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.*

ЗНАХОДЖЕННЯ СУМИ БАГАТОРОЗРЯДНИХ ЧИСЕЛ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕННЯ

Анотація. Запропоновано новий метод реалізації операції знаходження суми двох і більше багатослівних доданків у паралельній моделі обчислення, який дає змогу звести знаходження суми великої кількості багатослівних доданків до операції знаходження двох багатослівних доданків за рахунок збереження знаків переносів для багатослівних чисел, ефективної в паралельній моделі обчислення на основі методу «прогнозування знаків переносів між групами слів». Запропоновано також алгоритми реалізації операції знаходження суми доданків на одному процесорі та k процесорах. Наведено аналіз складності таких алгоритмів.

Ключові слова: багаторозрядна арифметика, багаторозрядне додавання, знак переносу, паралельна модель обчислення.

ВСТУП

Використання нових паралельних обчислювальних систем, таких як багатоядерні процесори, графічні прискорювачі, кластери, розподілені системи, системи з розподіленою пам'яттю та інші, зумовлено потребою розв'язання складних прикладних задач у різних галузях. Серед них можна виділити задачі обчислення систем лінійних алгебричних рівнянь з кількістю невідомих 33–35 млн, моделювання фізичних процесів, аеродинаміки, захисту інформації тощо. Новітні технології значно розширюють використання багаторозрядної арифметики, тому що неврахування похибок заокруглення призводить до того, що іноді отримують комп'ютерні рішення, які не відповідають фізичному змісту. Багаторозрядна операція множення є складовою операції піднесення до степеня за модулем, від швидкодії якої залежить швидкодія асиметричних криптографічних програмно-апаратних комплексів [1–7]. У процесі реалізації операції множення час її виконання залежить від того, наскільки швидко можна додати два і більше великих чисел. У паралельній моделі обчислення час виконання операції додавання залежить від методу, на основі якого враховуються знаки переносу, які виникають в разі переповнення у відповідних розрядах (машинних словах) багаторозрядної суми двох чисел.

У роботі [8] подано аналіз імовірності появи знака переносу та запропоновано ефективний метод додавання не тільки додатних, а й від'ємних цілих чисел на основі інтерференційних суматорів, які дають змогу виконувати додавання N-бітових цілих чисел за $O(N) = N + K$ кроків, де K — кількість інтерференційних перетворень. У [9] запропоновано алгоритми ефективної реалізації

арифметичних функцій з використанням Addition Machines, швидкодія яких саме залежить від швидкодії операції додавання. У роботі [10] описано метод, який дає змогу додати три N-бітових цілих числа за $O(N) = 1 + 2\log_2(N+1)$ кроків у разі задіяння $N+1$ паралельних обчислювальних вузлів і за умови, що 1-бітова операція додавання містить один такт (один крок) процесора. Цей метод називається «методом зі збереженням знака переносу» (carry save addition). Однак метод [10] є ефективнішим для схемотехніки та апаратної реалізації, оскільки оперує бітами, а не машинними словами, якими оперують під час програмної реалізації.

У цій роботі наведено «метод зі збереженням знаків переносів для багатослівних чисел», що оперує словами та групами слів. Метод дає змогу звести операцію знаходження суми великої кількості N-розрядних доданків до операції додавання двох доданків (одного N-розрядного доданку та одного $N+1$-розрядного доданку), яку в паралельній моделі обчислення можна ефективно реалізувати на основі «методу передбачення знаків переносів між групами слів» [11].

ПОСТАНОВКА ЗАДАЧІ ТА ПОЗНАЧЕННЯ

Дано nm-бітові цілі додатні числа $X(d)$, $d = \overline{0, h-1}$. Потрібно побудувати швидкий алгоритм знаходження їхньої суми в паралельній моделі обчислення.

Нехай $X(d)$, $d = \overline{0, h-1}$, — nm-бітові цілі додатні числа, S — їхня $nm\lceil \log_2 h \rceil$-бітова сума. Число $X = \sum_{i=0}^{n-1}(X_i \cdot 2^{m \cdot i})$ представимо у вигляді $\{X_{n-1}, \ldots, X_0\}$, де $X_i = \sum_{b=0}^{m-1}(x_{im+b} \cdot 2^b)$, $i = \overline{0, n-1}$, є m-бітовим словом ($X_i = \{x_{im+m-1}, \ldots, x_{im}\}$, $i = \overline{0, n-1}$). Інші числа представимо так само: $\{X_{n-1}(d), \ldots, X_0(d)\}$, $d = \overline{0, h-1}$.

