КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.6 DOI:10.34229/2707-451X.21.2.4

А.М. ТЕРЕЩЕНКО, В.К. ЗАДІРАКА

ГЕНЕРУВАННЯ ВЕЛИКИХ ЧИСЕЛ ДЛЯ ТЕСТУВАННЯ АЛГОРИТМІВ БАГАТОРОЗРЯДНОЇ АРИФМЕТИКИ

Наведено прості залежності, використовуючи які можна візуально перевірити правильність обчислення багаторозрядних операцій додавання, віднімання, множення, множення за модулем та піднесення до степеня за модулем. Наведено алгоритми генерування таких вхідних та вихідних багаторозрядних даних, що дозволяють перевіряти цілісність результату при делегуванні обчислень у розподілені системи.

Ключові слова: *багаторозрядна арифметика, паралельна модель обчислення.*

Вступ. Поява нових паралельних обчислювальних систем таких, як багатоядерні процесори, графічні прискорювачі, кластери, розподілені системи, системи з розподіленою пам'яттю та інші обумовлена вирішенням складних прикладних задач у різних галузях. Серед таких задач можна виділити задачі обчислення систем лінійних алгебраїчних рівнянь з кількістю невідомих 33–35 мільйонів, розрахунок оболонок ядерних реакторів, моделювання фізичних, хімічних процесів, аеродинаміки, гідродинаміки, захисту інформації тощо. Це значно розширює використання багаторозрядної арифметики, із-за того, що неврахування похибок заокруглення приводить до того, що іноді отримуються комп'ютерні рішення, які не відповідають фізичному змісту. Операція багаторозрядного множення є складовою операції піднесення до степеня за модулем, від її швидкодії залежить швидкодія асиметричних криптографічних програмно-апаратних комплексів. Розмаїття існуючих методів розпаралелювання обчислення операцій багаторозрядної арифметики зумовлено відмінністю пристроїв, для яких реалізуються паралельні алгоритми. Як наслідок, це потребує включення бібліотеки програм, яка виконує операції над багаторозрядними числами, у штатне математичне забезпечення сучасних багатопроцесорних систем. У більшості випадків вартість розробки програмного забезпечення не є меншою ніж вартість обладнання, на якому розв'язується задача.

Дуже часто не має можливості розробити новий алгоритм, тому що на цьому етапі ще не має тестових даних, за допомогою яких можна проаналізувати результат розв'язання задачі. Тому задача з підготовки тестових даних та результатів є не менш важливою ніж сама розробка алгоритму. Від якості підготовлених даних залежить якість розробленого алгоритму та час необхідний для знаходження та усунення помилок алгоритму-програми та його реалізації. Під якістю тестових даних розуміється наступне: для знаходження ефективного алгоритму дуже важливо мати такі тестові дані, структура яких не буде впливати на швидкодію виконання алгоритму Або такі тестові дані,

які допомагають швидше локалізувати помилку (наприклад, при виникненні знаку переносу). Під якістю реалізації алгоритму мається на увазі, що реалізація алгоритму буде ефективною (або найшвидшою) для великої кількості граничних випадків (наприклад, всі біти є нулі, або всі біти є одиницями) та більшого діапазону довжин вхідних даних. Тому дуже важливо мати вже підготовленим великий набір тестових даних для тестування граничних випадків та тестових даних для різної довжини вхідних даних. Але якщо розглядається розробка та реалізація багаторозрядного алгоритму для широкого діапазону даних, наприклад, від 1000 до 4000 бітів з кроком 128 бітів, то етап з підготовки тестових вхідних та вихідних даних може бути одним з найбільших за тривалістю етапів з розробки алгоритмів. Так, наприклад, підготовка тестових даних для паралельної моделі даних займає більше часу, ніж розробка алгоритму у послідовній моделі обчислень, так як для отримання тестових даних необхідно реалізувати алгоритми у послідовній моделі обчислень з використанням однакових технологій, а саме мови програмування для спрощення інтеграції при тестуванні. Зрозуміло, що зменшення часу на підготовку тестових даних, дає можливість приділити більше часу на більш якісну реалізацію алгоритму. Тому найбільш цікавими є такі тестові багаторозрядні дані, які при генерації мають дуже прості залежності між вхідними та вихідними даними для даної операції і дають можливість їх відтворення «вручну» без використання спеціальних програмних та апаратних засобів. Одним з таких прикладів можна навести, розглядаючи множення двозначного числа на 11. Щоб помножити будь-яке число на 11, треба між першою та другою цифрами множника вписати суму першої та другої цифри. Наприклад: **23**$\cdot$11, пишемо 2 і 3, а між ними ставимо суму (2 + 3). **23**$\cdot$ 11 = **2**(2 + 3)**3** = **253**, **34**$\cdot$ 11 = **374**.

У даній роботі наведені деякі прості залежності, використовуючи які можна візуально перевірити правильність виконання обчислення багаторозрядних операцій додавання, віднімання, множення, множення за модулем та піднесення до степеня за модулем. З урахування наведених залежностей наведені прості алгоритми генерування вхідних та вихідних багаторозрядних даних.

Паралельна модель обчислень. Під паралельною моделлю обчислення розглядається модель GPU (Graphics Processing Unit) відеоплати. Фізично це є окремий пристрій, який інсталюється додатково в комп'ютер. GPU побудована за технологією SIMD (Single Instruction–Multiple Data), де потокові процесори можуть виконувати одну інструкцію одночасно, оперуючи з різними даними. В такій моделі 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. При збільшенні кількості задіяних процесорів всі обчислення необхідно розбивати на декілька робочих груп, між якими необхідно виконувати синхронізацію, яка є дуже витратною операцією. Якщо у послідовній моделі обчислень операції знаку переносу можна не враховувати, так як знак переносу враховується в операції додавання автоматично, то в паралельній моделі обчислень врахування знаку переносу потребує використання спеціальної логіки, яка збільшує складність обчислення не менш ніж у два рази. Крім того, що знак переносу потрібно враховувати між розрядами, його також потрібно переносити і враховувати між даними, які опрацьовують різні процесори або різні групи процесорів. При підготовці даних необхідно мати такі тестові дані, які окрім помилок обчислення допомагають виявляти помилки синхронізації даних та легко локалізувати частини алгоритму, де виникають такі помилки. Зменшення часу на підготовку таких даних значно зменшує час на реалізацію алгоритму та є окремою задачею і дає можливість приділити більше часу знаходженню більш ефективної реалізації.

