутого алгоритму при різних значеннях параметрів  $p$ ,  $R$  і різних початкових умовах  $X_0$ ,  $Y_0$ ,  $Z_0$ . Тут  $K_t$  – коефіцієнти деформації відповідно тригонометричного і гіперболічного векторів, рівні:  $K_t = (\prod (1+2^{-2(i+1)}))^{1/2}$ ;  $K_g = (\prod (1-2^{-2(i+1)}))^{1/2}$ , де знаком  $\prod$  позначено добуток чисел, взятих в дужки, при  $i = 0, 1, 2, \dots, n-1$ .

Таблиця 6.10

$X_0$	$X_0$	$Z_0$	$P$	$R$	$X_n$	$Y_n$	$Z_n$
$X$	$Y$	$Z$	1	0	$K_t(X \cos Z - Y \sin Z)$	$K_t(Y \cos Z + X \sin Z)$	0
$X$	$Y$	$Z$	-1	0	$K_t(X \operatorname{Ch} Z - Y \operatorname{Sh} Z)$	$K_t(Y \operatorname{Ch} Z + X \operatorname{Sh} Z)$	0
$1/K_t$	$1/K_t$	$Z$	-1	0	$\operatorname{Exp} Z$	$\operatorname{Exp} Z$	0
$X$	$Y$	$Z$	0	0	$X$	$X + YZ$	0
$1/K_t$	0	$Z$	1	0	$\cos Z$	$\sin Z$	0
$1/K_t$	0	$Z$	-1	0	$\operatorname{Ch} Z$	$\operatorname{Sh} Z$	0
$1/K_t$	$-1/K_t$	$Z$	-1	0	$\operatorname{Exp} Z$	$-\operatorname{Exp}(-Z)$	0
$X$	$Y$	$Z$	0	1	$X$	0	$Y/X + Z$
$X$	$Y$	0	1	1	$K_t(X^2 + Y^2)^{1/2}$	0	$\operatorname{Arctg} Y/X$
$X$	$Y$	$Z$	-1	1	$K_t(X^2 + Y^2)^{1/2}$	0	$\operatorname{Arth} Y/X + Z$
$X-1$	$X-1$	0	-1	1	$2K_t(X)^{1/2}$	0	$\frac{1}{2} \lg X$

Алгоритми обчислення окремих елементарних функцій методом "цифра за цифрою" мають меншу обчислювальну складність порівняно з уніфікованим алгоритмом, тому часто їх використання є доцільнішим.

### 6.5.3. Табличний метод обчислення елементарних функцій

Ідея цього методу доволі проста: формується таблиця з наперед обчисленими значеннями функції для всіх значень аргументу. Тоді, коли виникає необхідність обчислення якоїсь функції, відбувається звертання до таблиці і зчитування з неї результату. Таблиця може мати, наприклад, наступний вигляд (табл. 6.11):

Таблиця 6.11

Аргумент	Значення
0000 0001	1001 0111
0000 0010	0011 0100
...	...
1111 1111	0011 1011

Часто застосовують комбінований підхід, коли деякі функції обчислюються таблично, а інші – за допомогою розкладу в ряд чи ітераційних формул уже з використанням функцій, що обчислюються таблично.

### 6.5.4. Таблично-алгоритмічний метод обчислення елементарних функцій

Застосування табличного методу вимагає великої за розміром таблиці при обробці аргументів з великою розрядністю, що ускладнює зберігання значень функції та їх пошук в таблиці. Тому бажано стиснути інформацію, а потім відновити її з мінімальними втратами точності. Вирішення цієї задачі можливе шляхом застосування таблично-алгоритмічного методу обчислення елементарних функцій. Для таблично-алгоритмічного методу характерна наявність арифметичних операцій, необхідних для обчислення поправки, яка додається до знайденого по таблиці значення, що залежить від старшої частини аргументу. Чим вищі вимоги до точності, тим більше арифметичних операцій необхідно виконати. При обчисленні поправки використовуються як загальний, так і частковий підходи. Загальний підхід застосовується для широкого класу функцій, а принципи використання часткового підходу ґрунтуються на застосуванні відомих тотожностей для кожної конкретної елементарної функції. При загальному підході використовується співвідношення:

$$f(X) = f(X_s + X_{n-s}) = f(X_s) + \Phi(X_{n-s}, X_s),$$

де  $X_s$  – число, утворене  $s$  старшими розрядами аргументу  $X$ ,  $X_{n-s}$  – число, утворене  $(n-s)$  молодшими розрядами аргументу  $X$ ,  $\Phi(X_{n-s}, X_s)$  – поправка.

