© Gnatyuk S., Kovtun V., Berdnik O., Kovtun M. Approaches to performance increasing of extended Euclidean algorithm for double precision division on single precision large integers // Ukrainian Scientific Journal of Information Security, 2015, vol. 21, issue 1, p. 40-51.

ПОДХОДЫ К ПОВЫШЕНИЮ ПРОИЗВОДИТЕЛЬНОСТИ РАСШИРЕННОГО АЛГОРИТМА ЕВКЛИДА ДЛЯ ДЕЛЕНИЯ БОЛЬШИХ ЧИСЕЛ ДВОЙНОЙ ТОЧНОСТИ НА БОЛЬШИЕ ЧИСЛА ОДИНАРНОЙ ТОЧНОСТИ

Сергей Гнатюк, Владислав Ковтун, Оксана Бердник, Мария Ковтун

Национальный авиационный университет, Украина

ГНАТЮК Сергей Александрович, к.т.н.

Дата и место рождения: 1985 год, г. Нетешин, Хмельницкая область, Украина.
Образование: Национальный авиационный университет, 2007 год.
Должность: доцент кафедры безопасности информационных технологий с 2012 года.
Научные интересы: информационная безопасность, квантовая криптография, защита гражданской авиации от киберугроз, менеджмент инцидентов информационной безопасности.
Публикации: более 100 научных публикаций, среди которых монографии, статьи в международных и отечественных рецензированных специализированных журналах, патенты, авторские свидетельства на программные продукты и т.п.
E-mail: s.gnatyuk@nau.edu.ua

КОВТУН Владислав Юрьевич, к.т.н.

Дата и место рождения: 1978 год, г. Кировоград, Украина.
Образование: Харьковский военный университет, 2000 год.
Должность: доцент кафедры безопасности информационных технологий.
Научные интересы: информационная безопасность, быстрые арифметические преобразования в полях Галуа, криптосистемы с открытым ключом, криптоанализ криптографических преобразований с открытым ключом.
Публикации: более 50 научных публикаций, среди которых статьи в специализированных и зарубежных научных журналах, материалы конференций, патенты и др.
E-mail: vladislav.kovtun@gmail.com

БЕРДНИК Оксана Михайловна, к.т.н.

Год и место рождения: 1973 год, г. Нежин, Черниговская область, Украина.
Образование: Нежинский государственный педагогический институт им. Н.В. Гоголя (с 2004 года – Нежинский государственный университет имени Николая Гоголя), 1996 год.
Должность: доцент кафедры прикладной математики с 2012 года.
Научные интересы: математическое моделирование, вычислительные методы, прикладные задачи математики.
Публикации: более 30 публикаций, среди которых научные статьи, материалы и тезисы докладов на конференциях, учебные пособия.
E-mail: o_berdnik@mail.ru

КОВТУН Мария Григорьевна

Дата и место рождения: 1989 год, г. Львов, Украина.
Образование: Национальный авиационный университет, 2011 год.
Должность: аспирант кафедры безопасности информационных технологий с 2014 года.
Научные интересы: информационная безопасность, быстрые криптографические преобразования в полях Галуа, криптосистемы с открытым ключом, криптоанализ криптографических преобразований с открытым ключом.
E-mail: mg.kovtun@gmail.com

Аннотация. *Криптографические преобразования с открытым ключом обладают значительной вычислительной и пространственной сложностью. В связи с этим, актуальной научно-технической задачей является повышение производительности таких преобразований. В работе рассматриваются подходы к повышению производительности операции деления больших целых чисел двойной точности на большие числа одинарной точности на основе расширенного алгоритма Евклида. К таким походам относятся: оперирование отличными от нуля машинными словами в наиболее часто использующихся операциях (сдвиги влево и вправо, сложение и вычитание больших чисел); использование приближенного сравнения больших целых чисел, без необходимости пословного сравнения (сравнение номеров старших битов этих чисел); знание закона изменения параметров уравнения Безу (эксплуатируется в предыдущих двух подходах). Предложенные подходы успешно реализованы в модифицированном алгоритме, который был запрограммирован. Для сравнения проводились эксперименты над числами, с условием, что двоичная длина делимого в два раза превосходит двоичную длину делителя для различного соотношения заполненной и общей двоичной длины большого числа. Модифицированный алгоритм показал лучшую производительность в 1,5-3 раза, с ростом двоичной длины делимого и делителя.*

Ключевые слова: *деление, остаток деления, большие целые числа, расширенный алгоритм Евклида.*

Введение

Криптографические методы защиты информации активно используются во всех современных информационных системах для решения задач, связанных с обеспечением конфиденциальности, целостности, авторства, аутентификации и т.д. Наибольшее количество перечисленных задач, перекрывается криптографическими преобразованиями с открытым ключом. Известно [37][38], что в отличие от симметричных криптографических преобразований, преобразования с открытым ключом обладают значительной вычислительной и пространственной сложностью. В связи с этим, актуальной научно-технической задачей является повышение производительности криптографических преобразований с открытым ключом. Широко распространенные криптосистемы с открытым ключом, такие как RSA, DSA, ECDSA [8], ДСТУ 4145-2002 [7] и другие, оперируют большими целыми числами. Среди наиболее часто используемых операций можно выделить: сложение / вычитание, умножение, приведение по модулю, деление с остатком и без. Большое число публикаций [2][9][12][17][25][34], посвященных оптимизации операциям умножения и приведения по модулю, не уделяют должного внимания операциям деления. Данная операция довольно часто используется на втором плане – при генерации общесистемных параметров, восстановлении точки эллиптической кривой из сжатого состояния, поиске случайно точки эллиптической кривой [7][8], этот список может быть продолжен.

В связи с этим, авторы проявили интерес к операции деления больших целых чисел двойной точности на большие целые числа одинарной точности с остатком и без него, с использованием расширенной версии алгоритма Евклида (РАЕ). Выбор соотношения длин делимого и делителя вызван необходимостью производить деление после умножения чисел одинарной точности, без их предварительного приведения по модулю. В дальнейшем, авторы сконцентрируют внимание на РАЕ, который является основным инструментом современной теории чисел.

Анализ существующих работ

Учитывая большое число научных публикаций и подходов к реализации операции приведения по модулю целых чисел, авторами был проведен анализ публикаций по тематике деления целых чисел и приведения по модулю, что позволило дополнить и обобщить классификацию предложенную [13]. На Рис. 1 и Рис. 2 приводится графическая интерпретация классификации алгоритмов деления с остатком и без, а также приведения по модулю, учитывающая различные аспекты самих алгоритмов, направленных на их оптимизацию:

1. Представление целого числа (делимого и делителя): двоичное (обычное) [25]; четверичное [25]; двоичное сокращенное (двоичное DRF) [17]; избыточное статическое [5][17]; избыточное динамическое [5]; несмежное (NAF) [20]; остаточных классов (RNS) [14][16]; двоичное, с отложенным переносом (DCF) [13].

2. Делитель – произвольные числа, не обязательно простые. Алгоритмы основаны на идее деления «в столбик», Барретта и Монтгомери [31].

3. По получаемым результатам. При делении целых чисел могут представлять интерес, как частное, так и остаток. В связи с этим можно выделить следующие алгоритмы:

– Деление целых чисел для получения только частного.

– Деление целых чисел для получения только остатка – приведение по модулю.

– Деление целых числе для получения, как частного, так и остатка.

4. Делитель – простые числа. Учитывая специфику представления делителя (модуля) в двоичном виде, можно оптимизировать алгоритм приведения, согласно следующим классам простых чисел:

– Простые числа общего вида. Алгоритмы основаны на идее деления «в столбик», Барретта и Монтгомери [31].

– Простые числа, являющиеся обобщенными мерсеновыми и псевдо-мерсеновыми. Специализированные алгоритмы, учитывающие свойства мерсеновых и псевдо-мерсеновых чисел в общем виде, а также конкретные простые мерсеновы и псевдо-мерсеновы числа [24][26][10][15].

5. Направление анализа делимого. Алгоритмы, учитывающие специфику направления анализа делимого, с начала или конца:

– С начала (с младших бит) в алгоритме Монтгомери [18][32].