Далі, для зручності викладу, $X(d)$, $d = \overline{0, h-1}$, називатимемо n-слівними числами, кожне слово яких є m-бітовим. Відповідно число S буде n+1-слівним числом для $h < 2^m$.

Потрібно побудувати швидкий алгоритм знаходження суми $S = \sum_{d=0}^{h-1} X(d)$ в паралельній моделі обчислення.

ДОДАВАННЯ ДВОХ ЧИСЕЛ У ПАРАЛЕЛЬНІЙ МОДЕЛІ ОБЧИСЛЕННЯ

Розглянемо спочатку алгоритм додавання двох чисел.

Додавання n-слівних чисел X і Y без урахування знака між словами позначимо числом $M = \{M_{n-1}, \ldots, M_0\}$, де $M_i = \langle X_i + Y_i \rangle_W$, $W = 2^m$, $i = \overline{0, n-1}$.

Розбиття чисел X, Y на групи довжиною m слів формує слівні групи. Групи з індексом p мають такий вигляд: $\{X_{pm+m-1}, \ldots, X_{pm}\}$, $\{Y_{pm+m-1}, \ldots, Y_{pm}\}$. Розбиття на групи з m слів необхідне для знаходження надалі слів T_p та C_p довжиною m бітів для формування чисел T та C. Групи довжиною m слів з індексами $p-1$, p, $p+1$ називатимемо групами $p-1$, p, $p+1$ з m слів, випускаючи слово «з індексом».

У числі T біти t_i, $i = \overline{0, n-1}$, згруповані в m біт, що утворює $k = n/m$ слів у $T = \{T_{k-1}, \ldots, T_0\}$, де

$$T_p = \sum_{b=0}^{m-1}(t_i \cdot 2^b),\ t_i = \begin{Bmatrix} 1, \text{якщо } M_i = W-1 \\ 0, \text{якщо } M_i \neq W-1 \end{Bmatrix},\ i = pm + b,$$

$$p = \overline{0, k-1},\ k = n/m,\ W = 2^m. \qquad (1)$$

У числі C біти c_i, $i=\overline{0,n-1}$, згруповані в m біт, що утворює $k=n/m$ слів у $C=\{C_{k-1},\dots,C_0\}$, де

$$C_p=\sum_{b=\left\{\begin{array}{l}0,\text{ якщо } p>0\\1,\text{ якщо } p=0\end{array}\right\}}^{m-1}(c_i\cdot 2^b),\ c_i=\left\{\begin{array}{l}1,\text{ якщо } X_{i-1}+Y_{i-1}\ge W\\0,\text{ якщо } X_{i-1}+Y_{i-1}<W\end{array}\right\},\ i=pm+b,$$

$$p=\overline{0,k-1},\ k=n/m,\ c_0=0. \tag{2}$$

Знак переносу називатимемо «вхідним» під час перенесення з молодшого (попереднього) слова — знак переносу «входить» до поточного слова із молодшого слова. Знак переносу називатимемо «вихідним» під час перенесення з поточного слова до старшого (наступного) слова — знак переносу «виходить» з поточного слова до старшого слова.

АНАЛІЗ ЗНАКА ПЕРЕНОСУ ПІД ЧАС ДОДАВАННЯ ГРУП З m СЛІВ

Використання чисел T та C дає змогу прогнозувати виникнення знака переносу між групами з m слів, на які розбивається n-слівне число $X+Y$.

Лема 1. Якщо під час додавання в групі p з m слів $\{X_{pm+m-1},\dots,X_{pm}\}+$ $+\{Y_{pm+m-1},\dots,Y_{pm}\}$ однослівні числа C_p і T_p, отримані за формулами (1), (2), такі,

- що $C_p+T_p\ge W$, $W=2^m$, то в групі p з m слів виникає знак переносу до групи $p+1$ з m слів (група p «генерує» знак переносу в групі $p+1$);
- що $T_p=W-1$, $W=2^m$, то група p з m слів «поширює» знак переносу до групи $p+1$ з m слів, якщо знак перенесено з групи $p-1$ довжиною m слів (група p «поширює» знак переносу до групи $p+1$ з групи $p-1$);
- що $C_p+T_p<W$ та $T_p\ne W-1$, то в групі p ніколи не виникає знака переносу до групи $p+1$ (група p «поглинає» знак переносу з попередньої групи $p-1$).