Постановка задачі. Багаторозрядними числами довжиною в N розрядів будемо називати числа, кожен з N розрядів (або слів) якого має 2^m, $m=\overline{3,9}$ бітів. Тобто в кожен розряд може бути записано число у діапазоні від 0 до $2^{2^m}-1$, а саме багаторозрядне число буде мати вигляд

$$U_N=\sum_{k=0}^{N-1}u_k 2^{nk},\ u_k=\overline{0,2^n-1},\ \ n=2^m,\ m=\overline{3,9}.$$

Наша задача визначити залежності між вхідними та вихідними даними, які є багаторозрядними цілими додатними числами та для яких існує простий алгоритм їх генерування, для перевірки правильності обчислення багаторозрядних операцій додавання, віднімання, множення, множення за модулем та піднесення до степеня за модулем та тестування модулів складних алгоритмів, які оперують даними з розмірами у межах від 1000 до 4000 бітів.

Операція додавання. Згенеруємо багаторозрядні цілі додатні числа для перевірки правильності обчислення операції додавання.

Лема 1. Для чисел вигляду $U_N = V_N = \sum_{k=0}^{N-1} 2^{nk}$, де $n = 2^m$, $m = \overline{3,9}$, N – ціле додатне число, результатом виконання операції додавання $U_N + V_N$ буде число вигляду $R_N = 2\sum_{k=0}^{N-1} 2^{nk}$.

У силу тривіальності доведення не наведено. Так як процесор може оперувати такими даними, як біти, байти, слова, подвійні слова, то для зручності сприйняття будемо використовувати 16-річну систему обчислення та додавати букву *h* для її позначення.

Для перевірки виконаємо операцію у 16-річній системі обчислення. При $N = 8$, $n = 16$ (загалом 128 бітів) в операції додавання будуть використані числа U_8, V_8, R_8.

$$U_8 = V_8 = \sum_{k=0}^{7} 2^{16k} = 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\text{h},$$

$$R_8 = 2\sum_{k=0}^{7} 2^{16k} = 0002\ 0002\ 0002\ 0002\ 0002\ 0002\ 0002\ 0002\text{h}.$$

Генерація даних. Приклад 1. $N = 8$, $n = 16$. Генерація чисел для тестування операції додавання.

	0001 0001 0001 0001 0001 0001 0001 0001h	U_8
+	0001 0001 0001 0001 0001 0001 0001 0001h	V_8
	0002 0002 0002 0002 0002 0002 0002 0002h	R_8

Лема 2. Для чисел вигляду

$$U_N = \sum_{k=0}^{N-1} 2^{nk},\ V_N = \sum_{k=0}^{N-1} ((2^n - 1)\cdot 2^{nk}),$$

де $n = 2^m$, $m = \overline{3,9}$, N – ціле додатне число, результатом виконання операції додавання $U_N + V_N$ буде число вигляду $R_N = \sum_{k=1}^{N-1} 2^{nk}$. Знак переносу у самий старший розряд не враховується.

Доведення. У виразі

$$U_N + V_N = \sum_{k=0}^{N-1} 2^{nk} + \sum_{k=0}^{N-1} ((2^n - 1)\cdot 2^{nk}) = \sum_{k=0}^{N-1} (((2^n - 1 + 1))\cdot 2^{nk}) = \sum_{k=1}^{N} 2^{nk}$$

індекс $k = N$ згідно умови не враховуємо. Лема доведена.

Генерація даних. Приклад 2. $N=8$, $n=16$. Генерація чисел для тестування операції додавання.

	0001	0001	0001	0001	0001	0001	0001	0001h	U_8
+	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFFh[1]	V_8
	0001	0001	0001	0001	0001	0001	0001	0000h	R_8

Лема 3. Для чисел вигляду

$$U_N = V_N = \sum_{k=0}^{N-1} ((2^n - 1) \cdot 2^{nk}),$$

де $n = 2^m$, $m = \overline{3,9}$, N – ціле додатне число, результатом виконання операції додавання $U_N + V_N$ буде число вигляду

$$R_N = (2^n - 2) + \sum_{k=1}^{N-1} ((2^n - 1) \cdot 2^{nk}).$$

Знак переносу в найстарший розряд не враховується.

Доведення. У виразі $U_N + V_N = \sum_{k=0}^{N-1} ((2^n - 1) \cdot 2^{nk}) + \sum_{k=0}^{N-1} ((2^n - 1) \cdot 2^{nk})$ отримуємо, що $U_N + V_N = \sum_{k=1}^{N-1} ((2^n + (2^n - 2)) \cdot 2^{nk}) + (2^n + (2^n - 2))$. Так як 2^n є знаком переносу у старший розряд, та, враховуючи, що розряд з номером індексу $k = N$ не розглядається, отримуємо, що $\sum_{k=1}^{N-1} ((1 + (2^n - 2)) \cdot 2^{nk}) + (2^n - 2) = \sum_{k=1}^{N-1} ((2^n - 1) \cdot 2^{nk}) + (2^n - 2)$. Лема доведена.

Генерація даних. Приклад 3. $N=8$, $n=16$. Генерація чисел для тестування операції додавання.

	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFFh	U_8
+	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFFh	V_8
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFEh[2]	R_8

Лема 4. Для чисел вигляду

$$U_N = V_N = \sum_{k=0}^{N-1} ((2^n - 2) \cdot 2^{nk}),$$

де $n = 2^m$, $m = \overline{3,9}$, N – ціле додатне число, результатом виконання операції додавання $U_N + V_N$ буде число вигляду

$$R_N = (2^n - 4) + \sum_{k=1}^{N-1} ((2^n - 3) \cdot 2^{nk}).$$

Знак переносу в найстарший розряд не враховується.

[1] У 16-річній системі обчислення число FFFFh відповідає числу $15 \cdot 4096 + 15 \cdot 256 + 15 \cdot 16 + 15 = 65535$ у десятковій системі обчислення.

[2] У 16-річній системі обчислення число FFFEh відповідає числу $15 \cdot 4096 + 15 \cdot 256 + 15 \cdot 16 + 14 = 65534$ у десятковій системі обчислення.

Доведення. У виразі

$$U_N + V_N = \sum_{k=0}^{N-1}((2^n-2)\cdot 2^{nk}) + \sum_{k=0}^{N-1}((2^n-2)\cdot 2^{nk})$$

отримуємо, що $U_N + V_N = \sum_{k=1}^{N-1}((2^n+(2^n-4))\cdot 2^{nk}) + (2^n+(2^n-4))$.

Так як 2^n – знак переносу у старший розряд, та, враховуючи, що розряд з номером індексу $k = N$ не розглядається, отримуємо, що

$$\sum_{k=1}^{N-1}((1+(2^n-4))\cdot 2^{nk}) + (2^n-4) = \sum_{k=1}^{N-1}((2^n-3)\cdot 2^{nk}) + (2^n-4).$$

Лема доведена.

Генерація даних. Приклад 4. $N=8$, $n=16$. Генерація чисел для тестування операції додавання.