Як видно, при загальному підході поправка залежить як від молодшої, так і від старшої частини аргументу. При частковому підході поправка в основному залежить тільки від молодшої частини аргументу. Обидва підходи передбачають використання таблично-апроксимаційного і таблично-ітераційного методів обчислень.

Це, зокрема, метод, що ґрунтується на представленні функції у виді суперпозиції двох підфункцій, де поправка задається у вигляді усередненого значення шуканої функції на кінцях підінтервалів, на які розбивається область визначення функції. Однак застосуван-

ня цього методу доцільне лише в комп'ютерах, де вимоги до точності невисокі. Можливості варіювання обчислювальною складністю та об'ємом таблиць надають методи кусочно-лінійної і кусочно-поліноміальної апроксимації, а також методи, що ґрунтуються на розбивці аргументу  $X$  на складові  $X_0$  і  $X_1$ , утворені відповідно його старшими та молодшими розрядами, і розкладанні функції в ряд Тейлора та використанні схеми Горнера.

## 6.6. Операції перетворення даних

### 6.6.1. Перетворення даних із формату з фіксованою у формат з рухомою комою та навпаки

Для перетворення даних із формату з фіксованою у формат з рухомою комою та навпаки спочатку мають бути визначені параметри форматів представлення даних, наприклад, це перетворення даних із формату з одинарною точністю за стандартом IEEE-754 до 32-розрядного формату з фіксованою комою.

При перетворенні даних із формату з фіксованою у формат з рухомою комою потрібно привести мантису до прийнятого в форматі з рухомою комою діапазону (за стандартом IEEE-754 зробити її нормалізованою), та забезпечити обчислення порядку з тим, щоб не змінилося значення числа. Потрібно відзначити, що будь-яке число в форматі з фіксованою комою може бути представлено в форматі з рухомою комою. При цьому, якщо розрядність цього числа менша або рівна розрядності мантиси, воно буде представлено точно, якщо ж більша – наблизено, оскільки частина молодших розрядів буде втрачена.

При перетворенні даних із формату з рухомою у формат з фіксованою комою потрібно привести мантису до прийнятого в форматі з фіксованою комою діапазону при зведенні порядку до нульового значення з тим, щоб не змінилося значення числа.

Розглянемо кілька прикладів.

Нехай потрібно перетворити ціле число  $X = 79_{10} = 1001111_2$ , яке представлене в двійковій формі в доповняльному 16-розрядному коді, як показано на рис. 6.24, у формат з рухомою комою розрядністю  $n=16 = m+k$ , де  $k=6$  – розрядність порядку, а  $m=10$  – розрядність мантиси.

15	14	...	10												
0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1

Рис. 6.24. Ціле число  $X = 79_{10} = 1001111_2$ , яке представлене в двійковій формі в доповняльному 16-розрядному коді

Значення порядку числа визначається з виразу  $E_x = P_x + D$ , ( $D = 2^{k-1} = 32$ ). Тут  $P_x = n-1$  – початкове значення порядку, рівне розрядності числа з фіксованою комою без врахування знакового розряду,  $D$  – зміщення порядку. Тобто початкове значення експоненти рівне  $E = 32+15 = 101111_2$ . Послідовність перетворення даних із формату з фіксованою у формат з рухомою комою показана в табл. 6.12.

Таблиця 6.12

Номер такту	Мантиса	Експонента
0	0,00000001001111	101111
1	0,000000010011110	101110

2	0,00000100111100	101101
3	0,000001001111000	101100
4	0,000010011110000	101011
5	0,000100111100000	101010
6	0,001001111000000	101001
7	0,010011110000000	101000
8	0,100111100000000	100111

Таким чином, елементи числа з рухомою комою визначені. Його значення в заданих межах представлено на рис. 6.25.

15	14	...	10
0	1 0 0 1 1 1	1 0 0 1 1 1 1 0 0	
,			

Рис. 6.25. Перетворене число

Для перевірки правильності результату перетворимо показане на рис. 6.25 число в десяткове:

$$X = +0,1001111 \cdot 2^{39-32} = +1001111,0_{(2)} = +79_{(10)}.$$

Нехай потрібно перетворити число  $X = -32768_{(10)} = -1000000000000000_2$  в формат з рухомою комою в тій самій розрядній сітці  $n = 16 = m+k$ , де  $k=6$  – розрядність порядку, а  $m=10$  – розрядність мантиси.

Двійковий еквівалент перетворюваного числа в 16-розрядній сітці з фіксованою комою в доповняльному коді має вигляд, показаний на рис. 6.26.