Зауваження 1 [11]. Оскільки в групі p при сумі старших слів $X_{pm+m-1}+Y_{pm+m-1}$ може виникати знак переносу до групи $p+1$ з m слів, вважаємо, що сума $X_{pm+m-1}+Y_{pm+m-1}<W$.

Старший біт m-бітового числа C_p зі значенням одиниця на рис. 1 вказує на те, що сума слів, які передують старшим словам у групі p з m слів, така, що $X_{pm+m-2}+Y_{pm+m-2}\ge W$ (для $m=16$, $X_{p16+14}+Y_{p16+14}\ge W$). Тут m-бітове число C_p містить біти вхідних знаків переносів для поелементних сум m-слівних чисел $\{X_{pm+m-2},\dots,X_{pm-1}\}+\{Y_{pm+m-2},\dots,Y_{pm-1}\}$. Слова X_{pm-1}, Y_{pm-1} є найстаршими в групі $p-1$.

	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	C_p
	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	T_p
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Рис. 1. Приклад виникнення знака переносу в групі p з m слів до групи p+1 з m слів

На рис. 1 старший біт m-бітового числа T_p зі значенням одиниця означає, що справедливе рівняння $X_{pm+m-1}+Y_{pm+m-1}=W-1$ у групі p з m слів (для $m=16$, $X_{p16+15}+Y_{p16+15}=W-1$).

Для знаходження знаків переносів для кожного слова необхідно використовувати побітові операції. Умова, за якою відповідні біти двох m-бітових слів C_p і T_p дорівнюють одиниці, визначає, що знак переносу додається до $W-1$, і це «поширює» новий знак переносу до старшого (наступного) слова. Пропонується така побітова операція для ітераційного корегування вхідних знаків переносів: $C_p\leftarrow C_p\vee\langle(C_p\wedge T_p)<<1\rangle_W$, $W=2^m$, де операція «<<1» здійснює зсув значень бітів вліво (у бік старших бітів) на один біт. На рис. 2 показано результат

з використанням цієї операції. Для знаходження решти бітів числа C_p потрібно виконати додатково 15 ітерацій.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C_p
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T_p
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	C_p

Рис. 2. Корегування послівних вхідних знаків переносу C_p (у знаменнику) на основі вхідних знаків переносу C_p (у чисельнику) в групі p з m=16 слів

АЛГОРИТМ ОБЧИСЛЕННЯ СУМИ ДВОХ ЧИСЕЛ

Алгоритм 1. Обчислення суми чисел $X + Y$ довжиною nm бітів (або n слів) з передбаченням знаків переносу.

Вхід: числа $X = \{X_{n-1},\dots, X_0\}$, $Y = \{Y_{n-1},\dots, Y_0\}$, $k = n / m$, $W = 2^m$.

Результат: $S = \{S_{n-1},\dots,S_0\}$; $C_k = 1$, якщо $X + Y \ge 2^{nm}$.

Крок 1. Обчислити $M = \{M_{n-1},\dots, M_0\}$, де $M_i = \langle X_i + Y_i\rangle_W$, $i = \overline{0,n-1}$.

Крок 2. Обчислити $T = \{T_{k-1},\dots,T_0\}$, $C = \{0, C_{k-1},\dots, C_0\}$, де

$$T_p = \sum_{b=0}^{m-1}(t_i \cdot 2^b),\ C_p = \sum_{b=\begin{Bmatrix}0,\text{ якщо } p>0\\1,\text{ якщо } p=0\end{Bmatrix}}^{m-1}(c_i \cdot 2^b),\ t_i = \begin{Bmatrix}1,\text{ якщо } M_i = W-1\\0,\text{ якщо } M_i \ne W-1\end{Bmatrix},$$

$$c_i = \begin{Bmatrix}1,\text{ якщо } X_{i-1} + Y_{i-1} \ge W\\0,\text{ якщо } X_{i-1} + Y_{i-1} < W\end{Bmatrix},\ i = pm + b,\ p = \overline{0,k-1},\ k = n/m,\ W = 2^m,\ c_0 = 0.$$

// «Передбачення» знаків переносів між групами

Крок 3. Для $p \leftarrow 0$ до $p < k-1$

Крок 4. Якщо $C_p + T_p \ge W$, то $C_{p+1} \leftarrow C_{p+1} \vee 1$

Крок 5. Кінець циклу p.