+	FFFE	FFFE	FFFE	FFFE	FFFE	FFFE	FFFE	FFFEh[3]	U_8
	FFFE	FFFE	FFFE	FFFE	FFFE	FFFE	FFFE	FFFEh	V_8
	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD[4]	FFFCh[5]	R_8

Операція віднімання. Згенеруємо багаторозрядні цілі додатні числа для перевірки правильності обчислення операції віднімання.

Лема 5. Для чисел вигляду

$$U_N = \sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk}),\ V_N = \sum_{k=0}^{N-1}2^{nk},$$

де $n=2^m$, $m=\overline{3,9}$, N – ціле додатне число, результатом виконання операції віднімання $(U_N - V_N)$ буде число вигляду

$$R_N = \sum_{k=0}^{N-1}((2^n-2)\cdot 2^{nk}).$$

Доведення. Отримуємо $\sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk}) - \sum_{k=0}^{N-1}2^{nk} = \sum_{k=0}^{N-1}((2^n-1-1)\cdot 2^{nk})$. Лема доведена.

Генерація даних. Приклад 5. $N=8$, $n=16$. Генерація чисел для тестування операції віднімання.

−	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFFh	U_8
	0001	0001	0001	0001	0001	0001	0001	0001h	V_8
	FFFE	FFFE	FFFE	FFFE	FFFE	FFFE	FFFE	FFFEh	R_8

[3] У 16-річній системі обчислення число FFFEh відповідає числу $15\cdot 4096+15\cdot 256+15\cdot 16+14=65534$ у десятковій системі обчислення.

[4] У 16-річній системі обчислення число FFFDh відповідає числу $15\cdot 4096+15\cdot 256+15\cdot 16+13=65533$ у десятковій системі обчислення.

[5] У 16-річній системі обчислення число FFFCh відповідає числу $15\cdot 4096+15\cdot 256+15\cdot 16+12=65532$ у десятковій системі обчислення.

Лема 6. Для чисел вигляду

$$U_N = \sum_{k=0}^{N-1} 2^{nk},\ V_N = \sum_{k=0}^{N-1} ((2^n - 1)\cdot 2^{nk}),$$

де $n = 2^m$, $m = \overline{3,9}$, N – ціле додатне число, результатом виконання операції віднімання $(U_N - V_N)$ буде число вигляду

$$R_N = 2 + \sum_{k=1}^{N-1} 2^{nk}.$$

Знак переносу з найстаршого розряду не враховується.

Доведення. Так як знак переносу з найстаршого розряду не враховується, то можна вважати, що розряд з індексом $k = N+1$ дорівнює 1. Враховуючи, що

$$U_{N+1} = (2^n + 1) + \sum_{k=1}^{N-1} (2^n \cdot 2^{nk}),$$

результат виразу

$$(U_{N+1} - V_N) \text{ буде } \sum_{k=1}^{N-1} ((2^n - (2^n - 1))\cdot 2^{nk}) + (2^n + 1 - (2^n - 1)) = 2 + \sum_{k=1}^{N-1} 2^{nk}.$$

Лема доведена.

Генерація даних. Приклад 6. $N = 8$, $n = 16$. Генерація чисел для тестування операції віднімання.

	0001	0001	0001	0001	0001	0001	0001	0001h	U_8
−	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFFh	V_8
	0001	0001	0001	0001	0001	0001	0001	0002h	R_8

Для тестування операції віднімання можна також використовувати зворотні операції з лем 1–4 та, навпаки, для тестування операції додавання використовувати числа з лем 5 та 6.

З точки зору теорії тестування операції додавання та віднімання, в яких одним з аргументів або результатом є нуль, є важливими для тестування граничних випадків, але, виходячи з їх тривіальності обчислення та тестування, у даній роботі не розглядаються.

Операція множення. Згенеруємо багаторозрядні цілі додатні числа для перевірки правильності обчислення операції множення.

Лема 7. Для чисел вигляду

$$U_N = V_N = \sum_{k=0}^{N-1} 2^{nk},$$

де $n = 2^m$, $m = \overline{3,9}$, $N < 2^n$, N – ціле додатне число, результатом виконання операції множення $U_N \cdot V_N$ буде число вигляду

$$R_{2N} = \sum_{k=N}^{2N-2} ((2N - 1 - k)\cdot 2^{nk}) + \sum_{k=0}^{N-1} ((k+1)\cdot 2^{nk}).$$

Доведення. Схема обчислення методу множення у стовпчик $(1\cdot 1)$.

				$2^{(N-1)n}$	…	2^{1n}	2^{0n}
			2^{Nn}	$2^{(N-1)n}$	…	2^{1n}	
…	…	…	…	…	…	…	…
	$1\cdot 2^{(2N-2)n}$	…	2^{Nn}	$2^{(N-1)n}$			
$0\cdot 2^{(2N-1)n}$	$1\cdot 2^{(2N-2)n}$	…	$(N-1)\cdot 2^{Nn}$	$N\cdot 2^{(N-1)n}$	…	$2\cdot 2^{1n}$	$1\cdot 2^{0n}$

Тоді добуток чисел $U_N\cdot V_N$ може бути записаний наступним чином: $\sum_{k=0}^{N-1}(N-1-k)2^{n(k+N)}+\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk})$. Якщо замінити діапазон індексів $k=\overline{0,N-1}$ у першому доданку на діапазон $k=\overline{N-1,2N-1}$, та, враховуючи, що значення розряду з індексом $k=2N-1$ дорівнює нулю, отримуємо $R_N=\sum_{k=N}^{2N-2}((2N-1-k)\cdot 2^{nk})+\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk})$. Лема доведена.

Генерація даних. Приклад 7. $N=4$, $n=16$. Генерація чисел для тестування операції множення $(1\cdot 1)$.

				×	0001	0001	0001	0001h U_4
					0001	0001	0001	0001h V_4
0000	0001	0002	0003	0004	0003	0002	0001h	R_8

Лема 8. Для чисел вигляду

$$U_N=\sum_{k=0}^{N-1}2^{nk},\ V_N=\sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk}),$$

де $n=2^m$, $m=\overline{3,9}$, $N<2^n$, N – ціле додатне число, результатом виконання операції множення $U_N\cdot V_N$ буде число вигляду

$$R_{2N}=\sum_{k=N+1}^{2N-1}2^{nk}+\sum_{k=1}^{N-1}((2^n-2)\cdot 2^{nk})+(2^n-1).$$

Доведення. Схема обчислення методу множення у стовпчик $((2^n-1)\cdot 1)$.

