НОВІ ЗАСОБИ КІБЕРНЕТИКИ, ІНФОРМАТИКИ, ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА СИСТЕМНОГО АНАЛІЗУ

УДК 519.6

В.К. ЗАДІРАКА
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *zvk140@ukr.net*.

А.М. ТЕРЕЩЕНКО
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *teramidi@ukr.net.*

ПАРАЛЕЛЬНІ МЕТОДИ ПРЕДСТАВЛЕННЯ ЧИСЕЛ ДЛЯ ТЕСТУВАННЯ ОПЕРАЦІЙ БАГАТОРОЗРЯДНОЇ АРИФМЕТИКИ

Анотація. Запропоновано методи реалізації операції представлення багаторозрядного числа у системі числення з іншою основою, потрібні для тестування арифметичних операцій у разі використання паралельних процесорів. Розглянуто представлення числа у системах числення на основі багаторозрядних операцій ділення та віднімання або багаторозрядних операцій множення та додавання. Алгоритм з розбиттям числа на групи цифр дає змогу враховувати довжину машинного слова та розподіляти обчислення між процесорами. Проаналізовано складність за кількістю операцій, обсяг додаткової пам'яті для алгоритмів на основі ітераційного та рекурсивного методів.

Ключові слова: система числення, багаторозрядна арифметика, багаторозрядне додавання, багаторозрядне множення, паралельна модель обчислень.

ВСТУП

Система числення визначає архітектуру пристрою, який реалізує операції для цієї системи. Представлення чисел залежить від конкретної фізичної реалізації елементів пристрою. Використання нових паралельних обчислювальних систем, таких як багатоядерні процесори, графічні прискорювачі, кластери, системи з розподіленою пам'яттю, розподілені системи та інші, зумовлено потребою розв'язання прикладних задач у різних галузях. Серед таких задач можна виокремити задачі обчислення оболонок ядерних реакторів, моделювання фізичних, хімічних процесів, аеродинаміки, гідродинаміки, захисту інформації, високоточних обчислень тощо. Алгоритми багаторозрядної арифметики залежать від системи числення, у якій їх використовують. Швидкодія асиметричних криптографічних програмно-апаратних комплексів [1–5] залежить від багаторозрядної операції множення, яка є складовою операції піднесення до степеня за модулем. Системи числення у залишкових класах [6, 7] мають швидкі реалізації багаторозрядної операції множення та піднесення до степеня за модулем. А для порівняння чисел у залишкових класах використовують метод, у якому багаторозрядні числа представляють у позиційній системі числення зі змішаною основою для порівняння. У логарифмічній системі числення операції додавання та віднімання відповідають операції множення та ділення у зви-

чайній арифметиці відповідно. В інформаційних технологіях застосовують двійкову, десяткову, вісімкову та шістнадцяткову системи. Десяткова система числення, яка має базове значення 10, є найпоширенішою системою числення. Найпростішою, а отже найдешевшою і найнадійнішою є конструкція комп'ютера, в якому для представлення чисел використовується двійкова система.

Найбільшу щільність запису інформації має система числення з основою, рівною основі натуральних логарифмів e (число Ейлера). Доведено, що найбільш швидкодійними та економічними для представлення чисел були би комп'ютери, в яких використовувалася би система числення з основою $e = 2.718281828...$ (основа натуральних логарифмів). (Проте технічно вони були би дуже складними.) Близькою до неї є трійкова система. Трійкова логіка елегантніша та ефективніша ніж двійкова [8]. Елементи трійкової системи числення використовувалися в системі числення стародавніх шумерів, в системах мір, ваг і грошей, в яких одиниці були рівні трьом. Італійський математик Фібоначчі об'єднав ці властивості в позиційній цілочисловій симетричній трійковій системі числення для розв'язання «задачі про гирі». Симетрична система дає змогу зображати від'ємні числа, не використовуючи окремого знака мінуса. Число 2 зображується виразом $1\,\overline{1}$, де в розряді трійок є цифра 1 і в розряді одиниць є цифра $\overline{1}$ (мінус одиниця).

Розмаїття наявних систем числення та методів реалізації операцій багаторозрядної арифметики для таких систем числення зумовлено різними вимогами до швидкодії виконання та точності результату, що впливає на розвиток великої кількості пристроїв, для яких реалізуються алгоритми у різних моделях обчислень. Як наслідок, це потребує включення бібліотеки програм, яка виконує операції над багаторозрядними числами, у штатне математичне забезпечення сучасних одно- та багатопроцесорних систем із заданими характеристиками якості за точністю та швидкодією. У більшості випадків вартість розроблення програмного забезпечення (ПЗ) не є меншою, ніж вартість обладнання, на якому воно виконується.

Дуже часто немає можливості розв'язувати задачі з розроблення та реалізації нового алгоритму, тому що на цьому етапі ще немає тестових даних, за допомогою яких можна проаналізувати результат розв'язання задачі. Від якості підготовлених даних залежить якість реалізованого алгоритму та час, необхідний для пошуку та усунення помилок алгоритму-програми і його реалізації. Тому дуже важливо мати вже підготовлений великий набір тестових даних для різної довжини вхідних даних та для тестування граничних випадків. Але якщо розглядається розроблення та реалізація багаторозрядного алгоритму для широкого діапазону даних, наприклад, від 1024 до 4096 бітів з кроком 128 бітів, то на підготовку тестових вхідних та вихідних даних може витрачатися найбільше часу. Так, наприклад, підготовка тестових даних для паралельної моделі обчислень займає більше часу, ніж розроблення алгоритму у послідовній моделі обчислень, оскільки для отримання тестових даних необхідно реалізувати алгоритми у послідовній моделі обчислень з використанням однакових технологій. Зрозуміло, що зменшення часу на підготовку тестових даних дає можливість приділити більше часу на якісну реалізацію алгоритму. Тому цікавішими є такі багаторозрядні дані, що легко генеруються у системі числення, у якій відбувається порівняння тестових результатів, і не потребують використання додаткового програмного та апаратного забезпечення. У випадку виконання чималої кількості операцій представлення тестових даних в іншій системі числення виникає потреба використання швидких методів такого представлення. Для тестування у паралельній моделі обчислень всі операції включно з операцією представлення числа в іншій системі числення мають базуватися на паралельних методах для підвищення ефективності задіяних процесорів.

У роботі розглянуто методи представлення чисел у системах числення для тестування операцій у послідовній та паралельній моделях обчислень. Розглянуто методи представлення чисел у системах числення на основі багаторозрядних операцій ділення та віднімання або багаторозрядних операцій множення та додавання. Запропоновано методи, які дають змогу розпаралелити алгоритм представлення числа у системі числення з іншою основою. Проаналізовано складність алгоритмів за кількістю однорозрядних операцій на основі ітераційного та рекурсивного методів, які дають змогу враховувати довжину машинного слова у бітах за рахунок розбиття числа на групи цифр, кожну з яких опрацьовує окремий процесор.

ПРЕДСТАВЛЕННЯ ЧИСЕЛ У ПОЗИЦІЙНИХ СИСТЕМАХ ЧИСЛЕННЯ

Будь-яке число може бути представлено у системі числення з основою D у вигляді лінійної комбінації степенів числа D [8]

$$V = \sum_{k=0}^{N-1} (d_k \cdot D^k), \tag{1}$$

де d_k — цифра у позиції k для позиційної системи числення з основою D; N — кількість цифр у числі у позиційній системі числення з основою D.

Систему числення з однією основою N також називають N-ковою системою числення, а числа у такій системі — N-ковими. Крім систем з однією основою, існують системи зі змішаною основою, тобто з послідовністю чисел, що зростає. Системи зі змішаною основою є узагальненням позиційних систем. Існують й інші системи числення, а саме: система числення Фібоначчі [9], числова система залишків, системи числення факторіальна і біноміальна, система числення мая, код Грея тощо. Кожна система числення потребує реалізації своєї арифметики. У роботі розглядаються алгоритми для системи з однією основою. У більшості випадків алгоритми можуть бути адаптовані і для інших систем числення. Далі будемо розглядати тільки позиційні системи числення, тому слово «позиційна» будемо випускати. Також можна випускати слова «числення», якщо з контексту відомо значення основи.