15	14	...	10
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		

Рис. 6.26. Перетворюване число

Значення порядку числа визначається з виразу  $E_x = P_x + D$ , ( $D = 2^{k-1} = 32$ ). Тобто початкове значення експоненти рівне  $E = 32+15 = 101111_2$ . Як видно з рис. 6.26, тут достатньо зсунути перетворюване число на один розряд праворуч, і, тим самим, отримати нормалізовану мантису, залишивши на місці знаковий розряд, та додати до початкового значення порядку одиницю. Тобто при від'ємному значенні мантиса зсувається праворуч, а до порядку добавляється одиниця. Таким чином, елементи числа з рухомою комою визначені. Його значення в заданих межах представлено на рис. 6.27.

15	14	...	10
0	1 1 0 0 0 0	1 0 0 0 0 0 0 0 0	
,			

Рис. 6.27. Перетворене число

Для перевірки вірності результату перетворимо показане на рис. 6.27 число в десяткове:

$$X = -0,1 \cdot 2^{48-32} = -1000000000 000000,0_{(2)} = -32768_{(10)}.$$

## 6.6.2. Перетворення даних з двійково-десяткового коду в двійковий та навпаки

Як ми вже бачили в попередньому розділі, двійково-десяткова система числення – це система, у якій кожну десяткову цифру від 0 до 9 подають 4-роздрядним (або більшої розрядності) двійковим еквівалентом. Для виконання перетворення можуть бути використані відповідні таблиці. Розглянемо приклади.

Нехай потрібно перетворити десяткове число  $3691_{10}$  в двійково-десятковий код.

При перетворенні кожну цифру десяткового числа перетворюють на його двійковий 4-роздрядний еквівалент.

Десяткове число	3	6	9	1
Двійково-десяткове число	0011	0110	1001	0001

Отже,  $3691_{10} = 0011\ 0110\ 1001\ 0001_{2-10}$ .

Нехай потрібно перетворити двійково-десяткове число  $1000000001110010_{2-10}$  на десятковий еквівалент.

Кожна тетрада двійково-десяткового числа перетворюється на десятковий еквівалент:

Двійково-десяткове число	1000	0000	0111	0010
Десяткове число	8	0	7	2

Отже,  $1000\ 0000\ 0111\ 0010_{2-10} = 8072_{10}$ .

Аналогічним чином здійснюється перетворення й інших кодів.

## 6.7. Операції реорганізації масивів і визначення їх параметрів

Масив – це впорядкований скінчений набір даних одного типу, які зберігаються в послідовно розташованих комірках оперативної пам'яті і мають спільну назву. Масив складається з елементів. Кожний елемент має ім'я та індекси, за якими його можна знайти в масиві. Кількість індексів елемента визначає розмір масиву. У математиці поняття масив відповідають поняття вектора та матриці.

Характеристикою масиву є його розмір – загальна кількість елементів у масиві. Інша характеристика масиву – його розмірність, оскільки масиви можуть бути як одновимірні, так і багатовимірні, а також розміри в кожному з вимірів.

Над масивами можуть виконуватись наступні операції: сортування, пошук максимуму або мінімуму, вибір заданого масиву, зсув елементів масиву, стиск масиву, визначення параметрів масиву.

Алгоритм сортування – це алгоритм для впорядкування елементів масиву. До алгоритмів сортування належать такі: сортування за методом бульбашки, сортування методом вставок, сортування злиттям, цифрове сортування, порозрядне сортування, двійкове дерево сортування, блокове сортування, сортування методом вибору, сортування методом Шелла та інші.

Розглянемо основні принципи виконання декількох вищезазначених алгоритмів сортування. Завдання сортування полягає в здійсненні перестановки елементів масиву таким чином, щоб впорядкувати їх по зростанню чи спаданню їх значень.

При виконанні сортування за методом бульбашки максимальні елементи ніби бульбашки спливають до початку масиву.

При сортуванні вставкою впорядкування елементів масиву здійснюється шляхом порівняння елемента масиву з усіма іншими та його розміщення відповідно до його значення.

При сортуванні за методом вибору впорядкований масив буде виконуватися шляхом багаторазового застосування вибору мінімального елемента з масиву, його вилученням з масиву і додаванням в кінці нового масиву, який спочатку має бути порожнім.

Сортування за методом бульбашки, вставкою та за методом вибору виконується за пропорційну квадрату розміру масиву кількість порівнянь.

Завдання пошуку максимуму або мінімуму вирішується шляхом розбиття масиву на підмасиви, аж до масиву із двох елементів, та рекурсивного вибору з них більшого або меншого. На рис. 6.28 приведено приклад послідовності дій при знаходженні максимуму або мінімуму для масиву із 8 чисел.