Крок 6. Для $p \leftarrow 0$ до $p < k-1$

// Корегування послівних знаків переносів

Крок 7. Для $r \leftarrow 0$ до $r < m-1$

Крок 8. $C_p \leftarrow C_p \vee \langle (C_p \wedge T_p) << 1\rangle_W$.

Крок 9. Кінець циклу r.

// Обчислення результату

Крок 10. $c_{\text{temp}} \leftarrow C_p$.

Крок 11. Для $r \leftarrow 0$ до $r < m-1$

Крок 12. $Z_{pm+r} \leftarrow \langle M_{pm+r} + \langle c_{\text{temp}}\rangle_2\rangle_W$.

Крок 13. $c_{\text{temp}} \leftarrow c_{\text{temp}} >> 1$.

Крок 14. Кінець циклу r.

Крок 15. Кінець циклу p.

Кроки 1, 2, 4, 7–14 можуть виконуватися на k процесорах одночасно.

Для аналізу алгоритму 1 за кількістю однослівних операцій сформулюємо лему.

Лема 2 [11]. Якщо числа $X_{i-1} < W$, $Y_{i-1} < W$, де $W = 2^m$, такі, що $X_{i-1} + Y_{i-1} \ge W$, то $(XY < X_{i-1}) \vee (XY < Y_{i-1})$, де $XY = \langle X_{i-1} + Y_{i-1}\rangle_W$.

Зауваження 2. Лема дає змогу звести двослівну операцію додавання і порівняння $X_{i-1} + Y_{i-1} \ge W$ до однослівних операцій.

Теорема 1. В алгоритмі 1 загальна кількість однослівних операцій додавання, порівняння та побітових операцій становить $O_1(n) = 14n + 7k$, де $n = km$ — довжина доданка в словах, m — кількість бітів у слові.

Доведення. Вважатимемо, що всі однослівні операції додавання, порівняння та побітові операції виконуються за однаковий час. Тоді крок 1 містить n операцій. Операції обчислення за модулем $M_i = \langle X_i + Y_i \rangle_W$ не враховуємо, оскільки довжина числа $W = 2^m$ у бітах збігається з довжиною машинного слова. Для знаходження T на кроці 2 потрібно виконати n порівнянь $M_i = W - 1$ і стільки ж бітових операцій $t_i \cdot 2^b$. Для знаходження C на кроці 2 з урахуванням леми 2 необхідно виконати $2n$ операції однослівного порівняння ($XY < X_{i-1}$, $XY < Y_{i-1}$), n однослівних операцій додавання $\langle X_{i-1} + Y_{i-1} \rangle_W$, n логічних операцій «$\vee$» та n побітових операцій $c_i \cdot 2^b$, що становить $5n$ операцій. Загалом на кроці 2 потрібно виконати $7n$ операцій. На кроці 4 додавання $C_p + T_p$ є двослівною операцією, оскільки можливе переповнення результату додавання. За аналогією до кроку 2 вираз $C_p + T_p \ge W$ можна замінити виразом $(CT < C_p) \vee (CT < T_p)$, де $CT = \langle C_p + T_p \rangle_W$, для обчислення якого необхідно п'ять однослівних операцій. З урахуванням операції $C_{p+1} \leftarrow C_{p+1} \vee 1$ крок 5 загалом потребує $6k$ операцій. Для виконання циклу на кроці 8 потрібно $3km = 3n$ операції. Операції $\langle \ldots \rangle_W$ не враховуємо, тому що довжина числа $W = 2^m$ у бітах дорівнює довжині машинного слова. Крок 10 виконується k разів. Для обчислення кроків 12 та 13 потрібно виконати $3km = 3n$ однослівні операції.

Загальна кількість однослівних операцій, необхідних для виконання алгоритму 1, становить $n + 7n + 6k + 3n + k + 3n = 14n + 7k$.

Теорему доведено.

ЗНАХОДЖЕННЯ СУМИ БАГАТОРОЗРЯДНИХ ЧИСЕЛ

Обчислимо n-слівне число G, де $G_i = \sum_{d=0}^{h-1} X_i(d)$, $i = \overline{0, n-1}$. Зазначимо, що кожне G_i, $i = \overline{0, n-1}$, є двослівним числом для $h < 2^m$. У виразі (1) тепер має бути враховано більше багатослівних чисел. Побудуємо обчислення таким чином, щоб формули (1) були однаковими для будь-якої кількості чисел під час знаходження їхньої суми. Це можливо, якщо знаходження суми h n-слівних чисел зведено до знаходження суми одного n-слівного числа M та одного $n+1$-слівного числа H, де $H_{i+1} \leftarrow \lfloor G_i / W \rfloor$, $W = 2^m$, $i = \overline{n-1, 0}$, $H_0 \leftarrow 0$.