Зауважимо, що кожна система числення має свій набір цифр (нумералів, алфавіт символів). Так, наприклад, шістнадцяткова система числення має такий набір цифр:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F,$$

де A відповідає значенню 10 у десятковій системі числення, B — 11, C — 12, D — 13, E — 14, F — 15.

Іноді для позначення системи числення у представленні числа додається буква, наприклад, b для представлення у бінарній (двійковій) системі числення, o — у вісімковій системі числення, d — у десятковій системі числення, h — у шістнадцятковій системі числення тощо.

Наприклад, $1000h$ представляє число у шістнадцятковій системі числення, яке у десятковій системі числення відповідає числу $4096d$:

$$1000h = 1 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16^1 + 0 = 4096d.$$

Число *FFFFh* займає 16 бітів (машинне слово) та у десятковій системі числення відповідає числу $65535d$:

$$FFFFh = 15 \cdot 16^3 + 15 \cdot 16^2 + 15 \cdot 16^1 + 15 = 65535d.$$

ПАРАЛЕЛЬНА МОДЕЛЬ ОБЧИСЛЕНЬ

Як паралельну модель обчислень розглядають GPU (Graphics Processing Unit) відеокарти. Фізично це є окремий пристрій, який інсталюють додатково в комп'ютер. Він побудований за технологією SIMD (Single Instruction Multiple Data), де потокові процесори можуть виконувати одну інструкцію одночасно, оперуючи різними даними. Такий GPU має власну пам'ять, яка значно швидша пам'яті основного процесора. Обчислення на GPU розбиваються на робочі групи паралельних процесорів, де всі операції можуть виконуватись синхронно.

ПОСТАНОВКА ЗАДАЧІ

Нехай число V у системі числення з основою D представлено у вигляді (1), а у позиційній системі числення з іншою основою B — у вигляді

$$V = \sum_{k=0}^{P-1} b_k B^k,$$

де b_k — цифра у позиції k для системи числення з основою $B = 2^w$ ($w = 4, 8, 16, 32, 64$ — довжина машинного слова у бітах); P — кількість позицій у системі числення з основою B. Усі числа, які після представлення у системі числення з основою 2 перевищуватимуть за кількістю бітів довжину машинного слова у бітах, будемо називати багатослівними або багаторозрядними числами.

Потрібно побудувати швидкий алгоритм представлення числа у системі числення на основі операцій, які виконуються у паралельній моделі обчислень. Алгоритм повинен враховувати довжину машинного слова (8, 16, 32, 64 бітів і т.д.).

Нехай $V(k) \leftarrow V(k) + D(k)$ — значення розряду з індексом k вектора V збільшується на значення $D(k)$ та результат записується в елемент $V(k)$; $L(X), H(X)$ — оператори знаходження молодших (*low*) та старших (*high*) елементів вектора X. Нумерація починається з нуля, тому результатом виконання оператора $L(X)$ над вектором X довжиною $2N$ елементів є елементи з індексами 0, 1, 2, 3 і т.д., а результатом виконання оператора $H(X)$ — елементи з індексами N, $N+1$, $N+2$, $N+3$ і т.д. Обчислення $\overline{L(X)}, H(X)$ здійснюється так: $[L(X)](k) = X(k)$, $[H(X)](k) = X(N+k)$, $k = \overline{0, N-1}$.

АЛГОРИТМ ПЕРЕХОДУ В СИСТЕМУ ЧИСЛЕННЯ З ІНШОЮ ОСНОВОЮ

Представлення числа з системи числення з однією основою у системі числення з іншою основою називають переходом до системи числення з іншою основою або переходом до іншої системи числення. Алгоритм, який виконує такий перехід, називають алгоритмом переходу у систему числення з іншою основою. Розглянемо виконання цього алгоритму над числом V у послідовній моделі обчислень.

Алгоритм 1 переходу в систему числення з іншою основою для послідовної моделі обчислень.

Вхід: $V = \sum_{k=0}^{N-1} (d_k \cdot D^k)$ — число, представлене у системі числення з основою D; N — кількість цифр у системі числення з основою D; B — основа системи числення, в якій потрібно представити число.

Результат: $\sum_{k=0}^{P-1} (b_k \cdot B^k)$ — число, представлене у системі числення з основою B; P — кількість цифр у системі числення з основою B.

Крок 1. $T \leftarrow V$, $i \leftarrow 0$.

Крок 2. Поки $T > 0$, то

Крок 3. $b_i \leftarrow T \bmod B$.

Крок 4. $T \leftarrow (T - b_i) / B$.

Крок 5. $i \leftarrow i + 1$.

Крок 6. Кінець поки.

Крок 7. $P \leftarrow i$.

Якщо в процесі реалізації алгоритму є можливість одночасного обчислення частки і лишку, то кроки 3 та 4 можуть бути поєднанні так: $(T, b_i) \leftarrow T / B$.

Розглянемо роботу алгоритму 1 на прикладі числа $579d$, яке представлено у десятковій системі числення та яке потрібно представити у шістнадцятковій системі числення:

$$T \leftarrow 579,\ i \leftarrow 0;$$

$$b_0 \leftarrow T \bmod B = 579 \bmod 16 = 3;$$

$$T \leftarrow (T - b_0) / B = (579 - 3) / 16 = 36,\ i \leftarrow 1;$$

$$b_1 \leftarrow T \bmod B = 36 \bmod 16 = 4;$$

$$T \leftarrow (T - b_1) / B = (36 - 4) / 16 = 2,\ i \leftarrow 2;$$

$$b_2 \leftarrow T \bmod B = 2 \bmod 16 = 2;$$

$$T \leftarrow (T - b_2) / B = (2 - 2) / 16 = 0,\ i \leftarrow 3.$$

Виконання алгоритму 1 продовжується до того часу, поки $T > 0$.

Отримуємо, що десяткове число $579d$ у шістнадцятковій системі числення представлено числом $243h$.

У алгоритмі 1 бачимо, що основною операцією є операція ділення, при чому вона повинна бути багаторозрядною, оскільки число може бути багаторозрядним.

Під час реалізації алгоритму 1 спершу виникає питання щодо багаторозрядної арифметики. З огляду на попередній приклад зручніше було би виконувати ділення у десятковій системі числення. Але сучасні процесори не підтримують операцій у десятковій системі числення. Процесори Z80 та похідні від них підтримують двійково-десятковий формат (binary-coded decimal, BCD) представлення чисел та операцій над ними. Реалізація BCD-арифметики є дуже простою, тому вона описана у наборах інструкцій майже всіх мікропроцесорів та мікроконтролерів. У BCD-форматі кожна цифра десяткового числа записується своїм двійковим представленням з доповненням нулями до чотирьох розрядів (тетради). Таким чином, кожен байт зберігає дві десяткові цифри або числа в діапазоні від 0 до 99 в упакованому форматі. Суттєвим недоліком операцій над числами у двійково-десятковому форматі є те, що після виконання кожної арифметичної операції (додавання, віднімання, множення, ділення і т.д.) можуть виникати заборонені комбінації у тетраді, а саме числа від 10 до 15. Такі переповнення треба корегувати за рахунок виконання додаткової операції з нормалізації числа, яка переносить частину числа у старші тетради.

Ділення на 16 у двійковій арифметиці є зсувом усіх бітів на чотири біти вправо, але для того, щоб виконати таку операцію, число вже повинно бути представлено у двійковій (або шістнадцятковій) системі числення.

СКЛАДНІСТЬ АЛГОРИТМУ 1 ЗА КІЛЬКІСТЮ ОДНОРОЗРЯДНИХ ОПЕРАЦІЙ

Складність за кількістю операцій будемо обчислювати за припущення, що всі арифметичні операції (додавання, віднімання, ділення, множення) реалізовані для системи числення з основою D, в якій представлено число, та кожна

цифра (позиція) займає один розряд. Для спрощення також вважатимемо, що ділення на B завжди є діленням на однорозрядне число, навіть якщо $D < B$.