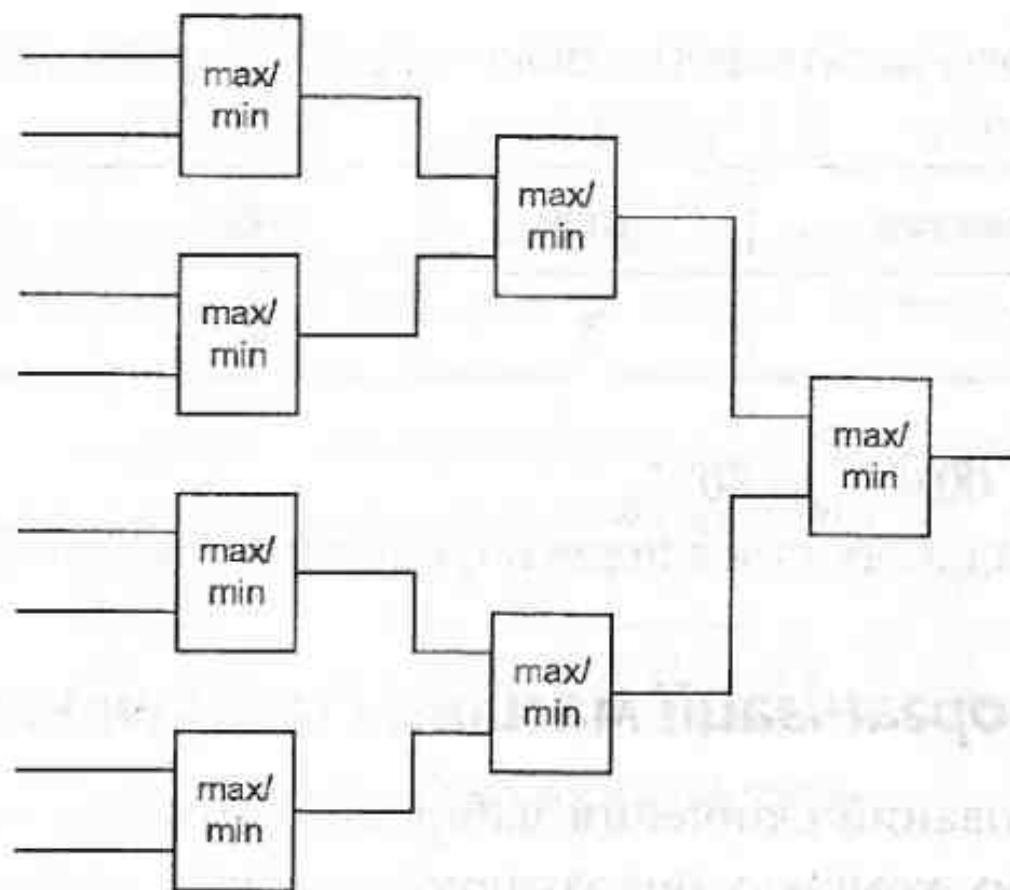


Рис. 6.28. Знаходження максимального і мінімального елементів масиву

Задачі зсуву елементів масиву по заданій координаті, транспонування масиву та визначення параметрів масиву, а також операція стиску (розширення) масиву по заданій координаті, вирішуються шляхом роботи з адресами його елементів у пам'яті.

## 6.8. Операції обробки символів та рядків символів

Є два типи операцій над символами – аналіз, тобто визначення значення символу, і перетворення, тобто зміна значення кодів символів. До основних операцій над символами належать наступні:

- Ідентифікація. Ця операція дозволяє визначити відповідність (невідповідність) значення коду аналізованого символу Х коду заданого символу С. Ідентифікація символу Х виконується шляхом перевірки збігу коду символу Х з кодом символу С, розміщених в основній пам'яті або реєстрах, і вказаних відповідними адресами. Результатом операції

є 1, якщо коди символів співпадають, і 0, якщо не співпадають. Операція ідентифікації виконується як операція порівняння двійкових кодів на збіжність, описана в п. 6.4.1.

- Перевірка за маскою. Операція виконується з метою визначення відповідності коду досліджуваного символу Х певній озnaці, що вказується кодом символу маски M. Вона передбачає перевірку збігу кодових розрядів символу X з кодовими розрядами символу маски M. Це дозволяє класифікувати код символу на приналежність до певної групи символів (наприклад, символи цифр, букв, спеціальних знаків і т. п.).

- Порівняння символів. Порівняння символу X із заданим символом С дозволяє визначити відношення ваг кодів цих символів. Коди символів при цьому зазвичай розглядаються як двійкові цілі числа без знаків (можливі й інші інтерпретації), і порівняння символів виконується як порівняння числових значень їх кодів, в результаті якого визначаються співвідношення  $X = C$ ,  $X > C$  і  $X < C$ . Порівняння символів проводиться як операція визначення старшинства двійкових кодів, описана в п. 6.4.2.