– С конца (со старших бит) в алгоритме Барретта [18].

– Двусторонние и многосторонние [18].

6. Модификации алгоритма. Учитывая различные подходы к делению и приведению, выделяют несколько основных алгоритмов и их модификаций [11]:

– Классический («школьный», «деление в столбик») [1][2][36].

– Монтгомери [12] и его модификации [17].

– Барретта [9] и его модификации [17][36].

– Универсальный и жестко запрограммированный специальный делитель (модуль). Как правило, применяется для простых модулей специального вида, например, Мерсеновых, псевдо-Мерсеновых или обобщенных Мерсеновых чисел [24][26].

– Расширенный алгоритм Евклида [3].

– Алгоритм итераций Ньютона [27][25].

– Алгоритмы деления Джебелина [34] и его модификации.

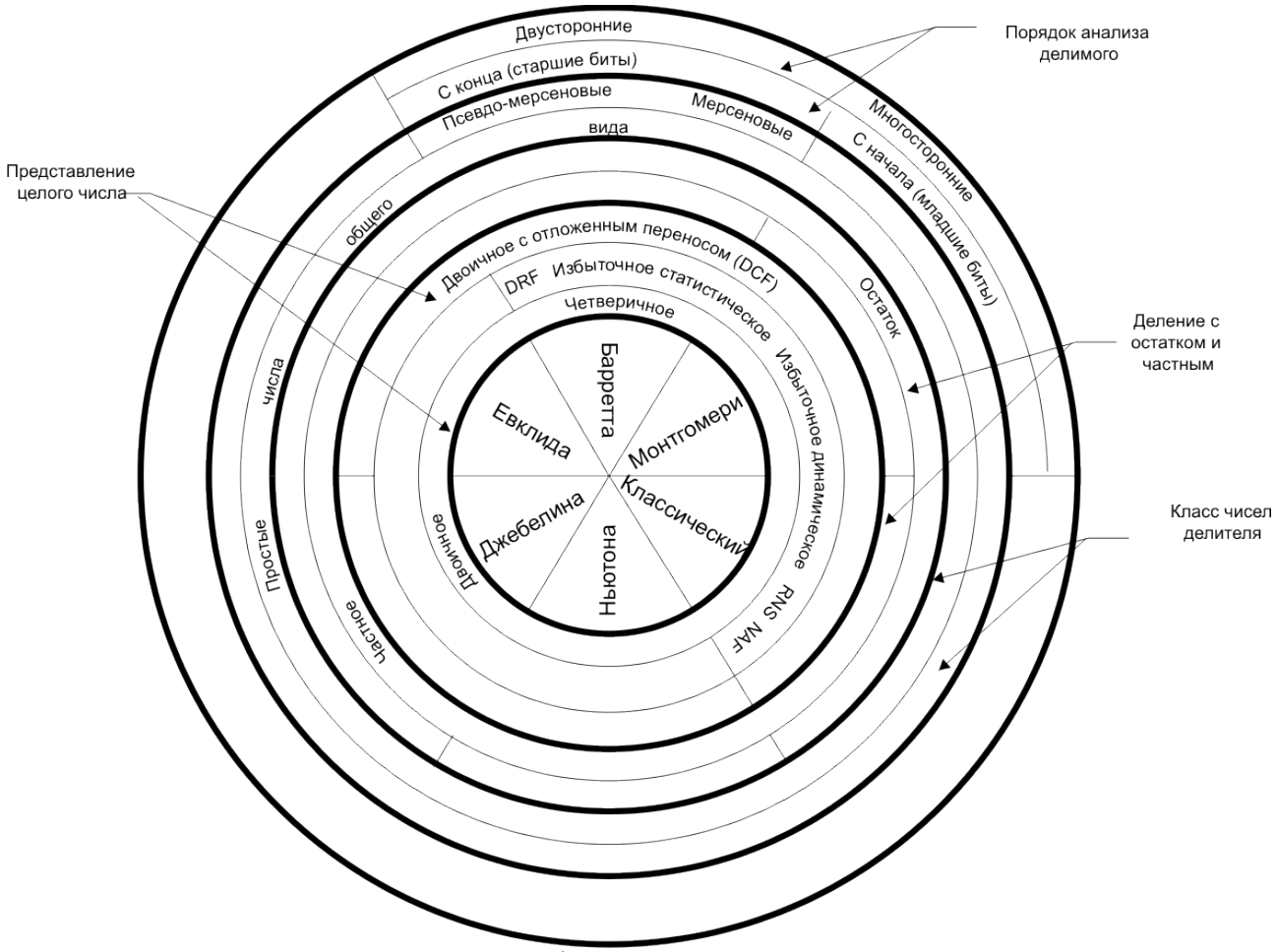

Рис. 1. Классификация алгоритмов деления

7. Точность получения результата – приближение. Не всегда представляет интерес точный результат (частное / остаток), достаточно получить результат, незначительно отличающийся от точного значения. Данный подход может быть применен ко всем известным алгоритмам приведения по модулю. Можно выделить несколько основных алгоритмов:

– частичное приведение для алгоритма Монтгомери [18];

– частичное приведение для алгоритма Барретта [18] ;

– частичное приведение для алгоритма приведения «в столбик» ;

– частичное приведение для модуля специального вида [19].

8. Распараллеливание. Большинство современных компьютеров, обладают несколькими процессорами или одним процессором с несколькими ядрами. Известны следующие алгоритмы с распараллеливанием деления (приведения по модулю):

– двустороннее частичное приведение для алгоритма Монтгомери [18][28] ;

– многостороннее частичное приведение для алгоритма Монтгомери [18][28] ;

– двустороннее частичное приведение для алгоритма Барретта [18][28] ;

– многостороннее частичное приведение для алгоритма Барретта [18][28] ;

– приведение для алгоритма Барретта на основе DCF [13]. В данной работе подлежит

распараллеливанию алгоритм Комба – частичного умножения целых чисел, что позволяет его применять не только к алгоритму Барретта, но и Монтгомери;

– распараллеливание алгоритма Карацубы, предложенное Джебелином [33][34] ;

– алгоритм распараллеливания представления многократной точности через одинарную точность целого числа: параллельный циклический метод приведения по модулю [27].

9. Предвычисления. При выполнении операции деления (приведения по модулю), используется один и тот же делитель (модуль), то представляет интерес предвычислить заранее некоторые промежуточные значения. Данный подход может быть применен ко всем известным на сегодня алгоритмам. Механизм получил развитие в следующих алгоритмах:

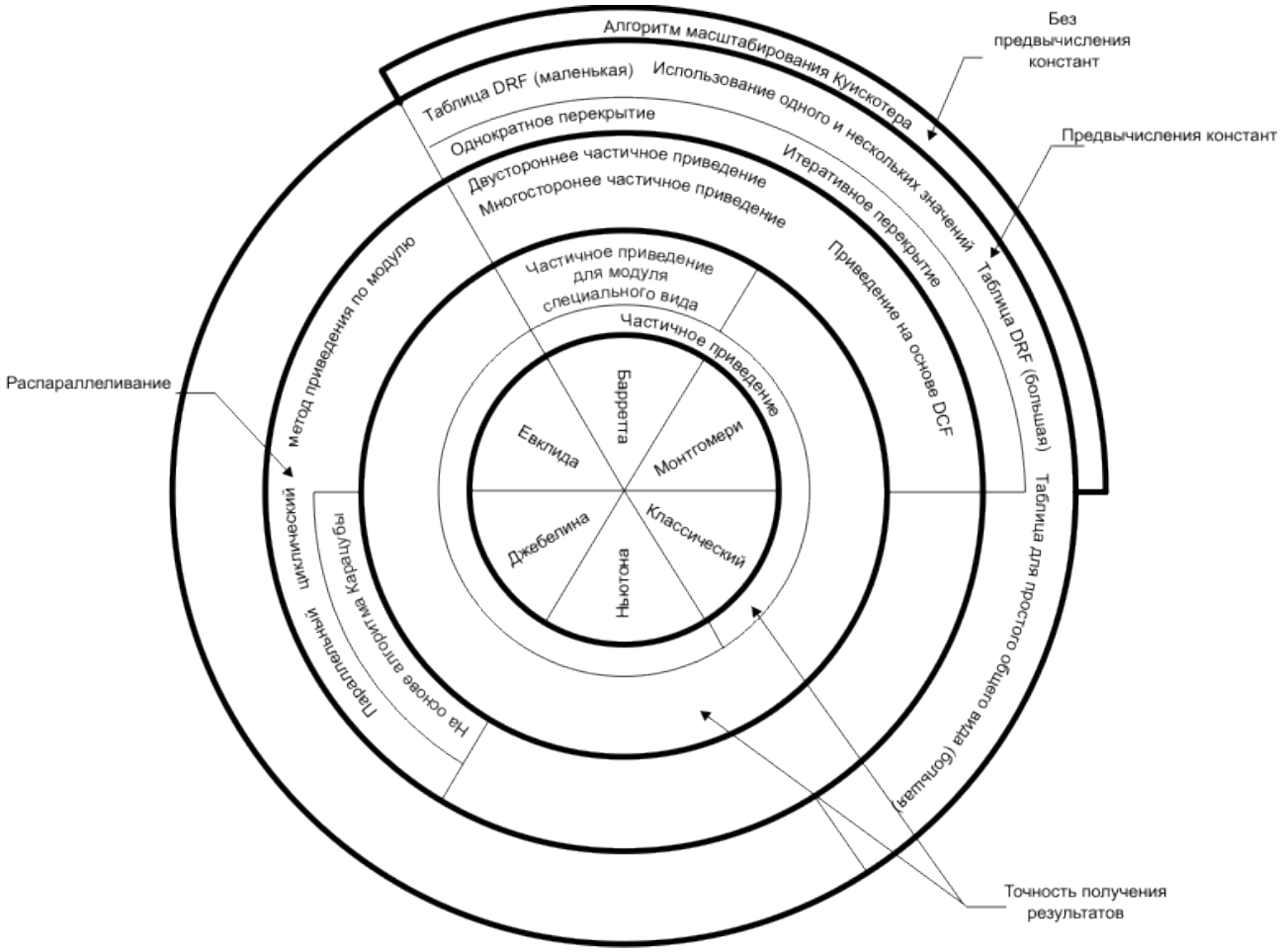

Рис. 2. Классификация алгоритмов деления

– однократное перекрытие алгоритма Барретта [17];

– итеративное (чередующееся) перекрытие алгоритма Барретта [10];

– использование одного или нескольких предвычисленных значений [21];

– таблица предвычислений для простого общего вида (большая) [21];

– таблица предвычислений для простого специального вида DRF (большая) [21];

– таблица предвычислений для модуля специального вида DRF (маленькая) [21].

10. Без предвычислений констант. Учитывая тот факт, что в алгоритмах Монтгомери и Барретта, необходимо предварительное вычисление ряда констант. Существует возможность уйти от предварительного вычисления констант:

– без вычислений констант для алгоритма Монтгомери (алгоритм масштабирования Куискотера) [10];

– без вычислений констант для алгоритма Барретта (алгоритм масштабирования Куискотера) [10].

11. Алгоритм умножения. Следует заметить, что в алгоритмах деления «в столбик» и итераций Ньютона, деления и приведения по модулю Барретта и Монтгомери, используется умножение целых чисел различной длины, от реализации которых, напрямую зависит производительность самого деления и приведения. Известно применение алгоритмов умножения: Алгоритм Комба [29]; Алгоритм Карацубы [30]; Расширенный алгоритм Евклида оперирует двоичным представлением и операциями над числами: сдвиг влево и вправо, сложение и вычитание. Данные бинарные операции,

по сути, реализуют операцию умножения целых чисел [3][4].

Классификация алгоритмов деления и приведения по модулю дает возможность выполнять предвычисления и использовать приближенные операции, например, для операции сравнения чисел в РАЕ.

Основная часть

Рассмотрим расширенный алгоритм Евклида для деления больших целых чисел: для любых целых чисел a и b, $b \neq 0$, $a,b \in \mathbf{Z}$ существуют единственным образом определенные целые числа q, r, которые удовлетворяют условие b. Где $b \neq 0$ - остаток от деления, q - неполное частное при делении a на $a = b \cdot q + r, 0 \le r < b$.

Ниже представлено более подробное описание алгоритма. Изначально $r_0 = a$, $r_1 = b$ и $r_2,\ldots,r_n$ - последующие делители в РАЕ, тогда последовательно вычисляются следующие равенства:

b,

$r_0 = a$,

$r_2 = r_3 \cdot q_3 + r_4, 0 \le r_4 < r_3$,

. . . . . .

$r_2,...,r_n$,

$r_{n-1} = r_n \cdot q_n$.

Вычисления прекращаются, когда остаток от деления r_i будет меньше $r_2 = r_3 \cdot q_3 + r_4, 0 \le r_4 < r_3$. Формализуем описанный алгоритм в виде псевдокода.

Алгоритм 1. Расширенный алгоритм Евклида для деления больших целых чисел, когда делимое длиннее делителя в два раза.

Вход: $a,b \in \mathbf{Z}$, $r_{n-1} = r_n \cdot q_n$, $n_a \leftarrow \lceil \log_2 a/w \rceil$, где n_a - количество машинных слов, которое занимает делимое; $n_b \leftarrow \lceil n_a/2 \rceil$ - количество машинных слов, которое занимает делитель; $a,b \in Z$ - ширина машинного слова, обычно $w = 32$.

Выход: $q,r \in \mathbf{Z}$.

1. $r \leftarrow a$ $m \leftarrow b$, $w = 32$, $q,r \in Z$.

2. While $r > m$ do

2.1 $m \leftarrow b$, $s \leftarrow 1$.

3. While $q \leftarrow 0$ do

3.1. While $r > m$ do

3.1.1. $m \leftarrow m << 1$, $s \leftarrow s << 1$.

3.2. $r > b$, $q \leftarrow q + s$.

4. Return $m \leftarrow m >> 1$.

При внимательном рассмотрении Алгоритма 1, можно обозначить ряд направлений для дальнейшего изучения и совершенствования РАЕ:

В п. 1 происходит инициализация параметров уравнения Безу [4]: остатка от деления $r \leftarrow a$; выровненного по старшему биту делимого s; промежуточного делителя k, для последующего вычитания; промежуточное частное w, искомое частное $r \leftarrow a$.

В цикле п. 2 происходит проверка условия s до тех пор, пока не произойдет выравнивание номеров старших битов промежуточного делителя $m \leftarrow b$ и делимого a. Пока условие п.2 истинно, происходит сдвиг влево промежуточного делителя m и промежуточного частного m. При сравнении больших чисел, выполняется вычислительно сложная проверка s, $r > m$ для всех слов и на каждой итерации. Хотя заведомо известно, что количество машинных слов, которое занимает делитель, в два раза меньше – делимого, т.е. происходят избыточные вычислительно сложные сравнения. Число данных проверок можно значительно сократить, используя подход приближенного сравнения больших целых чисел, с помощью сравнения номеров старших битов $r_i > m_i$ и $i = \overline{n-1,0}$, а с помощью найденной разницы m (разница между номерами старших битов s и r), совершить сдвиги влево на m бит п. 2.1 за одну итерацию. Полностью от проверки условия п. 2 избавиться невозможно, т.к. в случае, когда номера старших битов k и r - раны между собой, может возникнуть неопределенность. Чтобы сократить вычислительную сложность при сравнении m, k, следует оперировать лишь значимыми машинными словами.

В цикле деления п. 3, для проверки условия $r > b$, происходит, как и в предыдущем пункте, вычислительно сложная проверка условия m, $r_i > m_i$ для всех машинных слов. Изначально сравнение целых чисел следует производить, используя подход сравнения номеров старших битов, и в случае равенства старших битов - проводить сравнение по значимым словам. Следует учитывать закон изменения параметров уравнения Безу, а также что b занимает в 2 раза меньше машинных слов чем r.