Додавання n-слівних чисел без урахування знака між словами також позначимо числом $M = \{M_{n-1}, \ldots, M_0\}$, але кожне слово обчислюється для всіх багатослівних чисел на основі чисел G та H, $M_i = \langle \langle G_i \rangle_W + H_i \rangle_W$, $W = 2^m$, $i = \overline{0, n-1}$. Обчислення T на основі числа M не змінюється: $T = \{T_{k-1}, \ldots, T_0\}$, де

$$T_p = \sum_{b=0}^{m-1} (t_i \cdot 2^b), \; t_i = \begin{Bmatrix} 1, \text{ якщо } M_i = W - 1 \\ 0, \text{ якщо } M_i \ne W - 1 \end{Bmatrix},$$

$$i = pm + b, \; p = \overline{0, k-1}, \; k = n/m, \; W = 2^m.$$

Обчислення C відрізняється від (2) в частині порівняння: $C = \{C_{k-1}, \ldots, C_0\}$, де

$$C_p = \sum_{b = \begin{Bmatrix} 0, \text{ якщо } p > 0 \\ 1, \text{ якщо } p = 0 \end{Bmatrix}}^{m-1} (c_i \cdot 2^b), \; c_i = \begin{Bmatrix} 1, \text{ якщо } H_{i-1} + \langle G_{i-1} \rangle_W \ge W \\ 0, \text{ якщо } H_{i-1} + \langle G_{i-1} \rangle_W < W \end{Bmatrix}, \; i = pm + b,$$

$$p = \overline{0, k-1}, \; k = n/m, \; c_0 = 0. \qquad (3)$$

Аналогічно до (1) та (2) біти згруповані в m біт, що утворює $k = n / m$ слів у числах C та T.

АЛГОРИТМ ЗНАХОДЖЕННЯ СУМИ ДОДАНКІВ НА ОСНОВІ «МЕТОДУ ЗІ ЗБЕРЕЖЕННЯМ ЗНАКІВ ПЕРЕНОСІВ ДЛЯ БАГАТОСЛІВНИХ ЧИСЕЛ»

За аналогією до «методу зі збереженням знаків переносів» [10], що оперує бітами, пропонується більш загальний алгоритм, який оперує групами слів та ефективно реалізується в паралельній моделі обчислення з використанням векторних операцій над групою довжиною 2, 4, 8, 16 слів.

Алгоритм 2. Знаходження суми h чисел $X(d)$, $d = \overline{0, h-1}$, довжиною nm біт (або n слів) зі «збереженням» та «передбаченням» знаків переносу.

Вхід: числа n, m, h, $k = n / m$, $W = 2^m$, $X(d)$, $d = \overline{0, h-1}$.

Результат: $S = \{S_n, \ldots, S_0\}$.

Крок 1. Обчислити $G = \{G_{n-1}, \ldots, G_0\}$, де $G_i = \sum_{d=0}^{h-1} X_i(d)$, $i = \overline{0, n-1}$.

Крок 2. Обчислити $H = \{H_n, \ldots, H_1, 0\}$, де $H_{i+1} \leftarrow \lfloor G_i / W \rfloor$, $i = \overline{0, n-1}$.

Крок 3. Обчислити $M = \{M_{n-1}, \ldots, M_0\}$, де $M_i = \langle G_i + H_i \rangle_W$, $i = \overline{0, n-1}$.

Крок 4. Обчислити $T = \{T_{k-1}, \ldots, T_0\}$, де $T_p = \sum_{b=0}^{m-1} (t_i \cdot 2^b)$,

$$t_i = \begin{Bmatrix} 1, \text{ якщо } M_i = W-1 \\ 0, \text{ якщо } M_i \neq W-1 \end{Bmatrix}, \; i = pm + b, \; p = \overline{0, k-1}, \; k = n / m.$$