- Запис символу за маскою. Ця операція дозволяє змінювати код символу X згідно з кодом символу маски M шляхом видалення не вказаних (або, навпаки, вказаних) в коді символу маски двійкових розрядів коду символу X. Тобто запис символу за маскою забезпечується вибірковим записом однічних значень тільки тих кодових розрядів символу X, яким відповідають одиничні значення розрядів коду символу маски M.

- Запис зони байта. Ця операція є окремим випадком операції запису символу за маскою, вживаній при обробці кодів символів байтової структури. Операція дозволяє переслати код лівої (зонової) або правої (цифрової) частини символу X у відповідну частину іншого символу Y, тобто чотирьох правих і лівих двійкових розрядів коду X в аналогічні розряди коду Y. Інша частина кодів Y і X не змінюється.

Рядок представляє послідовність символів, що зберігаються в пам'яті комп'ютера та створюють певну семантичну одиницю оброблюваної інформації. Рядки символів можуть мати фіксовану, змінну або довільну довжину. Одиноцею обробки інформації рядків символів є не двійковий розряд, а символ. Тому і всі операції над рядками символів розглядаються як операції над впорядкованими послідовностями символів.

Аналогічно до операцій над символами, операції над рядками символів можна розділити на два види: аналізу, які не змінюють вмісту оброблюваних рядків, і перетворення, за допомогою яких змінюється семантичний або синтаксичний вміст рядків.

До операцій аналізу належать операції пошуку символу та перевірка за маскою. Операція пошуку символу дозволяє визначити наявність і місцезнаходження певного (заданого) символу в рядку. Операція завершується при виявленні первого шуканого символу в рядку, або закінченням посимвольної перевірки цього рядка, якщо шуканий символ відсутній в цьому рядку. Положення виявленого символу визначається порядковим номером символу в рядку або його адресою (при посимвольній системі адресації). Пошук виконується шляхом посимвольного порівняння кодів символів рядка з кодом заданого символу на виконання заданого відношення ( $=, >, <$ ). Зазвичай передбачаються окремі операції пошуку для кожного з можливих відношень. Коди символів інтерпретуються як двійкові числа без знаків. Залежно від прийнятої системи кодування можливі й інші закони визначення ваг двійкових розрядів кодів символів. Результат виконання операції відображається станом спеціального індикатора і вмістом регистра адреси пам'яті, відповідним адресі останнього звернення до чергового досліджуваного символу.

лу. Кінець досліджуваного рядка визначається після закінчення обробки певного числа символів (при фіксованій довжині слова), після обробки вказаної в команді кількості символів (при змінній посимвольній адресації), або при виявленні ознаки кінця рядка (для слів довільної довжини). Очевидно, що ситуація відсутності шуканого символу в рядку утворює один із станів коду умов або індикаторного реєстра. Перевірка за маскою аналогічна операції пошуку символу, тільки виконується пошук не одного символу, а їх послідовності.

До операцій перетворення рядків символів належать операції компонування, редагування, перекодування та перетворення форматів байтів.

За допомогою компонування (запису за маскою) можна виконати об'єднання вмісту двох рядків  $X$  і  $Y$  за кодом символу маски  $M$  шляхом запису символів одного рядка ( $X$ ) в місце розташування символів іншого рядка ( $Y$ ) в тих позиціях, які вказані кодом маски. Інші символи первого рядка, другого рядка і код маски не змінюються. Компонування використовується для формування нового рядка з послідовностей символів двох початкових рядків зміною значення деяких символів рядка або зміною послідовності символів в рядку. В операції беруть участь три операнди: змінний рядок  $Y$ , рядок даних  $X$  і символ (може бути рядок) маски  $M$ , послідовність двійкових розрядів якої відповідає символам перших двох операндів зліва направо (або навпаки). Перші два операнди зазвичай задаються їх адресами, а код маски вказується безпосередньо в команді. За цих умов зміні підлягають символи  $Y$ , яким відповідає однічне значення двійкового розряду коду маски  $M$ .

Редагування дозволяє видозмінити заданий рядок  $F$  (звичайно цей текст представляє число, що зберігається для економії пам'яті в найбільш компактному вигляді) згідно з певним шаблоном  $S$ . Необхідність в такій зміні часто виникає при виведенні наборів різних змінних даних на друк згідно з формою їх відображення, що вимагає, наприклад, проставлення спеціальних знаків, записів, пояснень і т. п. Необхідні редакторські дії задаються відповідними керуючими символами, що включаються в необхідній послідовності в рядок-шаблон, відповідно до якого і виконується операція редагування. Достатню гнучкість редагування забезпечують наступні операції над керуючими символами в рядку-шаблоні:

- заміна символів-заповнювачів в тексті шаблону символами рядка даних у порядку їх проходження (зазвичай зліва направо);
- проставлення десяткової крапки (або коми) в заповнюваному числовим рядком контексті шаблону;
- видалення незначущих нулів при заповненні контексту шаблону числовим рядком (заміна їх пропусками);
- вказівка місця початку значущості числових даних в контексті шаблону (для виключення заданого раніше видалення незначущих нулів, починаючи з цього місця);
- задання в рядку шаблону будь-яких послідовностей постійних текстових, числових даних або спеціальних символів;
- вказівка кінця шаблону (якщо формат рядка шаблону не має постійної або змінної довжини).