В п. 3.1 происходит промежуточное деление, которое требует циклической проверки условия $r > b$, $r_i > b_i$ для всех слов. Сдвиг вправо всех машинных слов на 1 выровненного делителя $i = \overline{n-1,0}$ и промежуточного частного $r < m$ выполняется в п. 3.1.1, пока выполняется условие п. 3.1, в противном случае выполняется сложение $q \leftarrow q + s$ и вычитание $r > b$ по всем машинным словам в п. 3.2. По аналогии с предыдущими замечаниями, количество проверок п. 3.1 можно сократить, перейдя к сравнению номеров старших битов m и s. Найденную разницу k (разница между номерами старших битов m и r) использовать для сдвигов вправо за 1 итерацию в п. 3.1.1. Также следует учесть известный закон изменения параметров уравнения Безу:

– для сравнения больших чисел, когда номера старших битов m и k - равны;

– для вычитания в п. 3.2 не всех машинных слов, а лишь значимых.

В случае невыполнения условия п. 3, происходит переход к п. 4 и возврат искомого частного q и остатка от деления s.

Оценка сложности PAE. При оценке сложности PAE авторы не отличают операции присвоения, сдвига, сложения и вычитания, из-за их близкой вычислительной сложности:

$$I(A_1) = 5 \cdot k \left\lceil \frac{n}{w} \right\rceil L + C \cdot k \left(\frac{1}{3} \left\lceil \frac{n}{w} \right\rceil + \left\lceil \frac{n - n_a}{w} \right\rceil + \left\lceil \frac{n - n_b}{w} \right\rceil \right), \quad (1)$$

где q - арифметические операции; r - операции сравнения; n_a - номер старшего бита делимого; n_b - номер старшего бита делителя; $I(A_1) = \left\lceil \frac{n}{w} \right\rceil L \cdot \left(6 + \frac{8}{3}k\right) + \frac{5}{2}k \cdot C + \frac{1}{2} \left\lceil \frac{n+n_a}{w} \right\rceil C$ - количество бит, зарезервированных в памяти для хранения максимального возможного числа; k - разница между номерами старших битов делителя и делимого; C - ширина машинного слова.

Сложность операции сравнения и сдвигов в п.п. 2 - 2.1 - $k \left\lceil \frac{n - n_a}{w} \right\rceil C + 2Lk \left\lceil \frac{n}{w} \right\rceil$, сложность операции сравнения п. 3 - $\frac{k}{2} \cdot \frac{2}{3} \left\lceil \frac{n}{w} \right\rceil C$, где $\mathbf{P}\left(r \underset{z}{>} b \mid z = \left\lceil \frac{n}{w} \right\rceil \right) = \frac{2}{3}$, сложность операции сравнения п. 3.1 - $\frac{2}{3} \cdot k \left\lceil \frac{n - n_b}{w} \right\rceil C$, где $\mathbf{P}\left(r \underset{z}{<} m \mid z = \left\lceil \frac{n - n_b}{w} \right\rceil \right) = \frac{2}{3}$, сдвигов п. 3.1.1 - $2k \left\lceil \frac{n}{w} \right\rceil L$, сложения и вычитания (п. 3.2) - $2 \cdot \frac{k}{2} \left\lceil \frac{n}{w} \right\rceil L$.

Предложенные выше подходы, авторы реализовали в виде модифицированного Алгоритма 2. Обозначим через R, M, S, T номера старших битов остатка от деления R, выровненного промежуточного делителя M, промежуточного частного S, делителя T. $\tilde{r}$, $\tilde{m}$, $\tilde{s}$, $\tilde{t}$ - количество значимых слов r, $\tilde{m}$, $\tilde{s}$, $\tilde{t}$, соответственно; $sgf(\cdot)$ – функция, которая вычисляет количество значимых слов большого числа; $msb(\cdot)$ - функция, которая вычисляет, номер старшего бита большого числа; $msb(\cdot,\tilde{\cdot})$ - функция, которая одновременно вычисляет количество значимых слов и номер старшего бита большого числа; $\mathrm{sgf}(\cdot)$ - сравнение больших целых чисел по значимым словам $\tilde{t}$.

Алгоритм 2. Модифицированный PAE для деления больших целых чисел, когда делимое длиннее делителя в два раза.

Вход: $a, b \in \mathbf{Z}$, $msb(\cdot,\tilde{\cdot})$, $n_a \leftarrow \lceil \log_2 a / w \rceil$, где n_a - количество машинных слов, которое занимает делимое; $n_b \leftarrow \lceil n_a / 2 \rceil$ - количество машинных слов, которое занимает делитель; $a, b \in Z$ - ширина машинного слова, обычно $a, b \neq 0$. Выход: $n \leftarrow \left\lceil \frac{\log_2 a}{w} \right\rceil$.

1. n, w, $s \leftarrow 1$, $q, r \in Z$, $r \leftarrow a$, $S \leftarrow 1$, $\tilde{s} \leftarrow 1$
2. $q \leftarrow 0$, $t \leftarrow b$, $S \leftarrow 1$, $\tilde{t} \leftarrow \tilde{m}$.
3. $M \leftarrow \mathrm{msb}(m, \tilde{m})$.
4. if $R \leftarrow \mathrm{msb}(r, \tilde{r})$ then $T \leftarrow M$, $\tilde{t} \leftarrow \tilde{m}$, $k \leftarrow R - M$, $S \leftarrow S + k$.
5. $m \leftarrow m \underset{\tilde{m}}{<<} k$.
6. While $s \leftarrow s \underset{\tilde{s}}{<<} k$ do
   6.1. $M \leftarrow M + k$.
   6.2. $m \leftarrow m \underset{\tilde{m}}{<<} 1$, $s \leftarrow s \underset{\tilde{s}}{<<} 1$, $M \leftarrow M + 1$, $S \leftarrow S + 1$, $m \leftarrow m \underset{\tilde{m}}{<<} 1$.
7. While $(R > T) or \left(r \underset{\tilde{t}}{\geq} t \right)$ do
   7.1. $k \leftarrow M - R$.
   7.2. if $S \leftarrow S + 1$ then $\tilde{m} \leftarrow sgf(m)$, $(R > T) or \left(r \underset{\tilde{t}}{\geq} t \right)$.
   7.3. $m \leftarrow m \underset{\tilde{m}}{>>} k$, $s \leftarrow s \underset{\tilde{s}}{>>} k$.
   7.4. $S \leftarrow S - k$, $M \leftarrow M - k$.
   7.5. $s \leftarrow s \underset{\tilde{s}}{>>} k$.
   7.6. While $(R < M) or \left(r \underset{\max(\tilde{m}, \tilde{r})}{<} m \right)$ do
      7.6.1. $S \leftarrow S - k$.
      7.6.2. $m \leftarrow m \underset{\tilde{m}}{>>} 1$, $s \leftarrow s \underset{\tilde{s}}{>>} 1$.
      7.6.3. $M \leftarrow M - 1$, $S \leftarrow S - 1$.
      7.6.4. $m \leftarrow m \underset{\tilde{m}}{>>} 1$.
   7.7. $s \leftarrow s \underset{\tilde{s}}{>>} 1$.
   7.8. $r \leftarrow r \underset{\tilde{r}}{-} m$.
   7.9. $S \leftarrow S - 1$.
   7.10. $q \leftarrow q + s$.
8. Return (q, r).

В п. 1 происходит инициализация параметров уравнения Безу: остаток от деления n, выровненный промежуточный делитель w по старшему биту делимого для последующего вычитания, промежуточное частное $s \leftarrow 1$, искомое частное $q, r \in Z$, дополнительная переменная $r \leftarrow a$, номер старшего бита промежуточного частного $S \leftarrow 1$, количество значимых слов $s \leftarrow 1$.

Можно сократить количество вычислительно сложных проверок п. 6, поскольку заранее известно, что двоичная длина r в 2 раза превышает двоичную длину m. Для этого в п. 2 происходят вычисления номеров старших битов и количества значимых слов: M, $\tilde{m}$, $S \leftarrow 1$, $\tilde{s} \leftarrow 1$, а также присвоения M, $\tilde{m}$. В дальнейшем вычисляется разница R между номерами старших битов r и m в п. 3.