$2^{(2N-1)n}$	$2^{(2N-2)n}$	...		2^{Nn}	$2^{(N-1)n}$	...	2^{1n}	2^{0n}
		...			2^n-1	...	2^n-1	2^n-1
		...		2^n-1	2^n-1	...	2^n-1	
...	...	...		...	...	...	...	...
	2^n-1	...		2^n-1	2^n-1			
$0\cdot(2^n-1)$	2^n-1	...	$(N-1)\cdot 2^n-(N-1)$		$N\cdot 2^n-N$	...	$2\cdot 2^n-2$	2^n-1

Беремо до уваги, що для розміщення числа $2^n - 1$ треба n бітів. Останній рядок можна переписати з урахуванням значень знаків переносів, які позначимо у дужках як (+ 1), (+ 2), (+ 3) і т. д. Зрозуміло, що $2 \cdot (2^n - 1)$ генерує один знак переносу у $n + 1$ біт, $3 \cdot (2^n - 1)$ генерує два знаки переносу, $4 \cdot (2^n - 1)$ генерує три знаки переносу і т. д.

1-е врахування знаків переносів

$2^{(2N-1)n}$	$2^{(2N-2)n}$	...	2^{Nn}	$2^{(N-1)n}$	...	2^{1n}	2^{0n}
	(+ 1)	...	(+ (N – 1))	(+ (N – 2))	...	(+ 0)	
$0 \cdot (2^n - 1)$	$2^n - 1$	...	$2^n - (N-1)$	$2^n - N$	...	$2^n - 2$	$2^n - 1$

З урахуванням переносів, виникають нові переноси. Остаточно маємо.

2-е врахування знаків переносів.

$2^{(2N-1)n}$	$2^{(2N-2)n}$	...	2^{Nn}	$2^{(N-1)n}$	...	2^{1n}	2^{0n}
1	1	...	0	$2^n - 2$	...	$2^n - 2$	$2^n - 1$

Тоді вираз приймає вигляд

$$R_{2N} = \sum_{k=N+1}^{2N-1} 2^{nk} + 0 \cdot 2^{Nn} + \sum_{k=1}^{N-1} ((2^n - 2) \cdot 2^{nk}) + (2^n - 1).$$

Якщо розглядати загальний випадок та не враховувати другий нульовий доданок, то отримаємо бажаний вираз. Що і потрібно було довести.

Генерація даних. Приклад 8. Генерація чисел для тестування операції множення $(1 \cdot (2^n - 1))$, $N = 4$, $n = 16$.

				×	0001 FFFF	0001 FFFF	0001 FFFF	0001h FFFFh	U_4 V_4
0001	0001	0001	0000	FFFE	FFFE	FFFE	FFFFh	R_8

Генерація даних. Приклад 9. Генерація чисел для тестування операції множення $(1 \cdot (2^n - 1))$, $N = 8$, $n = 16$.

								×	01 FF	01 FF	01 FF	01 FF	01 FF	01 FF	01 FF	01h FFh	U_8 V_8
01	01	01	01	01	01	01	00	FE	FE	FE	FE	FE	FE	FE	FFh	R_{16}

Лема 9. Для чисел вигляду

$$U_N = V_N = \sum_{k=0}^{N-1} ((2^n - 2) \cdot 2^{nk}),$$

де $n = 2^m$, $m = \overline{3,9}$, $N < 2^n$, N – ціле додатне число, результатом виконання операції множення $U_N \cdot V_N$ буде число вигляду

$$R_{2N} = (2^n - 3) \cdot 2^{(2N-1)n} + (2^n - 1) \cdot 2^{(2N-2)n} + \sum_{k=N+1}^{2N-4} ((2N - 3 - k) \cdot 2^{nk}) + (N - 5) \cdot 2^{Nn} +$$

$$+ \sum_{k=1}^{N-1} ((k + 3) \cdot 2^{nk}) + 4. \qquad (1)$$

Доведення. Схема обчислення методу множення у стовпчик $((2^n-2)\cdot(2^n-2))$.

$2^{(2N-1)n}$	$2^{(2N-2)n}$	$2^{(2N-3)n}$	...	2^{Nn}	$2^{(N-1)n}$	$2^{(N-2)n}$	...	2^{1n}	2^{0n}
			...		$2^{2n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n$	...	$2^{2n}-4\cdot 2^n$	$2^{2n}-$
					$+4$	$+4$		$+4$	$4\cdot 2^n+4$
			...	$2^{2\,n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n+$	...	$2^{2n}-4\cdot 2^n$	
				$+4$	$+4$	4		$+4$	
...	...	...	...	...	...	...	...	...	...
		$2^{2n}-4\cdot 2^n$	...	$2^{2n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n$	...		
		$+4$		$+4$	$+4$	$+4$			
	$2^{2n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n$	...	$2^{2n}-4\cdot 2^n$	$2^{2n}-4\cdot 2^n$		...		
	$+4$	$+4$		$+4$	$+4$				

Доданки вигляду $-4\cdot 2^n$ перенесемо у старші розряди та позначимо (– 4), доданки вигляду 2^{2n} перенесемо у старші через один розряд та позначимо (+ 1) та використаємо додаткові рядки.

Врахування знаків переносу.

$2^{(2N-1)n}$	$2^{(2N-2)n}$		$2^{(N+1)n}$	2^{Nn}	$2^{(N-1)n}$	...	2^{2n}	2^{1n}	2^{0n}
		...	(+ 1)	(+ 1)	(+ 1)	...	**(+ 1)**		
...	...	...	...	...	...	...	...	...	...
	(+ 1)	...	(+ 1)	(+ 1)	(+ 1)	...			
(+ 1)	(+ 1)	...	(+ 1)	(+ 1)		...			
(+ 1)	(+ 1)	...	(+ 1)			...			
2^n		...				...			
		...		(– 4)	(– 4)	...	(– 4)	**(– 4)**	
...		...	(– 4)	(– 4)	(– 4)	...	(– 4)		
	...	...	...	...	...	...	...	...	...
(– 4)	(– 4)	...	(– 4)	(– 4)	(– 4)	...			
	(– 4)	...	(– 4)	(– 4)		...			
		...			+ 4	...	+ 4	+ 4	**+ 4**
		...		+ 4	+ 4	...	+ 4	+ 4	
...	...	...	...	...	...	...	...	...	...
		...	+ 4	+ 4	+ 4	...			
	+ 4	...	+ 4	+ 4	+ 4	...			