Лема 1. Загальну кількість однорозрядних операцій алгоритму 1 можна обчислити так:

$$O^1_{div}(P) = (P-1)\cdot P/2,\ O^1_{sub}(P) = P,$$

де P — кількість цифр числа, яке представлене у системі числення з основою B, $P \geq 1$; O^1_{div} та O^1_{sub} — кількість однорозрядних операцій ділення та віднімання для послідовної моделі обчислень.

Доведення. Результатом алгоритму буде число, яке займатимемо P позицій у системі числення з основою B. Тому на першій ітерації алгоритму буде виконано $P-1$ послідовних однорозрядних операцій ділення числа T на однорозрядне число B та одну операцію однорозрядного віднімання. Вважаємо, що операція ділення одночасно обчислює і частку, і залишок від ділення, а під час віднімання не виникає знака запозичення зі старшого розряду, або вважаємо, що знак запозичення автоматично враховується на наступних ітераціях. На другій ітерації число T буде у B разів меншим, тому треба буде виконати на одне однорозрядне ділення менше $(P-2)$ та завершити ітерації одним однорозрядним відніманням. На останній ітерації відбувається тільки однорозрядне віднімання. Загальна кількість однорозрядних операцій ділення може бути обчислена на основі арифметичної прогресії

$$O^1_{div}(P) = \frac{0+P-1}{2}\cdot P.$$

Кількість однорозрядних операцій віднімання дорівнює кількості ітерацій P.

Лему доведено.

ПРЕДСТАВЛЕННЯ У СИСТЕМІ ЧИСЛЕННЯ, У ЯКІЙ ВИКОНУЮТЬСЯ АРИФМЕТИЧНІ ОПЕРАЦІЇ

Розглянемо, коли арифметичні операції реалізовані тільки у системі числення, у якій необхідно представити число. У цьому разі можна не використовувати операцію ділення. Далі наведено покроковий опис алгоритму, побудованого на основі співвідношення (1).

Алгоритм 2 представлення числа у системі числення на основі арифметичних операцій цієї системи для послідовної моделі обчислень.

Вхід: $\sum_{k=0}^{N-1}(d_k \cdot D^k)$ — число, представлене у системі числення з основою D; N — кількість цифр у системі числення з основою D.

Результат: V — число, представлене у системі числення, у якій виконуються арифметичні операції множення та додавання.

Крок 1. $T \leftarrow d_{N-1}$.

Крок 2. Для i від 1 до $N-1$

Крок 3. $T \leftarrow T \cdot D,\ T \leftarrow T + d_{N-i-1}$.

Крок 4. Кінець циклу i.

Крок 5. $V \leftarrow T$.

Розглянемо роботу алгоритму 2 на прикладі числа $579d$, яке представимо у десятковій системі числення та яке потрібно представити у шістнадцятковій системі числення, тоді $N = 3$:

$$T \leftarrow 5d = 5h;$$

$$T \leftarrow T \cdot B = 5d \cdot 10d = 5h \cdot Ah = 32h = 50d;$$

$$T \leftarrow T + d_1 = 50d + 7d = 32h + 7h = 39h = 57d;$$
$$T \leftarrow T \cdot B = 57d \cdot 10d = 39h \cdot Ah = 23Ah = 570d;$$
$$T \leftarrow T + d_0 = 570d + 9d = 23Ah + 9h = 243h = 579d.$$

Отримуємо, що десяткове число $579d$ у шістнадцятковій системі числення представлено числом $243h$.

СКЛАДНІСТЬ АЛГОРИТМУ 2 ЗА КІЛЬКІСТЮ ОДНОРОЗРЯДНИХ ОПЕРАЦІЙ

Складність за кількістю операцій будемо обчислювати за припущення, що всі арифметичні операції (додавання, множення) виконуються у системі числення з основою B, в якій потрібно представити число, та кожна цифра (позиція) займає один розряд. Для спрощення також вважатимемо, що множення на D завжди є множенням на однорозрядне число, навіть якщо $D < B$.

Лема 2. Загальну кількість однорозрядних операцій алгоритму 2 можна обчислити так:

$$O^2_{mul}(N) = (N-1) \cdot N / 2,\ O^2_{add}(N) = (N-1) \cdot N / 2,$$

де N — кількість цифр числа, яке представлене у системі числення з основою D; O^2_{mul} та O^2_{add} — кількість однорозрядних операцій множення та додавання для послідовної моделі обчислень.

Доведення. Вхідним параметром алгоритму 2 є число, яке займає N позицій у системі числення з основою D. На початку алгоритму число T ініціалізується цифрою з найстаршої позиції числа. На першій ітерації число T буде мати один розряд, тому треба буде виконати одне однорозрядне множення та одне додавання цифри з молодшого розряду. На другій ітерації число T буде мати два розряди, для множення якого на однорозрядне число буде потрібно виконати дві однорозрядні операції множення та одну операцію додавання для отримання результату множення. Вважаємо, що знаки переносу, які виникають у результаті операції додавання, у послідовній моделі числення враховуються автоматично. На другій ітерації також потрібно додати наступну молодшу цифру до числа T. На останній ітерації відбувається множення однорозрядного числа D на число з $N-1$ розрядів, що потребує $N-2$ однорозрядних операцій додавання для отримання результату множення та одне додавання наймолодшої цифри. Загальна кількість однорозрядних операцій множення та додавання може бути обчислена на основі арифметичної прогресії

$$O^2_{mul}(N) = \frac{1+N-1}{2} \cdot (N-1),\ O^2_{add}(N) = \frac{0+N-2}{2} \cdot (N-1) + (N-1).$$

Після спрощення отримуємо очікуваний результат.

Лему доведено.

ПРЕДСТАВЛЕННЯ ЧИСЛА У СИСТЕМІ ЧИСЛЕННЯ НА ОСНОВІ ОПЕРАЦІЙ, ЯКІ ВИКОНУЮТЬСЯ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

Алгоритми 1 та 2 мають зв'язаність ітерацій. А саме результат наступної ітерації базується на результаті попередньої. Розглянемо метод, який дає змогу розпаралелити ітерації між процесорами та зменшує складність обчислення на кожному процесорі з квадратичної до лінійної складності.

Запишемо вираз (1) у вигляді

$$V = d_{N-1}D^{N-1} + d_{N-2}D^{N-2} + \ldots + d_1 D + d_0, \qquad (2)$$

де d_k, $k = \overline{0, N-1}$, — цифра у позиції k для системи числення з основою D; N — кількість цифр у числі у системі числення з основою D.

Уведемо додаткові позначення $D_{N-1} = D^{N-1}$, $D_{N-2} = D^{N-2}, \ldots, D_1 = D^1$, $D_0 = D^0$ та передобчислимо їх. Будемо вважати, що кожне число з $D_k = D^k$, $k = \overline{1, N-1}$, займає k розрядів відповідно.

Згідно з виразами леми 2 кількість однорозрядних операцій множення та додавання для кожного процесора у разі розподілення між N процесорами буде $(N-1) \cdot N / 2 / N = (N-1)/2$. Ця оцінка складності є апріорною.

Обчислення (2) може бути розподілено між N процесорами за допомогою наведеного далі алгоритму 3.

Алгоритм 3 переходу в систему числення з використанням операцій, якій виконуються у паралельній моделі обчислень.

Вхід: $\sum_{k=0}^{N-1}(d_k \cdot D^k)$ — число, представлене у системі числення з основою D; N — кількість цифр у системі числення з основою D; $D_k = D^k$, $k = \overline{0, N-1}$, — передобчислені значення.

Результат: V — число, представлене у системі числення, у якій виконуються арифметичні операції множення та додавання.

Крок 1. Для процесора k від 0 до $N-1$

Крок 2. $dD_k \leftarrow d_k \cdot D_k$.

Крок 3. Кінець циклу за k.

Крок 4. Для процесора k від 0 до $N-1$

Крок 5. $V(k) \leftarrow dD_k(k)$.

Крок 6. Для i від $k+1$ до $N-1$

Крок 7. $V(k) \leftarrow V(k) + dD_i(k)$.

Крок 8. Кінець циклу за i.

Крок 9. Кінець циклу за k.

У виразі $V(k) \leftarrow V(k) + dD_i(k)$ число k позначає індекс розряду, з яким відбувається операція.