В п. 4 осуществляется проверка условия m, после чего, если условие – истинно, за 1 шаг производятся сдвиги влево только значимых слов m и m на данную разницу m_j бит. После указанных операций номера старших битов s_j и m будут равны. В дальнейшем вычисляются номера старших битов модифицированных m и $M == R$, а также количество значимых слов s_j, п. 5.

Дополнительное условие сравнения номеров старших битов m и m в п. 6, позволяет выполнять

вычислительно сложную проверку (по значимым словам) $r \underset{\tilde{m}}{\geq} m$, лишь когда $sgf(m)$.

Если условие п. 6 истинно, то в.п. 6.1 вычисляется количество значимых слов $r \underset{\tilde{m}}{\geq} m$ и осуществляются сдвиги влево на 1 бит значимых слов m_j, R и s_j, $\tilde{s} \leftarrow sgf(s)$. После сдвигов, номера старших битов m и $j = \overline{\tilde{m}-1,0}$ принимают новые значения, и в дальнейшем вычисляется s_j. Если условие п. 6 ложно, то осуществляется переход к п. 7.

Проверка условия (по значимым словам) $r \underset{\tilde{t}}{\geq} t$ в п. 7 (промежуточное деление) производится лишь, когда m. Если условия п. 7 - истинно, то выполняются действия п.п. 7.1 – 7.10, в противном случае - переход к п. 8.

П.п. 7.1 -7.5 позволяют свести количество вычислительно сложных сравнений больших чисел в п.7.6 к минимуму. В п. 7.1 вычисляется разница $R > T$ между номерами старших битов $r_j \underset{\tilde{t}}{\geq} t_j$ и $j = \overline{\tilde{t}-1,0}$. Если разница положительная, то вычисляется количество значимых слов k и m, п.7.2. В дальнейшем производятся сдвиги вправо значимых слов m и s на m бит, после чего s, а также снова вычисляются номера старших битов (см. п.5.4) и количество значимых слов k, см. п. 7.5.

В цикле п. 7.6 производится сравнение номеров старших битов $R < M$. Если m, то выполняется дополнительное сравнение $r \underset{\max(\tilde{r},\tilde{m})}{<} m$ по значимым словам. Если условие п.7.6 – истинно, то выполняются п.п. 7.6.1 – 7.6.4, в противном случае переход к п. 7.7.

На шаге п. 7.6.1 вычисляется количество значимых слов $sgf(m)$, после чего выполняются сдвиги вправо на 1 бит значимых слов и вычисляются новые номера старших битов m и s (п. 7.6.2 – п. 7.6.3), в дальнейшем определяются значимые слова $\tilde{m} \leftarrow sgf(m)$, п. 7.6.4.

В п. 7.7 вычисляется количество значимых слов $\tilde{m} \leftarrow sgf(m)$, которое нужно для вычитания $r \leftarrow r \underset{\tilde{r}}{-} m$ п. 7.8. После вычитания в п. 7.9 вычисляется номер старшего бита и количество значимых слов $\tilde{r}$. В п. 7.10 выполняется накопление частного $q \leftarrow q + s$.

В п. 8 происходит возврат значений остатка от деления $\tilde{r} \leftarrow sgf(r)$ и искомого частного $r \leftarrow r \underset{\tilde{r}}{-} m$.

Оценка сложности модифицированного РАЕ. Оценку сложности алгоритма, можно представить:

$$I(A_2) = L\left(\begin{array}{l} 63 + \left\lceil \frac{n_a}{w} \right\rceil \left(\frac{5}{3} + \frac{7}{12}k\right) + \\ + \left\lceil \frac{k}{w} \right\rceil \left(\frac{k}{3} + \frac{5}{3}\right) + \left\lceil \frac{n}{w} \right\rceil \frac{k}{2} + \frac{207}{12}k \end{array}\right) + C\left(4 + \frac{5}{2}k\right), \quad (2)$$

где L - арифметические операции; C - операции сравнения; n_a - номер старшего бита делимого; n_b - номер старшего бита делителя; n - количество бит, зарезервированных в памяти для хранения максимального возможного числа; k - разница между номером старшего бита делимого и делителя; w - ширина машинного слова.

Вычислительную сложность операции вычисления номеров старших битов и значимых слов п. 2 - $54L$; операции проверки $k > 0$, сдвиги п. 4. и присвоений номеров старших битов - $C + \left\lceil \frac{n_a}{w} \right\rceil L + \left\lceil \frac{k}{w} \right\rceil L + 2L$; вычисление значимых слов п. 5 и сравнение п. 6 составит - $2L + C + C\left\lceil \frac{n_a}{w} \right\rceil \left\lceil \frac{n_a}{w} \right\rceil^{-1}$, то есть операция сравнения больших чисел выполняется лишь в случае, когда $R == M$ и вероятность того, что сравниваться будут числа по всем словам $\mathbf{P}\left(r \underset{z}{>} m \mid z = \left\lceil \frac{n_a}{w} \right\rceil\right) = \left\lceil \frac{n_a}{w} \right\rceil^{-1}$; вычисления значимых слов п. 6.1 и операции сдвига в п. 6.2 - $\frac{2}{3}L + \frac{1}{3}\left\lceil \frac{n_a}{w} \right\rceil L + \frac{1}{3}\left\lceil \frac{k}{w} \right\rceil L$; сравнение п. 7 - $\frac{k}{2}C + \left\lceil \frac{n_b}{w} \right\rceil^{-1} \left\lceil \frac{n_b}{w} \right\rceil C$, где $\mathbf{P}\left(r \underset{z}{>} t \mid z = \left\lceil \frac{n_b}{w} \right\rceil\right) = \left\lceil \frac{n_b}{w} \right\rceil^{-1}$ - вероятность того, что сравнение чисел проводится по всем словам; $\frac{k}{2}C$ - сложность операции проверки $k > 0$, вычисление значимых слов – $\frac{2}{3}kL + \frac{2}{3}kL$ в п. 7.2, где $\mathbf{P}(k > 0) = \frac{1}{3}$; сдвиги вправо п. 7.3 - $\frac{k}{3}\left\lceil \frac{n_a}{w} \right\rceil L + \frac{k}{3}\left\lceil \frac{k}{w} \right\rceil L$; присвоение новых номеров старших битов п. 7.4 и вычисление значимых слов п. 7.5 - $\frac{4}{3}kL$; операция сравнения п. 7.6 - $kC + \frac{k}{2}C\left\lceil \frac{n_a}{w} \right\rceil \left\lceil \frac{n_a}{w} \right\rceil^{-1}$, где $\mathbf{P}\left(r \underset{z}{<} m \mid z = \left\lceil \frac{n_a}{w} \right\rceil\right) = \left\lceil \frac{n_a}{w} \right\rceil^{-1}$ - вероятность сравнения всех машинных слов представления чисел при условии равенства номеров старших битов; вычисление значимых слов и сдвигов вправо п.п. 7.6.1 – 7.6.4 - $kL + \frac{k}{3}\left\lceil \frac{n_a}{w} \right\rceil L + \frac{k}{3}\left\lceil \frac{k}{w} \right\rceil L$, где $\mathbf{P}(r < m) = \frac{1}{3}$ - вероятность попадания в цикл п.7.6; вычисление значимых слов п. 7.7 - $k\left(C + \frac{1}{3}C\right)$; вычитание п.7.8 - $\frac{k}{4}\left(1 + \left\lceil \frac{n_a}{w} \right\rceil\right)$; вычисление номера старшего бита и количества значимых слов п. 7.9 - $\frac{27 \cdot k}{2}$; сложение в п.7.10 - $\frac{k}{2}\left\lceil \frac{n}{w} \right\rceil L$.