Крок 5. Обчислити $C = \{0, C_{k-1}, \ldots, C_0\}$, де $C_p = \sum_{b=\begin{Bmatrix} 0, \text{ якщо } p>0 \\ 1, \text{ якщо } p=0 \end{Bmatrix}}^{m-1} (c_i \cdot 2^b)$,

$$c_i = \begin{Bmatrix} 1, \text{ якщо } H_{i-1} + \langle G_{i-1} \rangle_W \geq W \\ 0, \text{ якщо } H_{i-1} + \langle G_{i-1} \rangle_W < W \end{Bmatrix}, \; i = pm + b, \; p = \overline{0, k-1}, \; k = n / m.$$

// «Передбачення» знаків переносів між групами

Крок 6. Для $p \leftarrow 0$ до $p < k-1$

Крок 7. Якщо $C_p + T_p \geq W$, то $C_{p+1} \leftarrow C_{p+1} \vee 1$.

Крок 8. Кінець циклу p.

Крок 9. $S_n \leftarrow H_n + C_k$.

Крок 10. Для $p \leftarrow 0$ до $p < k-1$

// Корегування послівних знаків переносів

Крок 11. Для $r \leftarrow 0$ до $r < m-1$

Крок 12. $C_p \leftarrow C_p \vee \langle (C_p \wedge T_p) << 1 \rangle_W$.

Крок 13. Кінець циклу r.

// Обчислення результату

Крок 14. $c_{\text{temp}} \leftarrow C_p$.

Крок 15. Для $r \leftarrow 0$ до $r < m-1$

Крок 16. $S_{pm+r} \leftarrow \langle M_{pm+r} + \langle c_{\text{temp}} \rangle_2 \rangle_W$.

Крок 17. $c_{\text{temp}} \leftarrow c_{\text{temp}} >> 1$.

Крок 18. Кінець циклу r.

Крок 19. Кінець циклу p.

Кроки 1–5, 7, 9, 11–18 можуть виконуватися на k процесорах одночасно.

Теорема 2. В алгоритмі 2 загальна кількість однослівних операцій додавання, порівняння та побітових операцій становить $O_2(n) = 2nh + 17n + 7k + 1$, де h — кількість доданків, $n = km$ — довжина доданка в словах, m — кількість бітів у слові.

Доведення. Вважатимемо, що всі однослівні операції додавання, порівняння та побітові операції виконуються за однаковий час. Порівняно з алгоритмом 1 кроки 1 та 2 необхідні для збереження знаків переносів. Для цього використовується число H. Крок 1 послівного додавання потребує $2nh$ операції, враховуючи, що для додавання двох слів потрібна додаткова операція для збереження знака переносу. На кроці 2 результат операції $\lfloor G_i / W \rfloor$, яка виділяє старшу частину двослівної суми, і на кроці 3 результат операції обчислення за модулем $M_i = \langle X_i + Y_i \rangle_W$ збігаються з довжиною машинного слова $W = 2^m$, тому кроки 2 та 3 не потребують арифметичних операцій, потрібна тільки операція збереження. Для обчислення T на кроці 4 необхідно виконати n порівнянь $M_i = W - 1$ і стільки ж бітових операцій $t_i \cdot 2^b$, що становить $2n$ операції. Для обчислення C на кроці 5 потрібно виконати n операцій однослівного порівняння $H_{i-1} + \langle G_{i-1} \rangle_W \geq W$ та n побітових операцій $c_i \cdot 2^b$, що становить $2n$ операції. На кроці 7 додавання $C_p + T_p$ є двослівною операцією, оскільки можливе переповнення результату додавання. З урахуванням леми 2 вираз $C_p + T_p \geq W$ можна замінити виразом $(CT < C_p) \vee (CT < T_p)$, де $CT = \langle C_p + T_p \rangle_W$, для обчислення якого необхідно п'ять однослівних операцій. З урахуванням операції $C_{p+1} \leftarrow C_{p+1} \vee 1$ крок 7 загалом потребує $6k$ операцій. На кроці 9, який потребує одну операцію додавання, завершується прогнозування знаків переносів. Для виконання циклу корегування всіх послівних знаків переносів на кроці 12 необхідно $3km = 3n$ операції. Операції $\langle \ldots \rangle_W$ не враховуємо, тому що довжина числа $W = 2^m$ у бітах дорівнює довжині машинного слова. Крок 14 виконується k разів. Для обчислення остаточного результату суми на кроках 16 та 17 потрібно виконати $3km = 3n$ однослівні операції.

Загальна кількість однослівних операцій, необхідних для виконання алгоритму 2, становить

$$2nh + 2n + 2n + 7n + 6k + 1 + 3n + k + 3n = 2nh + 17n + 7k.$$

Теорему доведено.