Лема 3. Кількість однорозрядних операцій алгоритму 3 можна обчислити так:

$$O^3_{mul}(N) \le N-1,\ O^3_{add}(N) \le N-1,$$

де N — кількість цифр числа, яке представлено у системі числення з основою D; O^3_{mul} та O^3_{add} — найбільша кількість однорозрядних операцій множення та додавання, яку необхідно виконати одному з процесорів у паралельній моделі обчислень, у разі задіяння N паралельних процесорів.

Доведення. Особливістю алгоритму 3 є те, що обчислення не розподілені рівномірно між усіма процесорами. Це означає, що є процесор, який виконає тільки одну операцію однорозрядного множення, або процесор, який виконає тільки одну операцію однорозрядного додавання. Будемо розглядати процесори, які завантажені максимально, і обчислимо оцінку складності для таких процесорів. Одним із найбільш завантажених процесорів буде той, який обчислює значення $dD_{N-1} \leftarrow d_{N-1} \cdot D_{N-1}$. Такому процесору потрібно виконати $N-1$ однорозрядних операцій множення та $N-2$ однорозрядних операцій додавання для обчислення dD_{N-1}. На кроці 7 найбільше навантаження буде мати процесор, який обчислює суму всіх молодших розрядів $dD_k(0)$, $k = \overline{1, N-1}$, та цифри d_0. Для цього потрібно виконати $N-1$ однорозрядних операцій додавання. Вважаємо, що на кроці 7 знаки переносу автоматично враховуються тими процесорами, які менш завантажені.

Лему доведено.

Врахування знаків переносу у паралельній моделі обчислень [10, 11] є окремою задачею та може збільшувати загальну складність алгоритму.

В алгоритмі 3 кроки 1–3 та кроки 4–9 можна об'єднати в одному циклі замість двох циклів за змінною k, але для цього потрібні додаткові обчислення, оскільки числа dD_k, $k = \overline{0, N-1}$, мають різну довжину, що зменшує ефективність алгоритму.

Згідно з лемою 3 маємо максимальне завантаження у разі розподілення між N процесорами впродовж опрацювання числа довжиною N позицій у системі числення з основою D.

Зауважимо, що алгоритм 3 потребує додаткових $N(N-1)/2+1$ комірок пам'яті для чисел $D_k = D^k$, $k = \overline{0, N-1}$, та $N(N+1)/2$ комірок пам'яті для чисел $dD_k = d_k \cdot D^k$, $k = \overline{0, N-1}$. Загалом це потребує N^2+1 додаткових комірок пам'яті і є недоліком алгоритму 3.

ПРЕДСТАВЛЕННЯ ЧИСЛА У СИСТЕМІ ЧИСЛЕННЯ НА ОСНОВІ ІТЕРАЦІЙНОГО МЕТОДУ З ВИКОРИСТАННЯМ ОПЕРАЦІЙ, ЯКІ ВИКОНУЮТЬСЯ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

Якщо число має велику кількість позицій у своєму представленні, то процесорів може бути недостатньо для використання алгоритму 3. Розглянемо метод, який дає змогу кожному із задіяних процесорів опрацювати групи цифр, та вимагає менше додаткової пам‘яті у порівнянні з методом, який реалізується алгоритмом 3.

Запишемо вираз (1) у такому вигляді:

$$V = \sum_{k=0}^{R-1} \left(D^{k \cdot M} \cdot \sum_{i=0}^{M-1} (d_{k \cdot M + i} \cdot D^i) \right), \qquad (3)$$

де d_k, $k = \overline{0, N-1}$, — цифра у позиції k для системи числення з основою D; N — кількість цифр у числі у системі числення з основою D; R — кількість задіяних процесорів; M — кількість позицій для опрацювання одним процесором, $N = R \cdot M$.

Вираз (3) може бути швидко обчислений за умови передобчислення числа $DM = D^M$ за $R-1$ ітерацій з використанням алгоритму 4. Розглянутий далі покроковий опис алгоритму 4 для різних N за рахунок підбору числа M та кількості задіяних процесорів R у паралельній моделі обчислень дає змогу максимально враховувати довжину машинного слова під час його реалізації. Якщо відомо, що число M є оптимальним, але число N не ділиться без залишку на M, то додаються старші нулі до числа V.

Алгоритм 4 представлення числа у системі числення на основі ітераційного методу з використанням операцій, які виконуються у паралельній моделі обчислень.

Вхід: $\sum_{k=0}^{N-1} (d_k \cdot D^k)$ — число, представлене у системі числення з основою D; N — кількість цифр у системі числення з основою D; R — кількість задіяних процесорів; M — кількість цифр (позицій) для опрацювання одним процесором, $N = R \cdot M$; $DM = D^M$ — передобчислене значення; H — кількість позицій, яке займає число $DM-1$ у системі числення з основою B.

Результат: V — число у системі числення з основою B, у якій реалізовані арифметичні операції множення та додавання.

// Опрацювання своєї групи позицій кожним процесором.

Крок 1. Для процесора k від 0 до $R-1$

Крок 2. $ix \leftarrow (k+1) \cdot M - 1$.

Крок 3. $RD_k \leftarrow d_{ix}$.

Крок 4. Для i від 1 до $M-1$

Крок 5. $RD_k \leftarrow RD_k \cdot D + d_{ix-i}$.

Крок 6. Кінець циклу за i.

Крок 7. Кінець циклу за k.

// Ітераційне обчислення, яке синхронне з усіма процесорами.

Крок 8. Для i від 1 до $R-1$

Крок 9. Для процесора k від $R-i$ до $R-1$

Крок 10. $RD_k \leftarrow RD_k \cdot DM$.

Крок 11. Кінець циклу за k.

Крок 12. Для процесора k від $R-i$ до $R-1$

Крок 13. $RD_{k-1} \leftarrow RD_{k-1} + L(RD_k)$.

Крок 14. $RD_k \leftarrow H(RD_k)$.

Крок 15. Кінець циклу за k.

Крок 16. Кінець циклу за i.

// Збереження результату.

Крок 17. Для процесора k від 0 до $R-1$

Крок 18. Для i від 0 до $H-1$

Крок 19. $T(k \cdot H + i) \leftarrow RD_k(i)$.

Крок 20. Кінець циклу за i.

Крок 21. Кінець циклу за k.

Крок 22. $V \leftarrow T$.

Зауважимо, що $M \neq H$, тому що число у системі числення з іншою основою може займати іншу кількість позицій. На кроках 12–15 за рахунок операцій $L(RD_k)$ та $H(RD_k)$, які виокремлюють молодшу та старшу чистини числа RD_k, виконується вирівнювання за групами довжиною H у системі числення, у якій отримуємо число.

ПОЯСНЕННЯ ДО АЛГОРИТМУ 4

На кроках 1–7 алгоритму 4 відбувається представлення R частин числа, кожна з яких довжиною у M позицій, у системі числення з основою B. На цих кроках всі процесори працюють паралельно і навантаження збалансоване рівномірно. Після закінчення кроків 4–6 кожне з чисел RD_k, $k = \overline{0, R-1}$, буде довжиною H позицій. Передобчислення числа $DM = D^M$ є важливим, тому що результат його обчислення дає можливість отримати число H та розуміти, скільки є розрядів числа V у системі числення з основою B. Вибір числа M впливає на ефективність використання довжини машинного слова. Але він не повинен бути таким, щоб машинне слово було максимально заповненим після завершення кроків 4–6. Потрібно також враховувати, що оптимальний вибір числа M дає змогу автоматично враховувати знаки переносу, що зменшує складність реалізації. У разі недостатньої кількості процесорів алгоритм повинен оперувати більшою кількістю машинних слів.

Результат виконання кроків 1–7 можна представити у вигляді, як на рис. 1, де число, представлене у системі числення з основою D, займає 16 позицій. У разі задіяння чотирьох процесорів кожен з них представляє відповідну частину числа у системі числення з основою B довжиною у чотири позиції. На рис. 1 числами від 0 до 15 показано індекси елементів числа $\sum_{k=0}^{15} (d_k \cdot D^k)$, а числами від 0 до 3 — індекси елементів чисел RD_k, $k = \overline{0, 3}$.