Таблица 1

Теоретические оценки сложностей алгоритмов PAE и MPAE,
выраженные в арифметических операциях и операциях сравнения

ρ	1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
$64\,бита \mid n_a = 2,\ 32\,бита \mid n_b = 1$										
k	29	32	35	39	42	45	48	51	55	58
$I_L(A_1)$	320	352	384	416	448	480	512	544	576	608
$I_L(A_2)$	701	788	860	932	1003	1075	1147	1219	1290	1362
$I_C(A_1)$	74	105	114	124	134	143	153	162	172	181
$I_C(A_2)$	77	85	93	101	109	117	125	133	141	149
$I_L(A_1)/I_L(A_2)$	0,46	0,45	0,45	0,45	0,45	0,45	0,45	0,45	0,45	0,45
$I_C(A_1)/I_C(A_2)$	0,97	1,24	1,24	1,23	1,23	1,23	1,23	1,22	1,22	1,22
$128\,бит \mid n_a = 4,\ 64\,бита \mid n_b = 2$										
k	61	67	74	80	87	93	99	106	112	119
$I_L(A_1)$	1 280	1 408	1 536	1 664	1 792	1 920	2 048	2 176	2 304	2 432
$I_L(A_2)$	1 589	1 783	1 946	2 109	2 271	2 434	2 648	2 814	2 980	3 145
$I_C(A_1)$	233	303	330	358	385	413	508	540	571	603
$I_C(A_2)$	157	173	189	205	221	237	253	269	285	301
$I_L(A_1)/I_L(A_2)$	0,81	0,79	0,79	0,79	0,79	0,79	0,77	0,77	0,77	0,77
$I_C(A_1)/I_C(A_2)$	1,49	1,75	1,75	1,75	1,75	1,75	2,01	2,01	2,01	2,01
$512\,бит \mid n_a = 16,\ 256\,бит \mid n_b = 8$										
k	242	268	293	319	344	370	396	421	447	472
$I_L(A_1)$	20 480	22 528	24 576	26 624	28 672	30 720	32 768	34 816	36 864	38 912
$I_L(A_2)$	10 479	11 713	12 972	14 096	15 394	16 718	18 067	19 441	20 617	22 030
$I_C(A_1)$	2 959	3 441	3 348	3 957	4 616	4 946	5 681	6 467	7 304	8 191
$I_C(A_2)$	609	673	737	801	865	929	993	1 057	1 121	1 185
$I_L(A_1)/I_L(A_2)$	1,95	1,92	1,89	1,89	1,86	1,84	1,81	1,79	1,79	1,77
$I_C(A_1)/I_C(A_2)$	4,86	5,11	4,54	4,94	5,34	5,32	5,72	6,12	6,52	6,91
$4096\,бит \mid n_a = 128,\ 2048\,бит \mid n_b = 64$										
k	1 950	2 155	2 360	2 564	2 769	2 974	3 179	3 384	3 588	3 793
$I_L(A_1)$	1 310 720	1 441 792	1 572 864	1 703 936	1 835 008	1 966 080	2 097 152	2 228 224	2 359 296	2 490 368
$I_L(A_2)$	404 284	454 263	504 494	557 238	610 029	664 049	720 889	777 469	837 074	896 214
$I_C(A_1)$	137 953	167 361	197 173	232 055	266 936	304 251	347 242	389 626	438 092	485 544
$I_C(A_2)$	4 879	5 391	5 903	6 415	6 927	7 439	7 951	8 463	8 975	9 487
$I_L(A_1)/I_L(A_2)$	3,24	3,17	3,12	3,06	3,01	2,96	2,91	2,87	2,82	2,78
$I_C(A_1)/I_C(A_2)$	28,27	31,04	33,40	36,17	38,54	40,90	43,67	46,04	48,81	51,18
$16384\,бита \mid n_a = 512,\ 8132\,бита \mid n_b = 256$										
k	8 162	8 981	9 800	10 620	11 439	12 258	13 077	13 896	14 716	15 535
$I_L(A_1)$	20 971 520	23 068 672	25 165 824	27 262 976	29 360 128	31 457 280	33 554 432	35 651 584	37 748 736	39 845 888
$I_L(A_2)$	6 423 899	7 180 839	7 963 161	8 761 471	9 585 982	10 431 792	11 292 361	12 180 360	13 082 299	14 012 487
$I_C(A_1)$	2 084 372	2 524 758	3 007 316	3 521 503	4 087 595	4 683 694	5 333 320	6 025 118	6 744 490	7 519 822
$I_C(A_2)$	20 409	22 457	24 505	26 553	28 601	30 649	32 697	34 745	36 793	38 841
$I_L(A_1)/I_L(A_2)$	3,26	3,21	3,16	3,11	3,06	3,02	2,97	2,93	2,89	2,84
$I_C(A_1)/I_C(A_2)$	102,13	112,43	122,72	132,62	142,92	152,82	163,11	173,41	183,31	193,61

Сравнение алгоритмов. Для объективности сравнения, рассматриваются теоретические оценки вычислительной сложности PAE и MPAE, а также производительности программных реализаций.

В таблице 1 представлены теоретические оценки вычислительных сложностей классического алгоритма Евклида и модифицированного, в зависимости от двоичной длины делителя и делимого. Так, под ρ, понимается отношение числа используемых бит к числу зарезервированных в большом числе (делителе). Двоичная длина делимого изменяется в диапазоне от 64 бит, до 16384 бит. Ширина машинного слова $w = 32$. Таблица с оценками сложностей разбита на блоки, каждый из которых описывает соответствующую часть, например, $64 бита \mid n_a = 2, 32 бита \mid n_b = 1$ указывает, что двоичная длина делимого составляет 64 бита (2 машинных слова), а делителя – 32 (одно машинное слово). Для удобства сравнения, количество арифметических операций, соответствующих алгоритмов $I_L(A_1)$ и $I_L(A_2)$, представлены друг над другом. Аналогичным образом представлены и количества операций сравнения - $I_C(A_1)$ и $I_C(A_2)$. Также приводятся отношения количества соответствующих операций $I_L(A_1)/I_L(A_2)$ и $I_C(A_1)/I_C(A_2)$, для наглядности.

Сравнивая теоретическую оценку вычислительной сложность двух алгоритмов, стоит заметить, что авторами, при проектировании алгоритма, было компенсировано уменьшение числа операций сравнения, посредством увеличения числа арифметических операций. Выигрыш в количестве операций сравнения MPAE над PAE наступает уже при представлении делимого в виде 2 слов (64 бита) и выше, а проигрыш в количестве арифметических операций становится менее заметным, с ростом двоичной длины.

Далее перейдем к сравнению производительности. Для этого авторами были запрограммированы PAE и предложенный MPAE на языке высокого уровня C++ в среде Microsoft Visual C++ 2010 Express Edition. Замеры времени производились, как среднее время выполнения 100 000 операций, на числах различной длины, с помощью вычислительных системах с процессором Intel Core i3 M350 (PC1) и Intel Xeon E5 – 2640 (PC2), под управлением ОС Windows 7 SP1 x86-64. В таблице 2 представлены экспериментальные оценки производительности PAE и MPAE, в зависимости от двоичной длины делителя и делимого. Так под ρ, понимается отношение числа используемых бит к числу зарезервированных в большом числе (делителе). Например, при $\rho = 0.1$ для представления числа зарезервировано 1024 бита, а используется лишь 102 бит. Двоичная длина делимого изменяется в диапазоне от 64 бит до 16 384 бит. Ширина машинного слова $w = 32$. Таблица с временными оценками разбита на блоки, каждый из которых описывает соответствующую часть результатов. Например, обозначение $64 бита \mid n_a = 2, 32 бита \mid n_b = 1$ указывает, что двоичная длина делимого составляет 64 бита (2 машинных слова), а делителя – 32 (одно машинное слово). Также в эксперименте учитывается ω - соотношение количества единиц делимого и делителя.

Замеры времени в мкс и мс выполнения операции деления приведены в таблице 2.