Вирази для обчислення доданків окремо по (+1), (–4), +4 будуть наступними:

$$R_{2N}^{(+1)} = \sum_{k=N+1}^{2N-1}((2N+1-k)\cdot 2^{nk}) + \sum_{k=2}^{N}((k-1)\cdot 2^{nk}),$$

$$R_{2N}^{(-4)} = -4\sum_{k=N+1}^{2N-1}((2N-k)\cdot 2^{nk}) - 4\sum_{k=1}^{N}(k\cdot 2^{nk}),$$

$$R_{2N}^{+4} = 4\sum_{k=N}^{2N-2}((2N-1-k)\cdot 2^{nk}) + 4\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk}),$$

$$R_{2N} = 2^{2Nn} + R_{2N}^{(+1)} + R_{2N}^{(-4)} + R_{2N}^{+4}.$$

Загалом отримуємо

$$R_{2N} = 2^{2Nn} + (\sum_{k=N+1}^{2N-1}((2N+1-k)\cdot 2^{nk}) + \sum_{k=2}^{N}((k-1)\cdot 2^{nk})) -$$

$$-4\sum_{k=N+1}^{2N-1}((2N-k)\cdot 2^{nk}) - 4\sum_{k=1}^{N}(k\cdot 2^{nk}) + 4\sum_{k=N}^{2N-2}((2N-1-k)\cdot 2^{nk}) + 4\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk}).$$

Після цього згрупуємо значення по k:

$$R_{2N} = 2^{2Nn} + \sum_{k=N+1}^{2N-1}((2N+1-k)\cdot 2^{nk}) - 4\sum_{k=N+1}^{2N-1}((2N-k)\cdot 2^{nk}) + 4\sum_{k=N}^{2N-2}((2N-1-k)\cdot 2^{nk}) +$$

$$+\sum_{k=2}^{N}((k-1)\cdot 2^{nk}) - 4\sum_{k=1}^{N}(k\cdot 2^{nk}) + 4\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk}).$$

Після додаткового групування доданків отримуємо

$$R_{2N} = 2^{n}\cdot 2^{(2N-1)n} + ((2N+1-(2N-1)) - 4(2N-(2N-1)))\cdot 2^{(2N-1)n} +$$

$$+\sum_{k=N+1}^{2N-2}(((2N+1-k) - 4(2N-k) + 4(2N-1-k))\cdot 2^{nk}) + (8N-4-4N+(N-1)-4N)\cdot 2^{Nn} +$$

$$+\sum_{k=2}^{N-1}((k-1)\cdot 2^{nk}) - 4\sum_{k=2}^{N-1}(k\cdot 2^{nk}) + 4\sum_{k=2}^{N-1}((k+1)\cdot 2^{nk}) - 4\sum_{k=1}^{1}(k\cdot 2^{nk}) + 4\sum_{k=0}^{1}((k+1)\cdot 2^{nk}).$$

Продовжуючи розкриття дужок, отримуємо

$$R_{2N} = (2^{n}-2)\cdot 2^{(2N-1)n} + \sum_{k=N+1}^{2N-2}((2N-3-k)\cdot 2^{nk}) + (N-5)\cdot 2^{Nn} +$$

$$+\sum_{k=2}^{N-1}((k-1-4k+4k+4)\cdot 2^{nk}) + (-4k+4k+4)\cdot 2^{n} + 4\cdot 2^{n0}.$$

Після скорочень отримуємо

$$R_{2N} = (2^{n}-2)\cdot 2^{(2N-1)n} + \sum_{k=N+1}^{2N-2}((2N-3-k)\cdot 2^{nk}) + (N-5)\cdot 2^{Nn} + \sum_{k=2}^{N-1}((k+3)\cdot 2^{nk}) + 4\cdot 2^{n} + 4.$$

Якщо врахувати, що при $k = 2N-2$ завжди буде $-2^{(2N-2)n}$ і це потребує знак переносу зі старшого розряду $2N-1$, а при $k = 2N-3$ буде завжди нуль, а 5-й доданок можна приєднати до 4-го доданку, то остаточно отримуємо (1). Лема доведена.

Генерація даних. Приклад 10. При $N = 4$ вираз буде мати вигляд, враховуючи, що розряд з індексом $k = 2N-3 = 5$ буде мати значення -1, що потребує перенос знаку зі старшого розряду з індексом $k = 2N-2 = 6$.

Значення розрядів при $N = 4$.

2^{7n}	2^{6n}	2^{5n}	2^{4n}	2^{3n}	2^{2n}	2^{1n}	2^{0n}
2^n-3	2^n-2	2^n-1	2^n-1	6	5	4	4

Значення розрядів при $N = 8$.

2^{15n}	2^{14n}	2^{13n}	2^{12n}	2^{11n}	2^{10n}	2^{9n}	2^{8n}	2^{7n}	2^{6n}	2^{5n}	2^{4n}	2^{3n}	2^{2n}	2^{1n}	2^{0n}
2^n-3	2^n-1	0	1	2	3	4	3	10	9	8	7	6	5	4	4

Генерація даних. Приклад 11. Генерація чисел для тестування операції множення $((2^n-2)\cdot(2^n-2))$, $N=4$, $n=16$.

				×	FFFE FFFE	FFFE FFFE	FFFE FFFE	FFFEh FFFEh	U_4 V_4
FFFD	FFFE	FFFF	FFFF	0006	0005	0004	0004h		R_8

Генерація чисел для тестування операції множення $((2^n-2)\cdot(2^n-2))$, $N=8,$, $n=16$.

								×	FE FE	FE FE	FE FE	FE FE	FE FE	FE FE	FE FE	FEh[6] FEh	U_8 V_8
FD	FF	00	01	02	03	04	03	0A	09	08	07	06	05	04	04h	R_{16}

Лема 10. Для чисел вигляду

$$U_N = V_N = \sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk}),$$

де $n=2^m$, $m=\overline{3,9}$, $N<2^n$, N – ціле додатне число, результатом виконання операції множення $U_N \cdot V_N$ буде число вигляду

$$R_{2N} = \sum_{k=N+1}^{2N-1}((2^n-1)\cdot 2^{nk}) + (2^n-2)\cdot 2^{Nn} + 1. \qquad (2)$$

Доведення. Числа вигляду $U_N = V_N = \sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk})$ можуть бути представлені у вигляді $U_N = V_N = 2^{Nn}-1$. Тоді операція множення чисел $U_N \cdot V_N$ має вигляд

$$U_N \cdot V_N = (2^{Nn}-1)\cdot(2^{Nn}-1) = 2^{2Nn} - 2\cdot 2^{Nn} + 1.$$

2^{2Nn} означає, що для такого числа всі розряди будуть нульовими окрім найстаршого. При відніманні числа $2\cdot 2^{Nn}$ від 2^{2Nn} у всіх старших розрядах, починаючи з $N+1$, треба робити знак переносу у молодші розряди, що змінює нулі на 2^n-1 окрім розряду N, з якого віднімається 2 згідно другого додатку $-2\cdot 2^{Nn}$. Тоді остаточна формула приймає вигляд (2). Лема доведена.

[6] У 16-річній системі обчислення число FEh відповідає числу $254 = 15\cdot 16 + 14$ у десятковій системі обчислення.

Генерація даних. Приклад 12. Генерація чисел для тестування операції множення $((2^n-1)\cdot(2^n-1)),\ N=4,\ n=16$.