АЛГОРИТМ ЗНАХОДЖЕННЯ СУМИ ДОДАНКІВ У РАЗІ РОЗПОДІЛЕННЯ ОБЧИСЛЕНЬ

У попередньому розділі зазначено, що алгоритм 2 може виконуватися на k паралельних процесорах одночасно. Далі надано алгоритм для виконання на кожному паралельному процесорі. Передається додатковий вхідний параметр ix ($ix = \overline{0, k-1}$), який є порядковим номером процесора та необхідним для розмежування груп слів, які опрацьовує кожен з k паралельних процесорів.

Алгоритм 3. Знаходження суми h чисел $X(d)$, $d = \overline{0, h-1}$, довжиною nm бітів (або n слів) зі збереженням та передбаченням знаків переносу в разі розподілення обчислень.

Вхід: числа n, m, h, $k = n / m$, $W = 2^m$; $X(d)$, $d = \overline{0, h-1}$; $H_i \leftarrow 0$, $i = \overline{0, n}$; $C_j \leftarrow 0$, $j = \overline{0, k}$; ix, $i_{ix} = ix \cdot m$, $i_{ix+1} = (ix+1) \cdot m$.

Результат: $S = \{S_n, \ldots, S_0\}$.

Крок 1. Обчислити $G_i = \sum_{d=0}^{h-1} X_i(d)$, $i = \overline{i_{ix}, i_{ix+1} - 1}$.

Крок 2. Обчислити $H_{i+1} \leftarrow \lfloor G_i / W \rfloor$, $i = \overline{i_{ix}, i_{ix+1} - 1}$.

Крок 3. Обчислити $M_i = \langle G_i + H_i \rangle_W$, $i = \overline{i_{ix}, i_{ix+1} - 1}$.

Крок 4. Обчислити $T_p = \sum_{b=0}^{m-1}(t_i \cdot 2^b)$, $t_i = \begin{Bmatrix} 1, \text{якщо } M_i = W-1 \\ 0, \text{якщо } M_i \neq W-1 \end{Bmatrix}$,

$i = pm + b$, $p = ix$.

Крок 5. Обчислити $C_p = \sum_{b=\begin{Bmatrix} 0, \text{якщо } p>0 \\ 1, \text{якщо } p=0 \end{Bmatrix}}^{m-1}(c_i \cdot 2^b)$,

$$c_i = \begin{Bmatrix} 1, \text{якщо } H_{i-1} + \langle G_{i-1} \rangle_W \geq W \\ 0, \text{якщо } H_{i-1} + \langle G_{i-1} \rangle_W < W \end{Bmatrix}, \quad i = pm + b, \ p = ix.$$

// «Передбачення» знаків переносів між групами

Крок 6. Для $p \leftarrow 0$ до $p < k-1$

Крок 7. Якщо $C_p + T_p \geq W$, то $C_{p+1} \leftarrow C_{p+1} \vee 1$.

Крок 8. Кінець циклу p.

Крок 9. $S_n \leftarrow H_n + C_k$.

Крок 10. $p \leftarrow ix$.

// Корегування послівних знаків переносів

Крок 11. Для $r \leftarrow 0$ до $r < m-1$

Крок 12. $C_p \leftarrow C_p \vee \langle (C_p \wedge T_p) << 1 \rangle_W$.

Крок 13. Кінець циклу r.

// Обчислення результату

Крок 14. $c_{\text{temp}} \leftarrow C_p$.

Крок 15. Для $r \leftarrow 0$ до $r < m-1$

Крок 16. $S_{pm+r} \leftarrow \langle M_{pm+r} + \langle c_{\text{temp}} \rangle_2 \rangle_W$.

Крок 17. $c_{\text{temp}} \leftarrow c_{\text{temp}} >> 1$.

Крок 18. Кінець циклу *r*.

Теорема 3. В алгоритмі 3 загальна кількість однослівних операцій додавання, порівняння та побітових операцій становить $O_3(n) = 2mh + 17m + 6k + 2$, де h — кількість доданків, $n = km$ — довжина доданка в словах, m — кількість бітів у слові.