Редагування цифрового рядка  $F$  здійснюється посимвольно зліва направо згідно з вмістом послідовності керуючих символів рядка шаблону  $S$ . В результаті виконання опе-

рації виходить послідовність символів рядка шаблону  $S$ , в якому символи-заповнювачі (а також символи керування видаленням незначущих нулів і початку значущості, якщо вони є) замінені конкретними значеннями цифр редагованого числового рядка. Саме редаговане число при цьому не змінюється. Шаблон і редаговане число задаються їх адресами, а операція закінчується при виявленні символу-покажчика кінця шаблону при послідовному його огляді під час виконання операції. Зазвичай шаблон створюється для редагування послідовності одного числа, проте це не є обов'язковим, оскільки виконання операції керується повністю контекстом шаблону і єдиною додатковою вимогою для редагування декількох чисел за одним шаблоном є послідовне розташування цих чисел в пам'яті в порядку, відповідному послідовності їх включення в контекст відредагованого рядка згідно з заданим форматом шаблону. Природно, при цьому необхідне багаторазове використання символів керування видаленням незначущих нулів і вказівки місця початку значущості. Приймається, що дія попереднього символу, задаючого видалення нулів, анулюється першим подальшим символом вказівки місця початку значущості і будь-яким символом тексту шаблону, окрім символів десяткової крапки (або коми) та символів-заповнювачів. Знак редагованого числа розглядається як чергова цифра цього числа.

Перекодування призначено для виконання переходу від однієї системи кодування символів до іншої, що довільно задається. Передбачається, що належний перекодуванню рядок зберігається в послідовних комірках пам'яті, починаючи з деякої адреси, і задана таблиця перекодування, що зберігається в пам'яті, також починається з деякої адреси. Таблиця перекодування містить перелік належних перекодуванню символів в новій системі кодування. Перекодування виконується заміною чергового символу одного рядка відповідним новим кодом цього символу з таблиці перекодування. Перекодування може супроводжуватися визначенням і відповідною індикацією деяких умов (наприклад, наявність тільки цифрових і алфавітних, або ж тільки визначених символів в рядку, що перекодовується) і завершенням операції при виконанні або невиконанні певних умов. Таким чином, можуть формуватися декілька варіантів операцій перекодування.

Перетворення форматів байтів застосовується для економного зберігання цифрових даних в пам'яті та зручності представлення їх вмісту на зовнішніх документах, що підлягають сприйняттю людиною. Як відомо, для зберігання цифрових даних достатньо 4 двійкових розряді на десяткову цифру, тобто в байті, призначенному для зберігання коду одного алфавітно-цифрового символу, можна розмістити коди двох десяткових цифр, або коди цифри і знаку, що удвічі скорочує об'єм пам'яті, необхідної для зберігання цих даних. Проте при виводі на друк або інші зовнішні носії інформації необхідно привести коди всіх символів до єдиного формату, оскільки інакше неможлива правильна їх інтерпретація з обліком тільки кодів байтів.

Таким чином, при байтовій організації пам'яті для зберігання одиниць чисової і символьної інформації застосовують два формати, які умовно називаються упакований – по дві десяткові цифри в байті і зоновий – одна десяткова цифра (або знак) в байті. В останньому випадку двійковий код цифри (або знаку) прийнято розміщувати в правих чотирьох розрядах коду байта і називати цю його частину цифровою, а ліва частина байта при цьому містить спеціальний код зони, використовуваний як ознака цифрових даних, і називається зоновою частиною.

Для перетворення форматів представлення цифрових даних застосовують 4 операції: пакування – перехід від розширеного формату до стислого, розпакування – зворотна операція пакуванню; пересилку цифр – перезапис цифрових частин байтів і пересилку зон – перезапис зонових частин байтів.

## 6.9. Короткий зміст розділу

В розділі розкриті основні питання виконання операцій обробки даних: логічних (логічне множення, логічне додавання, інверсія і т. д.), зсуву (праворуч, ліворуч), відношення (менше, більше, рівне, менше-рівне, більше-рівне), арифметичних (додавання, віднімання, множення та ділення), обчислення елементарних функцій, перетворення даних (перетворення із формату з фіксованою в формат з рухомою комою і навпаки, перетворення з двійково-десяткового коду в двійковий і навпаки), реорганізації масивів і визначення їх параметрів (сортування, пошук максимуму або мінімуму, вибір заданого масиву, зсув елементів масиву, стиск масиву), обробки символів та стрічок символів (пошук символу, зсув, заміна символів в стрічці, пакування стрічок символів, порівняння стрічок символів).