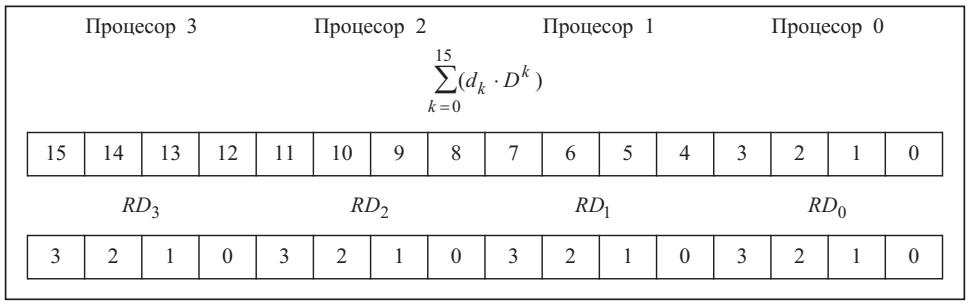

Рис. 1. Результат виконання кроків 1–7 алгоритму 4

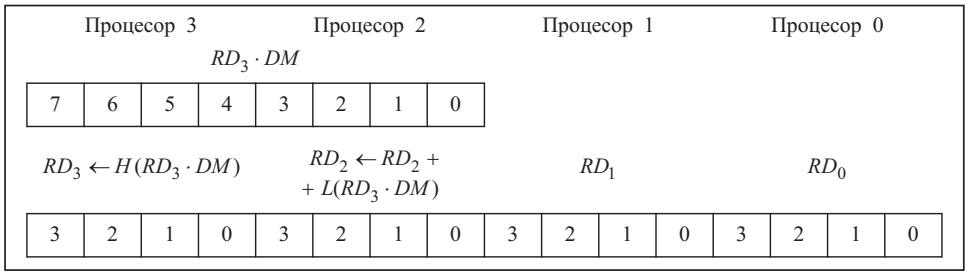

Рис. 2. Результат виконання кроків 8–16 алгоритму 4, $i = 1$

На кроках 8–16 навантаження на процесори не є рівномірним. На ітерації $i = 1$ працює тільки один процесор з індексом $R - 1$, який множить передобчислене число DM на RD_{R-1}. Як зображено на рис. 2 старша частина результату множення залишається у пам'яті процесора з індексом $R - 1$, а молодша частина додається до значення, яке зберігається у пам'яті процесора з індексом, меншим на одиницю, $R - 2$.

На кроках 8–16 найбільш навантаженим є процесор з індексом $R - 1$, який задіяно на всіх ітераціях під час виконання кроків 8–16. Як показано на рис. 3, на наступній ітерації з індексом $i = 2$ процесори з індексами $R - 1$ та $R - 2$ паралельно виконують множення передобчисленого числа DM на числа RD_{R-1} та RD_{R-2}. Важливим є те, що спочатку потрібно виконати всі множення і тільки потім додавати молодші частини результатів множення.

На рис. 4 показано результат виконання кроків 8–16 після ітерації з індексом $i = 2$. У результаті додавань молодших частин результатів множення можуть виникати знаки переносів, які потрібно враховувати у числах, опрацьова-

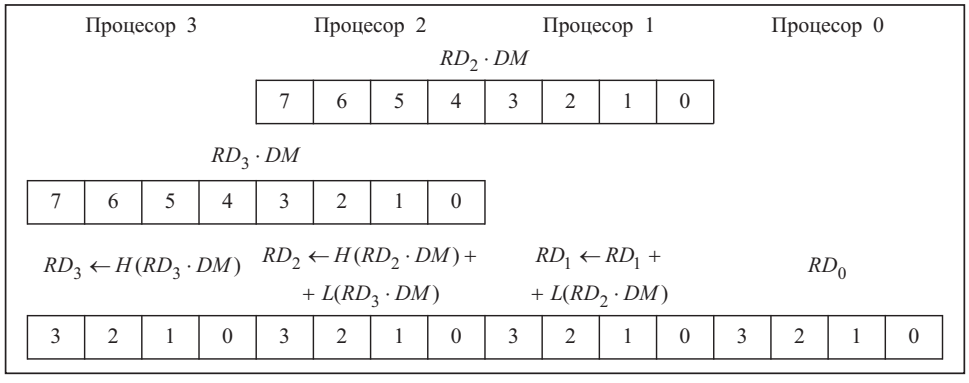

Рис. 3. Результат виконання кроків 8–16 алгоритму 4, $i = 2$

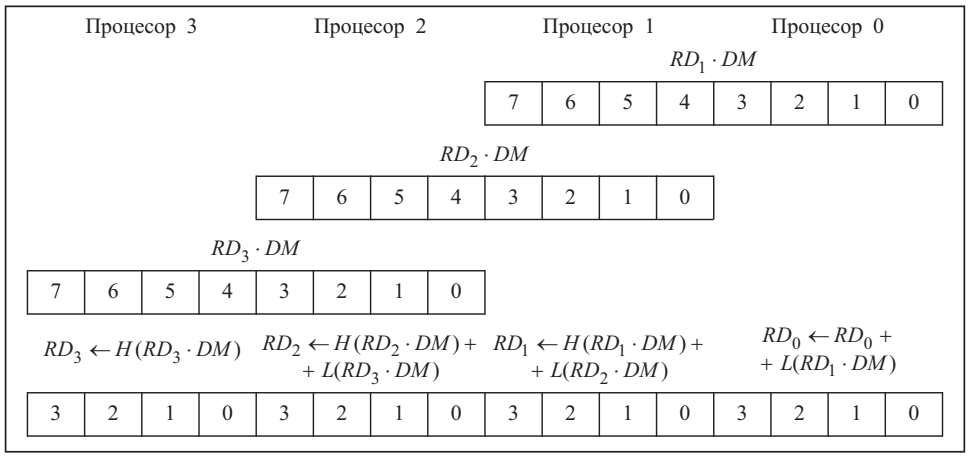

Рис. 4. Результат виконання кроків 8–16 алгоритму 4, $i = 3$

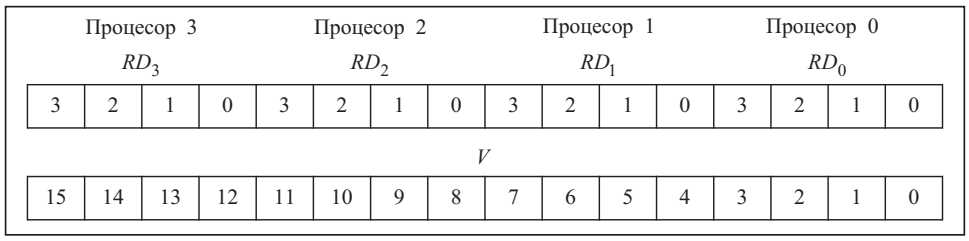

Рис. 5. Результат виконання кроків 17–22 алгоритму 4

них процесором з індексом більшим на одиницю. Щоб не перевантажувати алгоритм, вважатимемо, що знаки переносу враховуються автоматично у розглянутій моделі, але у практичній реалізації їх потрібно враховувати.

На кроках 17–22 результати обчислення на процесорах переносяться у загальний результат, як показано на рис. 5. На кроках 17–22 відбувається звичайний переніс слів. Під час практичної реалізації потрібно враховувати, що результати переповнення у старших розрядах на цьому етапі виконання алгоритму треба компенсувати за рахунок додавання до молодших розрядів чисел, які опрацьовують процесори з індексом більшим на одиницю.

АНАЛІЗ СКЛАДНОСТІ АЛГОРИТМУ НА ОСНОВІ ІТЕРАЦІЙНОГО МЕТОДУ З ВИКОРИСТАННЯМ ОПЕРАЦІЙ, ЯКІ ВИКОНУЮТЬСЯ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

Лема 4. Кількість однорозрядних операцій алгоритму 4 можна обчислити так:

$$O^4_{mul}(M,R,H) \le M \cdot (M-1)/2 + (R-1)H^2,$$

$$O^4_{add}(M,R,H) \le M \cdot (M-1)/2 + (R-1)(2H^2 - 2H),$$

де N — кількість позицій числа у системі числення з основою D; R — кількість задіяних процесорів; M — кількість позицій для опрацювання одним процесором, $N = R \cdot M$; $DM = D^M$ — передобчислене значення; H — кількість позицій, яке займає число $DM - 1$, у системі числення з основою B; O^4_{mul} та O^4_{add} — найбільша кількість однорозрядних операцій множення та додавання, яку потрібно виконати одному з процесорів у паралельній моделі обчислень, під час задіяння R паралельних процесорів.