Таблица 2

Результаты экспериментальной оценки времени выполнения операции деления

	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2
ρ	0,1		0,2		0,4		0,5		0,8		0,9	
t	**мкс**											
$128 бит \mid n_a = 4, 64 бита \mid n_b = 2$												
ω	57/3	57/3	57/8	57/8	57/12	57/12	57/16	57/16	57/26	57/26	57/29	57/29
div2*	5,241	3,634	4,836	3,026	4,914	3,026	4,571	2,668	3,058	2,637	2,964	2,043
div2	8,112	5,772	7,707	5,211	7,036	4,899	6,52	4,43	4,898	3,728	4,399	3,089
t	**мс**											
$512 бит \mid n_a = 16, 256 бит \mid n_b = 8$												
ω	224/12	224/12	224/23	224/23	224/42	224/42	224/54	224/54	224/97	224/97	224/111	224/111
div2*	0,049	0,039	0,047	0,033	0,041	0,030	0,040	0,030	0,030	0,025	0,030	0,022
div2	0,098	0,072	0,092	0,062	0,080	0,057	0,075	0,055	0,058	0,043	0,053	0,038
$1024 бита \mid n_a = 32, 512 бит \mid n_b = 16$												
ω	445/18	445/18	445/46	445/46	445/84	445/84	445/93	445/93	445/176	445/176	445/199	445/199
div2*	0,156	0,129	0,160	0,120	0,139	0,117	0,131	0,108	0,113	0,096	0,108	0,083
div2	0,380	0,277	0,360	0,256	0,309	0,235	0,289	0,210	0,229	0,172	0,209	0,152
$4096 бит \mid n_a = 128, 2048 бит \mid n_b = 64$												

Окончание табл. 2

ω	1696/ 81	1696/ 81	1696/ 170	1696/ 170	1696/ 334	1696/ 334	1696/ 413	1696/ 413	1696/ 705	1696/ 705	1696/ 774	1696/ 774
div2*	2,371	1,732	2,169	1,653	1,996	1,514	1,919	1,419	1,575	1,139	1,31	1,129
div2	6,224	4,305	5,803	4,072	5,133	3,572	4,805	3,276	3,791	2,590	3,37	2,32
16184 *бита* | $n_a = 512$, 8132 *бита* | $n_b = 256$												
ω	6969/ 314	6969/ 314	6969/ 691	6969/ 691	6969/ 1405	6969/ 1405	6969/ 1742	6969/ 1742	6969/ 2787	6969/ 2787	6969/ 3137	6969/ 3137
div2*	35,007	25,054	32,65	24,617	30,467	21,715	28,876	21,076	23,743	17,378	21,559	15,896
div2	96,299	63,944	90,683	60,668	79,685	53,165	74,474	49,452	58,359	39,484	53,025	36,099

div2 - расширенный алгоритм Евклида;

div2* - модифицированный расширенный алгоритм Евклида;

n_a, n_b - количество машинных слов делимого и делителя, соответственно;

$32\,\text{бита} | n_b = 1$ - плотность заполнения единичными битами делимого и делителя.

Для наглядности, зависимость времени выполнения алгоритмов div2 и div2* от параметра $n_b = 1024$ для случая: $n_a = 1024$, $n_b = 512$, $i = \overline{0, n-1}$, представлена на рис. 3.

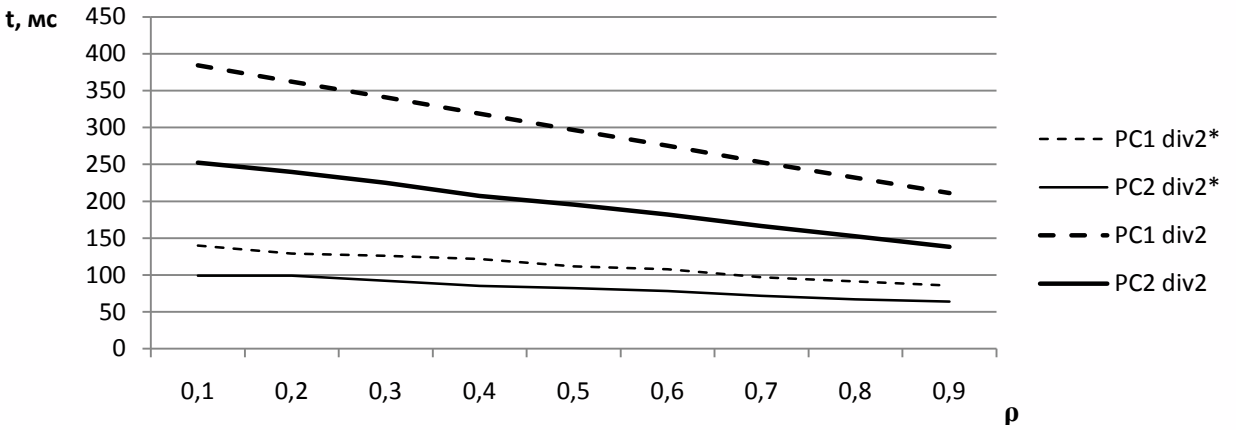

Рис. 3. Зависимость времени реализации расширенного алгоритма Евклида и предложенного расширенного алгоритма Евклида от параметра ρ

Выводы

По результатам выполнения исследований, авторами сформулированы следующие выводы:

1. Предложена классификация алгоритмов деления и приведения по модулю по различным критериям, которая позволяет сориентироваться в многообразии подходов к усовершенствованию известных алгоритмов.

2. Изучен классический РАЕ и сформулированы подходы к его усовершенствованию: использовать сравнения номеров старших битов, использовать знание закона изменения параметров уравнения Безу, для вычисления числа значимых машинных слов при выполнении элементарных операций (вычитания, сдвига и сравнения).

3. Сформулированные направления усовершенствования классического РАЕ нашли место в модифицированном РАЕ, предложенном авторами.

4. Проведенная оценка вычислительной сложности классического и модифицированного РАЕ, показала выигрыш в 1,24-196,91 раз при сравнении количества операций сравнения и выигрыш в 1,34-3,26 раз (начиная с числа 256 бит) при сравнении арифметических операций с ростом двоичной длины большого целого числа. Следует заметить, что аналитическая оценка сложности совпала с результатами моделирования. Удалось сократить количество вычислительно сложных операций сравнения за счет арифметических операций, которые быстрее выполняются современными процессорами и их можно распараллеливать, в отличии от операций сравнения.

5. Проведенная экспериментальная оценка производительности программной реализации РАЕ и МРАЕ, показала линейную зависимость времени выполнения от двоичной длинны делителя, которая также зависит от соотношения единичных битов ω и плотности делителя в диапазоне $\rho \in \{0.1, 0.2, \ldots, 0.9\}$.

6. Предложенные подходы оптимизации РАЕ для деления больших целых чисел, позволили повысить производительность программной реализации МРАЕ в 1,5-3 раза с ростом двоичной длины чисел.

7. При $\rho \in \{0.1, 0.2, \ldots, 0.5\}$, $n_a \in \{4, 8, 16, 32, 64\}$, $n_b \in \{2, 4, 8, 16, 32\}$ производительность РАЕ сильно ухудшается и прослеживается разница в производительности с МРАЕ до 3 раз. Когда длина делимого $n_a \in \{128, 256, 512, 1024\}$ и делителя

$n_b \in \{64,128,256,512\}$, тогда улучшение производительности в 2-3 раза, прослеживается для всех ρ.

8. Результаты экспериментальных оценок показали, что у PAE, для деления больших целых чисел, ограниченная область применения, несмотря на его универсальность и простоту, поскольку при уменьшении делителя (параметр ρ уменьшается) производительность сильно ухудшается, поскольку увеличивается количество операций сравнения. Такое поведение заметно даже на числах небольшой длины и требует применения специализированных алгоритмов.

9. Описанный MPAE для деления больших целых чисел, не ориентирован на многопоточное выполнение, что не позволило полностью реализовать потенциал современных многоядерных процессоров, однако такая возможность будет предложена в дальнейшем.

Литература

[1] Библиотека многократной точности GMP. URL: https://gmplib.org

[2] Bhattacharya J.: Rudiments of computer science. Kolkata.2010.ISBN: 978-93-80599-02-1

[3] Stehle, P.Zimmermann: A Binary Recursive Gcd Algorithm. Algorithmic Number Theory. LNCS Volume 3076, 2004, pp 411-425.

[4] L.Lhote, B.Vallée: Sharp Estimates for the Main Parameters of the Euclid Algorithm. LATIN 2006: Theoretical Informatics. LNCS Volume 3887, 2006, pp 689-702.