Розглянуті основні алгоритми виконання вищезазначених операцій.

## 6.10. Література для подальшого читання

Алгоритми виконання логічних (логічне множення, логічне додавання, інверсія і т. д.), зсуву (праворуч, ліворуч), відношення (менше, більше, рівне, менше-рівне, більше-рівне) та арифметичних (додавання, віднімання, множення та ділення) наведені в багатьох працях, присвячених питанням побудови комп’ютерів. В першу чергу серед них потрібно відзначити роботи [1–5]. В роботах [6–11] описані алгоритми обчислення елементарних функцій за методом “цифра за цифрою”. Питанням побудови табличних та таблично-алгоритмічних методів обчислення елементарних функцій присвячені роботи [12–25]. Алгоритми сортування детально описані в роботах [26, 27]. Клас алгоритмів паралельного сортування описаний в роботі [28]. Виконання операцій обробки символів та стрічок символів наведено в [3].

## 6.11. Література до розділу 6

1. Прикладная теория цифровых автоматов / К. Г. Самофалов, А. М. Романкевич, В. Н. Валуйский, Ю. С. Каневский, М. М. Пиневич. – К.: Вища школа, 1987. – 375 с.
2. Корнейчук В. И., Тарасенко В. П. Основы компьютерной арифметики. – К. Корнейчук, 2002. – 176 с.
3. Рабинович З. Л., Раманаускас В. А. Типовые операции в вычислительных машинах. – К.: Техника, 1980. – 264 с.
4. Карцев М. А. Арифметика цифровых машин. – М.: Наука, 1969.
5. Савельев А. Я. Арифметические и логические основы цифровых автоматов. – М.: Наука, 1980. – 255 с.
6. Бойков В. Д., Смолов В. Б. Аппаратурная реализация элементных функций в ЦВМ. – Л. ЛГУ. – 96 с.
7. Благовещенский Ю. В., Теслер Г. С. Вычисление элементарных функций на ЭВМ. – К.: Техника, 1977. – 208 с.

8. Оранский А. М. Аппаратные методы в цифровой вычислительной технике. – Минск, Из во БГУ, 1977. – 208 с
9. Volder J. E. The CORDIC trigonometric computing technique. – "IRE Trans.", 1959, 3, pp. 330–334.
10. Walther I. S. -In: Proc. Spring Joint Comput. Conf Monthvale, 1971, V.38, N.J. : AFIPS Press, 1971.
11. Meggitt I. E. Pseudodivision and Pseudomultiplication process. – IBMJ. Res. Develop., v. 6., 1962, № 2
12. Смолов В. Б., Байков В. Д. Анализ табличных и таблично алгоритмических методов воспроизведения элементарных функций. – Электронное моделирование, 1980, № 1, с. 22–27.
13. Колубай С. К., Мурашко А. Г. Принципы построения процессоров типа “память система поиска” // УСИМ, 1977, № 4, с. 58–62.
14. Хемел А. Выполнение математических операций с помощью ПЗУ // Экспрессинформация, серия “Вычислительная техника”, 1970, № 32, с. 27–29.
15. Смолов В. Б., Байков В. Д. Анализ табличных и таблично алгоритмических методов воспроизведения элементарных функций // Электронное моделирование, 1980, № 1, с. 22–27.
16. Балашов Е. П., Смолов В. Б. и др. К вопросу применения сокращенных таблиц функций для построения высокопроизводительных однородных процессоров // УСИМ, 1975, № 3, с. 99–102
17. Ильин В. А., Попов Ю. А., Дружинина И. И. Об использовании сокращенных таблиц при вычислении элементарных функций // УСИМ, 1979, № 1, с. 58–60.
18. Палагин А. В., Кургаев А. Ф., Кондрачук И. М. К выбору метода вычисления элементарных функций в мини ЭВМ // УСИМ, 1973, № 5, с. 65–69.
19. Потапов В. И., Нестерук В. Ф., Флоренсов А. Н. Быстродействующие арифметико логические устройства ЦВМ. – Новосибирск, 1978.
20. Потапов В. И., Флоренсов А. Н. Таблично аддитивная организация вычисления в ЭВМ функций, принадлежащих к классу дважды непрерывно дифференцируемых // Автоматика и вычислительная техника, 1977, № 6, с. 78–84.
21. Потапов В. И., Флоренсов А. Н. Таблично алгоритмическая организация вычислений элементарных функций в ЦВМ. – Изв. вузов, сер. Приборостроение, 1978, т. 21, № 9, с. 63–66.
22. Потапов В. И., Флоренсов А. Н. Таблично алгоритмический метод реализации в ЦВМ функции логарифма // УСИМ, 1978, № 4, с. 90–94.
23. Мухопад Ю. Ф., Федченко А. И., Лукашенко В. М. Таблично функциональные преобразователи с ограниченным числом хранимых констант // УСИМ, 1975, № 4, с. 99–102.
24. Голубков Ю. А., Лебедев А. В. Некоторые пути повышения скорости вычисления элементарных функций на ЦВМ. – М., 1962. – 64 с.
25. Пелед Ф., Лиу Б. (США). Цифровая обработка сигналов: Теория, проектирование, реализация: Пер. с англ. – Киев: Вища школа. Головное изд. во, 1979. – 264 с
26. Кнут. Сортировка и поиск.
27. Кун С. Матричные процессоры на СБИС: Пер. с англ. – М.: Мир, 1991. – 672 с
28. Мельник А. А., Илькив В. С. Реализация алгоритмов сортировки. В кн. "Систолические вычислительные структуры". Препринт АН УССР. Инт ИППММ, № 3, 1987