Доведення. На кроках 1–7 всі процесори опрацьовують частини числа довжиною M позицій. Згідно з лемою 2 для представлення числа у системі числення з основою D кожному процесору потрібно виконати $(M-1)\cdot M/2$ операцій множення і додавання. Як було показано раніше, на кроках 8–16 найбільш завантаженим є процесор з індексом $R-1$, якщо індексація починається з нуля. Цей процесор виконає $R-1$ ітерацій. Кожна ітерація одного множення чисел довжиною H позицій потребує H^2 множень та $2H^2-2H$ додавань однопозиційних чисел з виконанням множення методом у стовпчик у системі числення з основою B, у якій реалізовано арифметичні операції множення та додавання. Тоді виконання $R-1$ ітерацій потребує $(R-1)\cdot H^2$ множень та $(R-1)\cdot(2H^2-2H)$ додавань однопозиційних чисел. Підсумовуючи кількість операцій на кроках 1–16, отримуємо очікувані оцінки.

Лему доведено.

Якщо кожне машинне слово містить k позицій $H=k\cdot W$, то можна стверджувати, що за умови $|M-H|<2$, загальна кількість машинних операцій множення та додавання, виконаних найбільш завантаженим процесором, зменшується приблизно у k^2 разів.

ПРЕДСТАВЛЕННЯ ЧИСЛА У СИСТЕМІ ЧИСЛЕННЯ НА ОСНОВІ РЕКУРСИВНОГО МЕТОДУ З ВИКОРИСТАННЯМ ОПЕРАЦІЙ, ЯКІ ВИКОНУЮТЬСЯ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

Алгоритм 4 потребує значно меншого обсягу оперативної пам'яті, ніж алгоритм 3, але розподіл обчислень між процесорами є нерівномірним у разі виконання алгоритму 4. Поряд з тим, що є процесор, який задіяний на всіх ітераціях, існує процесор, який взагалі не задіяний на ітераціях, де виконується множення. Наведений далі метод дає змогу задіяти всі процесори на ітераціях, де виконується множення.

Запишемо вираз (3) у такому вигляді:

$$V = D^{\frac{R}{2}\cdot M}\cdot\sum_{k=0}^{R/2-1}\left(D^{k\cdot M}\cdot\sum_{i=0}^{M-1}(d_{(k+R/2)\cdot M+i}\cdot D^i)\right)+ \\ +\sum_{k=0}^{R/2-1}\left(D^{k\cdot M}\cdot\sum_{i=0}^{M-1}(d_{k\cdot M+i}\cdot D^i)\right), \qquad (4)$$

де d_k, $k=\overline{0,N-1}$, — значення цифри у позиції k для системи числення з основою D; N — кількість цифр у числі у системі числення з основою D; R — кількість задіяних процесорів; M — кількість позицій для опрацювання одним процесором, $N=R\cdot M$.

Вираз (4) представляє позиції числа V у вигляді двох частин, кожну з яких опрацьовує половина паралельних процесорів. Старшу частину потім множать на $D^{(R/2)\cdot M}$ і додають молодшу частину.

Якщо число R ділиться без залишку на 4, то вираз (4) можна записати у вигляді чотирьох доданків, розділених попарно:

$$V = D^{\frac{R}{2}\cdot M}\cdot\left(D^{\frac{R}{4}\cdot M}\cdot\sum_{k=0}^{R/4-1}\left(D^{k\cdot M}\cdot\sum_{i=0}^{M-1}(d_{(k+3R/4)\cdot M+i}\cdot D^i)\right)+\right.$$

$$+\sum_{k=0}^{R/4-1}\left(D^{k\cdot M}\cdot\sum_{i=0}^{M-1}(d_{(k+R/2)\cdot M+i}\cdot D^{i})\right)\Bigg)+$$
$$+\Bigg(D^{\frac{R}{4}\cdot M}\cdot\sum_{k=0}^{R/4-1}\left(D^{k\cdot M}\cdot\sum_{i=0}^{M-1}(d_{(k+R/4)\cdot M+i}\cdot D^{i})\right)+$$
$$+\sum_{k=0}^{R/4-1}\left(D^{k\cdot M}\cdot\sum_{i=0}^{M-1}(d_{k\cdot M+i}\cdot D^{i})\right)\Bigg), \qquad (5)$$

де d_k, $k=\overline{0, N-1}$, — значення цифри у позиції k для системи числення з основою D; N — кількість цифр у числі у системі числення з основою D; R — кількість задіяних процесорів; M — кількість позицій для опрацювання одним процесором, $N = R \cdot M$.

Вираз (5) представляє позиції числа V у вигляді чотирьох частин, кожну з яких опрацьовує четверта частина паралельних процесорів.

Якщо число $R = 2^r$, то позиції числа V можуть бути представлені у вигляді R частин на початку виконання алгоритму та обчислені за r рекурсивних ітерацій.

Число V може бути швидко представлено у системі числення з іншою основою за умови передобчислення чисел $DM_r = D^m$, $m = 2^r \cdot M$, $r = \overline{0, \log_2 R - 1}$, які потребують $2R-1$ додаткових комірок пам'яті. Алгоритм 5 дає змогу розпаралелити обчислення у разі задіяння кількості паралельних процесорів, яка є степенем двійки.

Алгоритм 5 представлення числа у системі числення на основі рекурсивного методу з використанням операцій, які виконуються у паралельній моделі обчислень.

Вхід: $\sum_{k=0}^{N-1}(d_k \cdot D^k)$ — число, представлене у системі числення з основою D; N — кількість цифр у системі числення з основою D; R — кількість задіяних процесорів, $R = 2^r$; M — кількість позицій для опрацювання одним процесором, $N = R \cdot M$; $DM_r = D^m$, де $m = M \cdot 2^r$, $r = \overline{0, \log_2 R - 1}$, — передобчисленні значення; H — кількість позицій, яке займає число $D^M - 1$ у системі числення з основою B.

Результат: V — число у системі числення з основою B, у якій реалізовані арифметичні операції множення та додавання.

// Опрацювання своєї групи позицій кожним процесором.

Крок 1. Для процесора k від 0 до $R-1$

Крок 2. $ix \leftarrow (k+1)\cdot M - 1$.

Крок 3. $RD_k \leftarrow d_{ix}$.

Крок 4. Для i від 1 до $M-1$

Крок 5. $RD_k \leftarrow RD_k \cdot D + d_{ix-i}$.

Крок 6. Кінець циклу за i.

Крок 7. Кінець циклу за k.

// Рекурсивне обчислення.

Крок 8. Для r від 0 до $\log_2 R - 1$

Крок 9. $m \leftarrow 2^{r+1}$.

Крок 10. Для i від 0 до $R/m-1$

Крок 11. Для процесорів k від $m\cdot i$ до $m\cdot(i+1)-1$

Крок 12. $RD_{2\cdot i+1} \leftarrow RD_{2\cdot i+1} \cdot DM_r$.

Крок 13. Кінець циклу за k.

Крок 14. Кінець циклу за i.

Крок 15. Для i від 0 до $R/m-1$

Крок 16. Для процесорів k від $m\cdot i$ до $m\cdot(i+1)-1$

Крок 17. $RD_{2\cdot i+1} \leftarrow RD_{2\cdot i+1} + RD_{2\cdot i}$.

Крок 18. Кінець циклу за k.

Крок 19. Кінець циклу за i.

Крок 20. Для i від 0 до $R/m-1$

Крок 21. $RD_i \leftarrow RD_{2\cdot(i+1)-1}$.

Крок 22. Кінець циклу за i.

Крок 23. Кінець циклу за r.

// Збереження результату.

Крок 24. $V \leftarrow RD_0$.