				×	FFFF	FFFF	FFFF	FFFFh	U_4
					FFFF	FFFF	FFFF	FFFFh	V_4
FFFF	FFFF	FFFF	FFFE	0000	0000	0000	0001h		R_8

Генерація чисел для тестування операції множення $((2^n-1)\cdot(2^n-1)),\ N=8,\ n=16$

								×	FF	FF	FF	FF	FF	FF	FF	FFh	U_8
									FF	FF	FF	FF	FF	FF	FF	FFh	V_8
FF	FF	FF	FF	FF	FF	FF	FE	00	00	00	00	00	00	00	01h		R_{16}

Операція множення за модулем. Згенеруємо багаторозрядні цілі додатні числа для перевірки правильності обчислення операції багаторозрядного множення за модулем.

Лема 11. Для чисел вигляду

$$M_N=\sum_{k=1}^{N-1}((2^n-1)\cdot 2^{nk})+(2^n-2),\ U_N=\sum_{k=0}^{N-1}2^{nk},\ V_N=\sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk}),$$

де $n=2^m$, $m=\overline{3,9}$, $N<2^n$, N – ціле додатне число, результатом виконання операції множення за модулем $(U_N\cdot V_N)\bmod M_N$ буде число вигляду $R_N=\sum_{k=0}^{N-1}2^{nk}$.

Доведення. Операцію множення за модулем $(U_N\cdot V_N)\bmod M_N$ представимо у наступному вигляді: $(U_N \bmod M_N\cdot V_N \bmod M_N)\bmod M_N$. З урахуванням того, що $V_N \bmod M_N=1$, так як $V_N=M_N+1$ отримуємо $(U_N \bmod M_N\cdot 1)\bmod M_N=U_N \bmod M_N$. Так як $U_N<M_N$, то знак модуля можна опустити $U_N \bmod M_N=U_N=\sum_{k=0}^{N-1}2^{nk}$. Лема доведена.

Генерація даних. Приклад 13. Генерація чисел для тестування операції множення за модулем, $N=4,\ n=16$.

	×	0001	0001	0001	0001h	U_4
		FFFF	FFFF	FFFF	FFFFh	V_4
mod		FFFF	FFFF	FFFF	FFFEh	M_4
		0001	0001	0001	0001h	R_4

Генерація чисел для тестування операції множення за модулем, $N=8,\ N=8$

	×	01	01	01	01	01	01	01	01h	U_8
		FF	FF	FF	FF	FF	FF	FF	FFh	V_8
mod		FF	FF	FF	FF	FF	FF	FF	FEh	M_8
		01	01	01	01	01	01	01	01h	R_8

Лема 12. Для чисел вигляду

$$U_N = \sum_{k=0}^{N-1} 2^{nk},\ U_N = V_N = \sum_{k=0}^{N-1} ((2^n - 1)\cdot 2^{nk}),\ M_N = \sum_{k=1}^{N-1} 2^{nk},$$

де $n = 2^m$, $m = \overline{3,9}$, $N < 2^n$, N – ціле додатне число, результатом виконання операції множення за модулем $(U_N \cdot V_N) \bmod M_N$ буде число вигляду $R_N = 2^n - 1$.

Доведення. Операцію множення за модулем $(U_N \cdot V_N) \bmod M_N$ представимо у наступному вигляді: $(U_N \bmod M_N \cdot V_N \bmod M_N) \bmod M_N$. З урахуванням того, що $U_N \bmod M_N = 1$, так як $U_N = M_N + 1$, отримуємо $(V_N \bmod M_N \cdot 1) \bmod M_N = V_N \bmod M_N$. Так як $V_N = (M_N + 1)\cdot(2^n - 1)$, або $V_N = (2^n - 1)\cdot M_N + (2^n - 1)$, то $V_N \bmod M_N = 2^n - 1$. Лема доведена.

Генерація даних. Приклад 14. Генерація чисел для тестування операції множення за модулем, $N = 4$, $n = 16$.

×	0001	0001	0001	0001h	U_4
	FFFF	FFFF	FFFF	FFFFh	V_4
mod	0001	0001	0001	0000h	M_4
	0000	0000	0000	FFFFh	R_4

Генерація чисел для тестування операції множення за модулем, $N = 8$, $N = 8$.

×	01	01	01	01	01	01	01	01h	U_8
	FF	FF	FF	FF	FF	FF	FF	FFh	V_8
mod	01	01	01	01	01	01	01	00h	M_8
	00	00	00	00	00	00	00	FFh	R_8

Лема 13. Для чисел вигляду

$$U_N = V_N = \sum_{k=0}^{N-1} ((2^n - 1)\cdot 2^{nk}),\ M_N = \sum_{k=1}^{N-1} ((2^n - 1)\cdot 2^{nk}) + (2^n - 2),$$

де $n = 2^m$, $m = \overline{3,9}$, $N < 2^n$, N – ціле додатне число, результатом виконання операції множення за модулем $(U_N \cdot V_N) \bmod M_N$ буде число вигляду $R_N = 1$.

Доведення. Операцію множення за модулем $(U_N \cdot V_N) \bmod M_N$ представимо у наступному вигляді: $(U_N \bmod M_N \cdot V_N \bmod M_N) \bmod M_N$. З урахуванням того, що $U_N \bmod M_N = V_N \bmod M_N = 1$, та $U_N = V_N = M_N + 1$, отримуємо $(1\cdot 1) \bmod M_N = 1$. Лема доведена.

Генерація даних. Приклад 15. Генерація чисел для тестування операції множення за модулем, $N = 4$, $n = 16$.

×	FFFF	FFFF	FFFF	FFFFh	U_4
	FFFF	FFFF	FFFF	FFFFh	V_4
mod	FFFF	FFFF	FFFF	FFFEh	M_4
	0000	0000	0000	0001h	R_4

Генерація чисел для тестування операції множення за модулем, $N=8,\ N=8.$

$\times$	FF	FF	FF	FF	FF	FF	FF	FFh	U_8
mod	FF	FF	FF	FF	FF	FF	FF	FFh	V_8
	FF	FF	FF	FF	FF	FF	FF	FEh	M_8
	00	00	00	00	00	00	00	01h	R_8

Операція піднесення до степеня за модулем. Згенеруємо багаторозрядні цілі додатні числа для перевірки правильності обчислення операції піднесення до степеня за модулем.