## 6.12. Питання до розділу 6

1. Назвіть основні операції обробки даних
2. Які основні логічні операції виконуються в комп’ютері? Наведіть таблицю істинності цих операцій
3. Дайте пояснення операцій логічного зсуву
4. Дайте пояснення операцій арифметичного зсуву
5. Дайте пояснення операцій циклічного зсуву
6. Наведіть правило та приклад додавання двійкових чисел без знаків

7. Наведіть правило та приклад віднімання двійкових чисел без знаків
8. Наведіть правило та приклад додавання двійкових чисел, представлених в прямому коді
9. Наведіть правило та приклад додавання двійкових чисел, представлених в оберненому коді
10. Наведіть правило та приклад додавання двійкових чисел, представлених в доповняльном коді
11. Як фіксується переповнення при додаванні двійкових чисел?
12. Приведіть граф алгоритму множення цілих двійкових чисел без знаків
13. Наведіть алгоритми багатомісної операції додавання часткових добутків з використанням операторів паралельного двомісного додавання
14. Наведіть граф алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу молодших розрядів множника
15. Наведіть граф алгоритму послідовного попарного додавання часткових добутків, отриманих починаючи з аналізу старших розрядів множника
16. Наведіть граф алгоритму паралельного попарного додавання часткових добутків з використанням структури бінарного дерева
17. Наведіть алгоритми багатомісної операції додавання часткових добутків з використанням операторів двомісного однорозрядного додавання
18. Приведіть граф алгоритму множення двійкових чисел із знаками
19. Поясніть суть та переваги алгоритму множення двійкових чисел за алгоритмом Бута
20. Приведіть та поясніть алгоритми ділення з відновленням та без відновлення залишку. Яка між ними різниця?
21. Як привести число з рухомою комою до нормалізованого вигляду?
22. Приведіть та поясніть алгоритми додавання та віднімання чисел з рухомою комою
23. Приведіть та поясніть алгоритми множення чисел з рухомою комою
24. Приведіть та поясніть алгоритми ділення чисел з рухомою комою
25. Поясніть як виконується операція порівняння двійкових кодів на збіжність та визначення їх старшинства
26. Які є методи обчислення елементарних функцій?
27. Дайте пояснення методу обчислення елементарних функцій шляхом розкладу в ряд
28. Дайте пояснення методу обчислення елементарних функцій шляхом використання ітераційних обчислень
29. Поясніть алгоритм обчислення елементарних функцій методом „цифра за цифрою”
30. Поясніть суть табличного методу обчислення елементарних функцій
31. Поясніть суть таблично алгоритмічного методу обчислення елементарних функцій
32. Опишіть алгоритм перетворення з формату з фікованою комою до формату з рухомою комою
33. Опишіть алгоритм перетворення з формату з рухомою комою до формату з фікованою комою
34. Як перетворити двійково десятковий код в двійковий і навпаки?
35. Що таке масив?
36. Назвіть характеристики масиву
37. Назвіть алгоритми сортування чисел
38. В чому полягає задача сортування чисел?
39. Опишіть суть алгоритму сортування методом бульки
40. Опишіть суть алгоритму сортування методом вставки
41. Опишіть суть алгоритму сортування методом вибору елементів масиву
42. Приведіть алгоритм знаходження максимального і мінімального елементів масиву
43. Приведіть перелік основних операцій над символами
44. Приведіть перелік основних операцій над рядками символів