Зауважимо, що значення RD_i, $i=\overline{0, R-1}$, містяться у загальній пам'яті, яка доступна всім процесорам, що є відмінністю алгоритму 5 від алгоритму 4. У разі реалізації алгоритму 4 дані містяться у власній пам'яті процесора під час виконання більшості кроків, що дає змогу використовувати кеш та значно прискорити обчислення. Іншою відмінністю алгоритму 5 є те, що кількість цифр у числах RD_i, $i=\overline{0, R-1}$, змінюється протягом виконання алгоритму.

АНАЛІЗ СКЛАДНОСТІ АЛГОРИТМУ НА ОСНОВІ РЕКУРСИВНОГО МЕТОДУ З ВИКОРИСТАННЯМ ОПЕРАЦІЙ, ЯКІ ВИКОНУЮТЬСЯ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

Лема 5. Кількість однорозрядних операцій алгоритму 5 можна обчислити так:

$$O^5_{mul}(M,R,H) = \frac{M}{2}(M-1) + H^2 \sum_{r=0}^{\log_2 R-1} 2^{r-1},$$

$$O^5_{add}(M,R,H) = \frac{M}{2}(M-1) + \sum_{r=0}^{\log_2 R-1} (2^r \cdot H^2 - H/2),$$

де N — кількість позицій у системі числення з основою D; R — кількість задіяних процесорів; M — кількість позицій для опрацювання одним процесором, $N = R\cdot M$; $DM_r = D^m$, $m = M\cdot 2^r$, $r=\overline{0, \log_2 R-1}$, — передобчисленні значення; H — кількість позицій, яка займає число D^M-1 у системі числення з основою B; O^5_{mul} та O^5_{add} — найбільша кількість однорозрядних операцій множення та додавання, яку потрібно виконати одному з процесорів у паралельній моделі обчислень, якщо задіяно R паралельних процесорів.

Доведення. На кроках 1–7 алгоритму 5 всі процесори опрацьовують частини числа довжиною M позицій. Згідно з лемою 2 для представлення числа у системі числення з основою D кожному процесору потрібно виконати $(M-1)\cdot M/2$ операцій множення і додавання однопозиційних чисел. На ітерації $r=0$ на кроках 10–14 одне множення чисел довжиною H позицій може бути виконано, якщо залучено пару процесорів, що зменшує обчислювальну складність удвічі для кожного з них. Здійснюючи множення методом у стовпчик, кожен процесор виконує $H^2/2$ множень та H^2-H додавань однопозиційних чисел. На ітерації $r=0$ на кроках 15–19, якщо залучено пару процесорів, кожен з них виконає $H/2$ операцій додавань. Операції переміщення даних між комірками на кроках 20–22 не будемо враховувати у загальній кількості операцій. На ітерації $r=1$ на кроках 10–14 одне множення чисел дов-

жиною $2H$ позицій може бути виконано вже у разі використання чотирьох процесорів. Кожен процесор виконає $4H^2 / 4 = H^2$ множень та $(8H^2 - 4H) / 4 = 2H^2 - H$ додавань однопозиційних чисел. На ітерації $r = 1$ на кроках 15–19 кожен із R процесорів виконає $2H / 4 = H / 2$ операцій додавань. Продовжуючи ітерацію r, на кроках 10–14 кожен процесор виконує $2^r \cdot H^2 / 2$ множень та $2^r \cdot H^2 - H$ додавань однопозиційних чисел та на кроках 15–19 виконує $H / 2$ однопозиційних операцій додавання. Загальну кількість операцій додавання та множення можна отримати за допомогою арифметичної прогресії

$$O^5_{mul}(M,R,H) = \frac{M}{2}(M-1) + \sum_{r=0}^{\log_2 R-1}\left(2^r \cdot \frac{H^2}{2}\right),$$

$$O^5_{add}(M,R,H) = \frac{M}{2}(M-1) + \sum_{r=0}^{\log_2 R-1}(2^r \cdot H^2 - H) + \sum_{r=0}^{\log_2 R-1}\frac{H}{2}.$$

Після спрощення отримуємо очікувані вирази.

Лему доведено.

ПРИКЛАД ПРЕДСТАВЛЕННЯ ЧИСЛА НА ОСНОВІ РЕКУРСИВНОГО МЕТОДУ З ВИКОРИСТАННЯМ ОПЕРАЦІЙ, ЯКІ ВИКОНУЮТЬСЯ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕНЬ

Представлення числа у шістнадцятковій системі числення наведено на прикладі числа 1 229 782 938 247 303 441, яке представлено у десятковій системі числення, у разі задіяння чотирьох процесорів, які виконують обчислення паралельно.

<table>
<tr><td>Крок 1–7</td><td>Процесор 3
$RD_3 \leftarrow$
$04CDh$
$(01229d)$</td><td>Процесор 2
$RD_2 \leftarrow$
$131D5h$
$(78293d)$</td><td>Процесор 1
$RD_1 \leftarrow$
$14229h$
$(82473d)$</td><td>Процесор 0
$RD_0 \leftarrow$
$00D71h$
$(03441d)$</td></tr>
<tr><td>Крок 10–14, $r = 0$</td><td colspan="2">Процесор 2, 3
$RD_3 \leftarrow 753\ 4E20h =$
$= 04CDh \cdot 186A0h$
$(01229d \cdot 100000d = 122\ 900\ 000d)$</td><td colspan="2">Процесор 0, 1
$RD_1 \leftarrow 1\,EB93\,CFA0h =$
$= 14229h \cdot 186A0h$
$(82473d \cdot 100000d = 8\ 247\ 300\ 000d)$</td></tr>
<tr><td>Крок 15–19, $r = 0$</td><td colspan="2">Процесор 2, 3
$RD_3 \leftarrow 754\ 7FF\ 5h =$
$= 753\ 4E20h + 1\ 31D5h$
$(0122978293d = 122\ 900\ 000d + 78\ 293d)$</td><td colspan="2">Процесор 0, 1
$RD_1 \leftarrow 1\,EB93\,DD11h =$
$= 1\,EB93\,CFA0h + 0D71h$
$(8247303441d = 8\ 247\ 300\ 000d + 3441d)$</td></tr>
<tr><td>Крок 20–22, $r = 0$</td><td colspan="2"></td><td colspan="2">$RD_0 \leftarrow RD_1$</td></tr>
<tr><td>Крок 20–22, $r = 0$</td><td colspan="2">$RD_1 \leftarrow RD_3$</td><td colspan="2"></td></tr>
<tr><td>Крок 10–14, $r = 1$</td><td colspan="4">Процесор 0, 1, 2, 3
$RD_1 \leftarrow 1111\ 110F\ 257D\ 3400h = 754\ 7FF\ 5h \cdot 2\ 540B\ E400h$
$(12297829300000000000d = 122\ 978\ 293d \cdot 10\ 000\ 000\ 000d)$</td></tr>
<tr><td>Крок 15–19, $r = 1$</td><td colspan="4">Процесор 0, 1, 2, 3
$RD_1 \leftarrow 1111\ 1111\ 1111\ 1111h\ 1111\ 110F\ 257D\ 3400h + 1\,EB93\,DD11h$
$(1229782938247303441 = 1\ 229\ 782\ 930\ 000\ 000\ 000d + 8\ 247\ 303\ 441d)$</td></tr>
<tr><td>Крок 20–22, $r = 1$</td><td colspan="4">$RD_0 \leftarrow RD_1$</td></tr>
<tr><td>Крок 24</td><td colspan="4">$V \leftarrow RD_0$</td></tr>
</table>

Рис. 6. Покроковий приклад обчислення на основі алгоритму 5

На початку роботи алгоритму 5 позиції числа потрібно розбити на чотири частини: 01229 78293 82473 03441, що потребує додавання одного старшого нуля.

На рис. 6 обчислення $RD_0 \leftarrow RD_1$ та $RD_1 \leftarrow RD_3$ наведено послідовно на кроках 20–22, $r = 0$ для того, щоб показати, що в процесі переміщення значень в RD_1 та з RD_1 операції зчитування та запису в ті самі комірки пам'яті не конкурують між собою.