Лема 14. Для чисел вигляду

$$U_N=\sum_{k=0}^{N-1}2^{nk},\ E_N=M_N=\sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk}),$$

де $n=2^m,\ m=\overline{3,9},\ N<2^n,\ N=2^p,\ \ p=2^f$ – ціле додатне число, результатом виконання операції піднесення до степеня за модулем $U_N^{E_N}\bmod M_N$ буде число вигляду $R_N=P\sum_{k=0}^{N-1}2^{nk},\ \ P=2^t,$ $t=(n-2p)\bmod n.$

Доведення. У першій частині доведення розглянемо тільки множення за модулем. Добутком чисел вигляду

$$U_N=\sum_{k=0}^{N-1}2^{nk}\text{, де } n=2^m,\ m=\overline{3,9},\ N<2^n,$$

N – ціле додатне число, буде число вигляду

$$U_N\cdot U_N=R_{2N}=\sum_{k=N}^{2N-2}((2N-1-k)\cdot 2^{nk})+\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk})$$

згідно леми 7. Добуток чисел $R_{2N}=U_N\cdot U_N$ можна відобразити наступним чином:

			$\times$	0001	0001	…	0001h	U_N
				0001	0001	…	0001h	U_N
0000	0001	…	$N-1$	N	$N-1$	…	0001h	R_{2N}

Число $M_N=\sum_{k=0}^{N-1}((2^n-1)\cdot 2^{nk})$ можна представити як $M_N=2^{Nn}-1$. Якщо у першому доданку числа

$$R_{2N}=\sum_{k=N}^{2N-2}((2N-1-k)\cdot 2^{nk})+\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk})$$

додавання починати з індексу нуль та винести множник за дужки, то отримаємо, що

$$R_{2N}=2^{Nn}\sum_{k=0}^{N-2}(N-1-k)\cdot 2^{nk}+\sum_{k=0}^{N-1}((k+1)\cdot 2^{nk}).$$

З урахуванням того, що $2^{Nn}\bmod M_N=(M_N+1)\bmod M_N=1$, а доданки

$$\sum_{k=0}^{N-2}(N-1-k)\ \text{ та }\ \sum_{k=0}^{N-1}((k+1)\cdot 2^{nk})$$

менше 2^{Nn}, то отримуємо, що добуток чисел $R_{2N}=U_N\cdot U_N$ за модулем M_N буде наступним:

$$R_{2N} \bmod M_N = (2^{Nn} \sum_{k=0}^{N-2} (N-1-k) \cdot 2^{nk} + \sum_{k=0}^{N-1} ((k+1) \cdot 2^{nk})) \bmod (2^{Nn} - 1) =$$

$$= (\sum_{k=0}^{N-1} ((N-1-k) + (k+1)) \cdot 2^{nk}) \bmod (2^{Nn} - 1) = (N \sum_{k=0}^{N-1} 2^{nk}) \bmod M_N = (N \cdot U_N) \bmod M_N .$$

Враховуючи, що $U_N^2 \bmod M_N = (N \cdot U_N) \bmod M_N$, отримуємо

$$U_N^{E_N} \bmod M_N = (N^{E_N - 1} \cdot U_N) \bmod M_N = N^{E_N + 1} \bmod M_N \cdot (N^{-2} \cdot U_N) \bmod M_N . \quad (3)$$

У другій частині доведемо, що $N^{E_N+1} \bmod M_N = 1$, тобто число N^{E_N+1} можна представити як степінь числа $(M_N + 1)^x$, де x – натуральне число. Число N^{E_N+1} представимо у наступному вигляді: $N^{E_N+1} = (2^p)^{E_N+1} = (2^p)^{2^{Nn}} = 2^{p2^{Nn}}$. Цей вираз можна записати наступним чином:

$$2^{p2^{Nn}} = 2^{Nn\frac{p2^{Nn}}{Nn}} = (2^{Nn})^{\frac{p2^{Nn}}{Nn}} = (2^{Nn})^{\frac{p2^{Nn}}{2^{m+p}}} = (2^{Nn})^{p2^{Nn-(m+p)}} = (M_N + 1)^{p2^{Nn-(m+p)}} .$$

При обчисленні за модулем $M_N + 1$ отримуємо, що

$$N^{E_N+1} \bmod M_N = (M_N + 1)^{p2^{Nn-(m+p)}} \bmod M_N = 1 . \quad (4)$$

При використанні (4) у (3) отримуємо, що $U_N^{E_N} \bmod M_N = (N^{-2} \cdot U_N) \bmod M_N$.

У третій частині доведення розглянемо циклічні зсуви. Помножимо число U_N на 2^n за модулем M_N. Так як число має вигляд $U_N = \sum_{k=0}^{N-1} 2^{nk}$, то отримуємо, що

$$(2^n \cdot U_N) \bmod M_N = (2^n \cdot \sum_{k=0}^{N-1} 2^{nk}) \bmod M_N = (\sum_{k=0}^{N-1} 2^{n+nk}) \bmod M_N =$$

$$= (2^{nN} + \sum_{k=1}^{N-1} 2^{nk}) \bmod M_N = 2^{nN} \bmod M_N + (\sum_{k=1}^{N-1} 2^{nk}) \bmod M_N =$$

$$= (M_N + 1) \bmod M_N + (\sum_{k=1}^{N-1} 2^{nk}) \bmod M_N = (1 + \sum_{k=1}^{N-1} 2^{nk}) \bmod M_N =$$

$$= (\sum_{k=0}^{0} 2^{nk} + \sum_{k=1}^{N-1} 2^{nk}) \bmod M_N = (\sum_{k=0}^{N-1} 2^{nk}) \bmod M_N = U_N \bmod M_N .$$

Отримуємо, що $(2^n \cdot U_N) \bmod M_N = U_N \bmod M_N$. Множення числа, яке є степеню двійки, на число U_N за модулем M_N можна представити наступним чином:

$$(2^g \cdot U_N) \bmod M_N = (2^{g-n} \cdot 2^n \cdot U_N) \bmod M_N = (2^{g-n} \cdot ((2^n \cdot U_N) \bmod M_N) \bmod M_N ,$$

$$(2^{g-n} \cdot U_N) \bmod M_N = (2^{g \bmod n} \cdot U_N) \bmod M_N = (2^q \cdot U_N) \bmod M_N ,$$

де $q = g \bmod n$.

Число U_N складається з N розрядів, довжина кожного з яких є n бітів. Якщо всі розряди мають однакове число 2^{nk}, а саме степінь двійки, то можна розглядати, що при множенні числа U_N за модулем M_N на 2^g, відбуваються зсуви вліво на g бітів, якщо $g < n$, або на $g \bmod n$ бітів, якщо $g \geq n$, з циклічністю у n бітів. Тобто зсуви відбуваються паралельно у кожному з N розрядів окремо без переносу значень у сусідні розряди. Після множення $2^x \cdot 2^{nk}$ у кожному розряді буде число $2^{(x+nk) \bmod n} = 2^{x \bmod n}$.