Доведення. Вважатимемо, що всі однослівні операції додавання, порівняння та побітові операції виконуються за однаковий час. Порівняно з алгоритмом 2 майже всі обчислення розподілені на k паралельних процесорах, тому кількість операцій зменшується в k разів. Кроки 6–8 виконуються всіма процесорами одночасно стільки разів, скільки k паралельних процесорів (або груп слів), для поширення знака переносу з наймолодшої групи слів до найстаршої групи слів. Крок 9 виконується всіма процесорами, що ефективніше за введення додаткової умови для виконання операції на виділеному процесорі.

З урахуванням наведеного загальна кількість однослівних операцій, необхідних для виконання алгоритму 3, становить

$$2nh / k + 2n / k + 2n / k + 7n / k + (6k) + (1) + 3n / k + k / k + 3n / k =$$
$$= 2mh + 17m + 6k + 2.$$

У дужках позначені операції, які не розподіляються між процесорами.

Теорему доведено.

У разі $m = 16$, де m — довжина вектора у словах, маємо $O_3(n) = 32h + 6k + 274$.

ВИСНОВКИ

У роботі запропоновано новий метод реалізації операції знаходження суми двох і більше багатослівних чисел у паралельній моделі обчислення. Якщо «метод зі збереженням знаків переносів» та «метод передбачення знаків пере-

носів» [10] оперують бітами, то метод, що описано, оперує словами та групами слів. Цей метод дає змогу звести операцію знаходження суми великої кількості N-розрядних доданків до операції додавання двох доданків (одного N-розрядного доданка та одного $N+1$-розрядного доданка), яку в паралельній моделі обчислення можна ефективно реалізувати на основі «методу передбачення знаків переносів між групами слів» [11]. Надано загальний алгоритм 2 обчислення суми доданків для виконання на одному процесорі та алгоритм 3 в разі розподілення обчислення на k процесорах. Наведено аналіз складності запропонованих алгоритмів у вигляді теорем. У теоремі 3 доведено, що складність алгоритму 3 за кількістю однослівних операцій має лінійні залежності від кількості задіяних процесорів та кількості багатослівних доданків. Для $m=16$ складність алгоритму 3 можна записати як $O_3(n)=32h+6k+274$, де m — довжина вектора в словах, h — кількість доданків, k — кількість задіяних процесорів.

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. Анісімов А.В. Алгоритмічна теорія великих чисел. Модулярна арифметика великих чисел. Київ: Академперіодика, 2001. 153 с. URL : http://books.zntu.edu.ua/book_info.pl?id=21106.
3. Задірака В., Олексюк О. Комп'ютерна арифметика багаторозрядних чисел. Київ: Наук. видання, 2003. 263 с.
4. Задирака В.К. Теория вычисления преобразования Фурье. Киев: Наук. думка, 1983. 213 с.
5. Задірака В.К., Терещенко А.М. Комп'ютерна арифметика багаторозрядних чисел у послідовній та паралельній моделях обчислень. Київ: Наук. думка, 2021. 136 с.
6. 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.
7. Хіміч О.М., Сидорук В.А. Використання мішаної розрядності у математичному моделюванні. *Математичне та комп'ютерне моделювання. Серія: Фіз.-мат. науки. Зб. наук. праць*. 2019. Вип. 19. С. 180–187. https://doi.org/10.32626/2308-5878.2019-19.180-187.
8. Анисимов А.В. Сложение без единиц переноса. *Кибернетика и системный анализ*. 1996. № 2. С. 3–15.
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. McGeoch C. Parallel addition. *The American Mathematical Monthly*. 1993. Vol. 100, N 9. P. 867–871. URL: http://www.jstor.org/stable/2324666.
11. Терещенко А.Н., Задирака В.К. Параллельное сложение на основе векторных операций. *Искусственный интеллект*. 2018. № 2. С. 122–137. URL: http://dspace.nbuv.gov.ua/handle/123456789/162381.

V.K. Zadiraka, A.M. Tereshchenko

CALCULATING THE SUM OF MULTIDIGIT VALUES IN A PARALLEL COMPUTATIONAL MODEL

Abstract. The authors propose a new method for implementing the operation of finding the sum of two or more multidigit values in a parallel computational model. The method reduces finding the sum of a large number of multidigit values to the sum of two multidigit values by carry-save addition, which can be efficiently implemented in a parallel computational model based on carry-lookahead addition of groups of words. The algorithms for implementing the operation of the sum of values on one processor and using k processors are proposed. The complexity analysis is carried out for the proposed algorithms.

Keywords: multidigit arithmetic, multidigit addition, carry sign, parallel computational model.

Надійшла до редакції 10.01.2022