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ АЛГОРИТМІВ ПІД ЧАС ПРЕДСТАВЛЕННЯ ЧИСЕЛ У ШІСТНАДЦЯТКОВІЙ СИСТЕМІ ЧИСЛЕННЯ З ДЕСЯТКОВОЇ СИСТЕМИ ЧИСЛЕННЯ

У табл. 1 наведено представлення у шістнадцятковій системі числення чисел, які відповідають круглим числам з N нулями у десятковій системі числення. В останній колонці табл. 1 наведено кількість бітів m, яка потрібна для збереження числа B^H.

З табл. 1 випливає, що кількість позицій N, яку займає число у десятковій системі числення, не є меншою кількості позицій H, яку займає таке саме число у шістнадцятковій системі числення, $H \le N$. З табл. 1 також можна зробити висновок, що кількість бітів m, які займає число у бітовому представленні не є пропорційним числу 4. Якщо позиції числа розбиваються на частини, кожна з яких займає дев'ять позицій у десятковій системі числення, то у двійковій системі достатньо 30 бітів для представлення максимального значення $10^9 - 1$. У цьому випадку ефективно використовувати 32-бітну арифметику. Якщо позиції числа розбиваються на частини, кожна з яких займає 10 позицій, то згідно з табл. 1 це потребує 34 біти і для реалізації потрібно використовувати 64-бітні операції.

Таблиця 1. Представлення у шістнадцятковій системі чисел, які відповідають числу з N нулями у десятковій системі числення

Кількості нулів, N	Кількість десяткових знаків, D^N	Представлення у шістнадцятковій системі числення, B^H	Кількість бітів, m
1	10	*Ah*	4
2	10^2	64*h*	7
3	10^3	3*E*8*h*	10
4	10^4	2710*h*	14
5	10^5	1 86*A*0*h*	17
6	10^6	*F* 4240*h*	20
7	10^7	98 9680*h*	24
8	10^8	5*F*5 *E*100*h*	27
9	10^9	3*B*9*A CA*00*h*	30
10	10^{10}	2 540*B E*400*h*	34
11	10^{11}	17 4876 *E*800*h*	37
12	10^{12}	*E*8 *D*4*A*5 1000*h*	40
13	10^{13}	918 4*E*72 *A*000*h*	44
14	10^{14}	5*AF*3 107*A* 4000*h*	47
15	10^{15}	3 8*D*7*E A*4*C*6 8000*h*	50
16	10^{16}	23 86*F*2 6*FC*1 0000*h*	54
17	10^{17}	163 4578 5*D*8*A* 0000*h*	57
18	10^{18}	*DE*0 *B*6*B*3 *A*764 0000*h*	60
19	10^{19}	8*AC*7 2304 89*E*8 0000*h*	64

ВИСНОВКИ

У роботі запропоновано методи реалізації операції представлення числа у системах числення для задач тестування операцій багаторозрядної арифметики у паралельній моделі обчислень. Розглянуто алгоритми представлення числа у системі числення на основі багаторозрядних операцій ділення та віднімання або багаторозрядних операцій множення та додавання. Вибір операцій залежить від того, чи виконуються операції у початковій системі числення чи у системі, у якій отримують число.

На основі десяткових операцій ділення та віднімання наведено приклад представлення десяткового числа у вигляді шістнадцяткового числа. На основі шістнадцяткових операцій множення та додавання наведено приклад представлення десяткового числа у вигляді шістнадцяткового числа. Запропоновано метод, який дає змогу розпаралелити обчислення між процесорами та зменшує складність обчислення для кожного процесора з квадратичної до лінійної складності, якщо задіяна кількість процесорів пропорційна довжині числа у цифрах. У разі реалізації алгоритму на основі такого методу кожен з N задіяних процесорів виконає не більше $N-1$ однорозрядних операцій множення та додавання для представлення числа довжиною N цифр в іншій системі числення.

Алгоритм з розбиттям числа на групи цифр дає змогу враховувати довжину машинного слова для процесорів з підтримкою арифметичних операцій різної довжини у бітах. На основі такого підходу операція знаходження представлення багаторозрядного числа розбивається на операції меншої розрядності з можливістю розподілення обчислень між процесорами.

Проаналізовано складність за кількістю операцій алгоритму на основі ітераційного методу. Такий алгоритм використовує додаткову пам'ять під час свого виконання. Також проаналізовано складність за кількістю операцій алгоритму на основі рекурсивного методу, який дає змогу розподіляти навантаження рівномірно між усіма процесорами, але є складнішим за кількістю кроків у порівнянні з алгоритмом на основі ітераційного методу.

Складність алгоритмів для паралельної моделі обчислень є лінійною, якщо кількість задіяних процесорів пропорційна кількості цифр у початковому числі, та є квадратичною від M та H (де M — кількість цифр у кожній групі, на які розбито початкове число, а H — кількість цифр у кожній групі, на які розбито отримане число). Ітераційний та рекурсивний методи є більш ефективними у паралельній моделі, у якій процесори виконують обчислення синхронно. Резервами оптимізації обчислень алгоритмів на основі ітераційного та рекурсивного методів є можливість використання передобчислених значень.

СПИСОК ЛІТЕРАТУРИ

1. Rivest R.L., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*. 1978. Vol. 21, N 2. P. 120–126. https://doi.org/10.1145/359340.359342.
2. Задірака В.К., Олексюк О.С. Комп'ютерна арифметика багаторозрядних чисел. Київ, 2003. 264 с.
3. Задирака В.К. Теория вычисления преобразования Фурье. Киев: Наук. думка, 1983. 213 с.
4. Задірака В.К., Терещенко А.М. Комп'ютерна арифметика багаторозрядних чисел у послідовній та паралельній моделях обчислень. Київ: Наук. думка, 2021. 136 с.

5. Хіміч О.М., Сидорук В.А. Використання мішаної розрядності у математичному моделюванні. *Математичне та комп'ютерне моделювання. Сер. Фіз.-мат. науки.* 2019. Вип. 19. С. 180–187.

6. Анісімов А.В. Алгоритмічна теорія великих чисел. Модулярна арифметика великих чисел. Київ: Академперіодика, 2001. 153 с.

7. Николайчук Я.М., Касянчук М.М., Якименко І.З., Івасьєв С.В. Ефективний метод модулярного множення в теоретико-числовому базисі Радемахера–Крестенсона. *Вісн. Нац. ун-ту «Львівська політехніка». Комп'ютерні системи та мережі*. 2014. № 806. С. 195–199. URL: http://nbuv.gov.ua/UJRN/VNULPKSM_2014_806_31.

8. Knuth D. The art of computer programming. Vol. 2. Seminumerical Algorithms. 3rd ed. Ch. 4. Arithmetic. "Positional Number Systems." Reading, Massachusetts: Addison–Wesley, 1997. P. 194–213.

9. Floyd R., Knuth D. Addition machines. *SIAM Journal on Computing*. 1990. Vol. 19, N 2. P. 329–340. https://doi.org/10.1137/0219022.

10. Анисимов А.В. Сложение без единиц переноса. *Кибернетика и системный анализ*. 1996. № 2. С. 3–15.

11. Терещенко А.Н., Задирака В.К. Параллельное сложение на основе векторных операций. *Штучний інтелект.* 2018. № 2. С. 122–137. URL: http://dspace.nbuv.gov.ua/handle/123456789/162381.

V.K. Zadiraka, A.M. Tereshchenko

PARALLEL METHODS OF REPRESENTING MULTIDIGIT NUMBERS IN NUMERAL SYSTEMS FOR TESTING MULTIDIGIT OPERATIONS

Abstract. The paper proposes methods for representing a multidigit number in a numeral system with a different base when using parallel processors. Representation of numbers in numeral systems based on multidigit operations of division and subtraction or multidigit operations of multiplication and addition is considered. The algorithm with the split of multidigit numbers into groups of digits, taking into account the length of the machine word, allows calculations involving parallel processors. The complexity by the number of operations and the amount of additional memory for the algorithms based on the iterative and recursive methods are analyzed.

Keywords: numeral system, multidigit arithmetic, multidigit addition, multidigit multiplication, parallel computing model.

Надійшла до редакції 30.06.2022