Для того, щоб зсуви відбувалися вправо, необхідно виконати обернену операцію, тобто операцію множення на двійку з від'ємним степенем

$$(2^{(-1) \bmod n} \cdot U_N) \bmod M_N = (2^{n-1} \cdot \sum_{k=0}^{N-1} 2^{nk}) \bmod M_N = (2^{n-1} \cdot U_N) \bmod M_N .$$

З урахуванням того, що зсуви вправо відбуваються циклічно, напишемо вираз у вигляді

$$(N^{-2} \cdot U_N) \bmod M_N = ((2^p)^{-2} \cdot U_N) \bmod M_N = (2^{-2p} \cdot U_N) \bmod M_N = (2^{(n-2p) \bmod n} \cdot U_N) \bmod M_N .$$

Остаточно можна записати, що $U_N^{E_N} \bmod M_N = 2^{(n-2p) \bmod n} U_N$, де $E_N = M_N = 2^{Nn} - 1$, $U_N = \sum_{k=0}^{N-1} 2^{nk}$, $N = 2^p$, $p = 2^f$, $n = 2^m$, $m = \overline{3,9}$, $N < 2^n$. Лема доведена.

Генерація даних. Приклад 16. Генерація чисел для тестування операції піднесення до степеня за модулем, $N = 4$, $n = 16$.

	0001	0001	0001	0001h	U_4
exp	FFFF	FFFF	FFFF	FFFFh	E_4
mod	FFFF	FFFF	FFFF	FFFFh	M_4
	1000	1000	1000	1000h	R_4

Висновки. Надані залежності між вхідними та вихідними даними, за допомогою яких можна перевіряти результати виконання багаторозрядної операції. Найцікавішим у наданих залежностях є те, що у силу своєї простоти є можливість візуальної перевірки без використання спеціальних програмних засобів, що є дуже важливим критерієм при розробці та налагодженні програми. Показано, що алгоритм генерування вхідних та вихідних даних – простий. Залежності надані у вигляді лем. Для кожної леми наведені приклади для зручності сприйняття залежності між вхідними та вихідними даними розглянутих операцій.

Використання таких залежностей також дозволяє перевіряти цілісність вихідних даних при делегуванні обчислень у розподілені системи, наприклад, такі, як хмарні обчислення [10].

Список літератури

1. Задірака В., Олексюк О. Комп'ютерна арифметика багаторозрядних чисел: Наукове видання. Київ. 2003. 263 с.
2. Карацуба А.А., Офман Ю.П. Умножение многоразрядных чисел на автоматах. *ДАН СССР*. 1962. **145** (2). С. 293–294. http://mi.mathnet.ru/dan26729
3. Николайчук Я.М., Возна Н.Я., Пітух І.Р. Проектування спеціалізованих комп'ютерних систем. Тернопіль: Терно-Граф, 2010. 392 с. http://library.kpi.kharkov.ua/uk/inftechnologies_nik

4. Хіміч О.М. Суперкомп'ютерні технології та математичне моделювання складних систем. *Вісник НАН України*. 2018. **5**. С. 69−72. http://dspace.nbuv.gov.ua/handle/123456789/140740
5. Новокшонов А.К. Контроль цілісності арифметичних обчислень. Матеріали XIV Міжнародної науково-практичної конференції *«Теоретичні та прикладні аспекти побудови програмних систем».* 2017. С. 149–154.
6. Анісімов А.В. Алгоритмічна теорія великих чисел. *Модулярна арифметика великих чисел.* Київ: Видавничий дім "Академперіодика", 2001. 153 с. http://books.zntu.edu.ua/book_info.pl?id=21106
7. Cooley J.W., Tukey J.W. An algorithm for the machine calculation of complex Fourier Series. *Math Compt.* 1965. **19**. P. 257–301. https://doi.org/10.1090/S0025-5718-1965-0178586-1
8. Davis W.F. A class of efficient convolution algorithms. *Applicat. Walsh Functions.* 1972. March. P. 318–329.
9. Montgomery P.L. Modular Multiplication Without Trial Division. *Math. Comp.* 1985. **44**. P. 519–521. https://doi.org/10.1090/S0025-5718-1985-0777282-X
10. Anisimov A.V., Novokshonov A. Verifiable Arithmetic Computations Using Additively Homomorphic Tags. *IEEE Advanced Trends in Information Theory*. 2019. P. 93–96. https://doi.org/10.1109/ATIT49449.2019.9030485

Одержано 12.03.2021

Терещенко Андрій Миколайович,
кандидат фізико-математичних наук, докторант
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
teramidi@ukr.net

Задірака Валерій Костянтинович,
академік НАН України, завідувач відділу
Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
https://orcid.org/0000-0001-9628-0454

MSC 90C15, 49M27

Andrii Tereshchenko *, Valeriy Zadiraka

Generating Big Numbers for Testing Multi-Digit Arithmetic Algorithms

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
** Correspondence: teramidi@ukr.net*

Introduction. The emergence of new parallel computational systems, such as multi-core processors, clusters, distributed systems, is due to the solution of various applied problems in various fields. The difference between devices for which parallel algorithms are implemented causes a variety of existing methods for parallelizing the calculation of multi-digit arithmetic operations. There is a problem of developing universal algorithms for implementing multi-digit arithmetic operations that are efficiently performed on various devices and on various systems.

Very often it is not possible to develop a new algorithm, since at this stage there is still no test data with which it is possible to analyze the result of calculation. Therefore, the task of preparing test data and results is no less important than the development of the algorithm itself. The quality of the prepared data determines the quality of the implemented algorithm and the time required to find and eliminate errors in the algorithm-program and its implementation.

In this paper, some simple dependencies are given, using which you can visually check the correctness of the calculation of multi-digit operations of addition, subtraction, multiplication, multiplication by modulo and exponentiation by modulo. Simple algorithms for generating input and output multi-digit data are presented. Using dependencies allows to check the integrity of the output when delegating computations to distributed systems such as cloud computing.

The purpose of the article is to show simple dependencies between the input data and the results of performing multi-digit operations of addition, subtraction, multiplication, multiplication by modulo and exponentiation by modulo. For the given dependencies, methods for generating input and output multi-digit numbers are shown, which can be used to check the correctness of the calculation of multi-digit operations, which significantly saves the time required for preparing test data.

Dependencies are provided in a generic way, which allows you to generate input data and results for devices that operate on words of different lengths (8, 16, 32, 64, 128, 256, etc. bits).

Results. The dependences between the input data and the results of performing multi-digit operations are analyzed. The provided dependencies are proved in the form of lemmas. The dependencies are presented in a general form, since to generate multi-digit sequences, it is needed to set two parameters: N – the number of digits in the multi-digit value and n – the length of the digits in bits. The examples show the generation of input data and results for various multi-digit operations.

Conclusions. The paper presents dependencies that are easy to remember and use for visual verification of the results of multi-digit calculations without using additional or special software or hardware, which allows to devote the saved time to developing new or more efficient modifications of multi-digit algorithms.

Keywords: multi-digit arithmetic, parallel computational model.