[5] V. Dupaquis, A. Venelli: Redundant Modular Reduction Algorithms. Smart Card Research and Advanced Applications. LNCS Volume 7079, 2011, pp 102-114.

[6] Уоррен Генри С. Алгоритмические трюки для программистов: Пер. с анг. – М.: Издательский дом «Вильямс» 2003. – 288 с.: ил. – Парал. тит. англ.

[7] ДСТУ 4145–2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що грунтується на еліптичних кривих. Формування та перевірка. – К.: Держстандарт України, 2002. –40с.

[8] IEEE P1363–2000: Standard Specifications for Public Key Cryptography. – 2000. – 206 p. URL: http://www.ieee.org

[9] P. Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. Proceedings CRYPTO'86, pp. 311-323.

[10] L. Hars. Long Modular Multiplication for Cryptographic Applications.//Cryptographic Hardware and Embedded Systems - CHES 2004.//LNCS Volume 3156, 2004, pp 45-61

[11] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. - CRC Press; First edition (December 16, 1996), Fifth Printing. - 2001. –780 p.

[12] P. Montgomery. Modular multiplication without trial division. Mathematics of Computation, 44(170):519–521, 1985.

[13] Ковтун В.Ю., Охрименко А.А. Метод повышения производительности операции приведения по простому модулю. – Информационные технологи и системы в управлении, образовании, науке: Монография / Под ред.. проф. В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014.– С. 204-220.

[14] J.-C. Bajard, S. Duquesne, M. Ercegovac. Combining leak–resistant arithmetic for elliptic curves defined over Fp and RNS representation. Publications Mathématiques de Besançon : Algèbre et Théorie des Nombres, 2013, pp.67-87. URL: http://pmb.univ-fcomte.fr

[15] M. Taschwer. Modular multiplication using special prime moduli. Kommunikationssicherheit im Zeichen des Internet. DuD-Fachbeiträge 2001, pp 346-371.

[16] N. Guillermin. A compressor for secure and high speed modular arithetic. Technical Report 354, Cryptology ePrint Archive (2011). URL: https://eprint.iacr.org/2015/193.pdf.

[17] W. Hasenplaugh, G. Gaubatz, V. Gopal. Fast Modular Reduction. Proceedings of the 18th IEEE Symposium on Computer Arithmetic, pp 225-229 . IEEE Computer Society Washington, DC, USA 2007.

[18] P. Giorgi, L. Imbert, T. Izard. Multipartite Modular Multiplication. RR-11024, 2011, pp.25.

[19] J. Guajardo, R. Blumel, U. Krieger, C. Paar: Efficient Implementation of Elliptic Curve Crypto systems on the TI MSP430x33x Family of Microcontrollers. Public Key Cryptography. LNCS Volume 1992, 2001, pp. 365-382.

[20] D. Hankerson, A. Menezes, and S.A. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004. p. 98.

[21] Chae Ho on Lim, Hyo Sun Hwang. Fast Modular Reduction With Precomputation. In Proceedings of Korea-Japan Joint Workshop on Information Security and Cryptology, Lecture.

[22] Z. Cao, R. Wei, Xiao dong Lin. A Fast Modular Reduction Method. IACR Cryptology ePrint Archive, 2014. URL: https://eprint.iacr.org/2014/040.pdf

[23] A. Bosselaers, R Govaerts, J Vandewalle. Comparions of three modular reduction functions. Advances in Cryptology—CRYPTO'93, 175-186.

[24] Jerome A. Solinas. Generalized Mersenne Numbers. http://cacr.uwaterloo.ca/techreports/1999/corr99-39.pdf

[25] D. E. Knuth. The Art of Computer Programming: Seminumerical Algorithms. Addison-Wesley Publishing Company, Reading, MA, 1981.

[26] H. Wu On Computation of Polynomial Modular Reduction, Tech. Rep., Centre of Applied Cryptographic Research, University of Waterloo, June 2000.

[27] D. Takahashi. A Parallel Algorithm for Multiple-Precision Division by a Single-Precision Integer. Large-Scale Scientific Computing. LNCS Volume 4818, 2008, pp 729-736.

[28] H.-Y. Lo, T.-Y. Chang, M.-C. Lee. Parallel Unidirectional Division Algorithms and Implementations. Journal of the Chinese Institute of

Engineers, Taylor & Francis, Volume 24, Issue 4, July 2001, Pages 487-496.

[29] Comba, P. G. Exponentiation cryptosystems on the IBM PC, IBM Systems Journal, Vol. 29, No. 4, December 1990. http://eprint.iacr.org/2012/482.pdf, http://eprint.iacr.org/2012/170.pdf

[30] Карацуба А., Офман Ю. Умножение многозначных чисел на автоматах // Доклады Академии Наук СССР. — 1962. — Т. 145. — № 2.

[31] M. Johnson, B. Phung, T. Shackelford, S. Rueangvivatanakij. Modular Reduction of Large Integers Using Classical, Barrett, Montgomery Algorithms.
URL: http://teal.gmu.edu/courses/ECE646/project/reports_2002/IP-1_report.pdf

[32] S. Gueron. Enhanced Montgomery Multiplication. Cryptographic Hardware and Embedded Systems — CHES 2002. Lecture Notes in Computer Science Volume 2523, 2003, pp 46-56.

[33] N. Emmart, C.-C. Weems: Parallel multiple precision division by a single precision divisor. HiPC 2011: 1-9.

[34] Алгоритмы деления Теодора Джебелина URL: https://gmplib.org/manual/Exact-Division.html, https://gmplib.org/manual/References.html#References.

[35] T. Jebelean. Using the Parallel Karatsuba Algorithmfor Long Integer Multiplication and Division. Euro-Par'97 Parallel Processing. LNCS Volume 1300, 1997, pp 1169-1172. URL: http://web-ng.info.uvt.ro/~tudor/Literature-Parallel/97-08.pdf.

[36] K. Hasselström. (2003). Fast Division of Large Integers A Comparison of Algorithm. Master's Thesis in Computer Science at the School of Computer Science and Engineering, Royal Institute of Technology, February.
URL: http://www.treskal.com/kalle/exjobb/original-report.pdf.

[37] Задірака В.К. Комп'ютерна арифметика багаторозрядних чисел // В.К. Задірака, О.С. Олексюк. — К.: 2003. — 264.

[38] Задирака В.К., Кудин А.М. Анализ стойкости криптографических и стеганографических систем на основе общей теории оптимальных алгоритмов, Journal of Qafgaz University. — N 30. —2010. —С. 49-58.

УДК 004.051 (045)

Гнатюк С.О., Ковтун В.Ю., Бердник О.М., Ковтун М.Г. Підходи для підвищення продуктивності розширеного алгоритму Евкліда для ділення великих чисел подвійної точності на великі числа одинарної точності

Анотація. *Криптографічні перетворення з відкритим ключем мають значну обчислювальну та просторову складність. У зв'язку з цим, актуальною науково-технічною задачею є підвищення продуктивності таких перетворень. У роботі розглядаються підходи до підвищення продуктивності операції ділення великих цілих чисел подвійної точності на великі цілі числа одинарної точності на основі розширеного алгоритму Евкліда. До таких походів відносяться: оперування відмінними від нуля машинними словами, які найбільш часто використовуються в операціях порівняння, зсуву вліво і вправо, додавання і віднімання великих чисел; використання наближеного порівняння великих цілих чисел, без необхідності послівного порівняння машинних слів за рахунок порівняння номерів старших бітів цих чисел; знання закону зміни параметрів рівняння Безу – використовується в попередніх двох підходах. Запропоновані підходи успішно реалізовані в модифікованому алгоритмі. Для порівняння проводилися експерименти над числами, за умови, що двійкова довжина діленого в два рази перевищує двійкову довжину дільника для різного співвідношення заповненої та загальної двійкової довжини великого числа. Модифікований алгоритм показав кращу продуктивність в 1,5-3 рази, з ростом двійкової довжини діленого і дільника.*

Ключові слова*: ділення, залишок від ділення, великі цілі числа, розширений алгоритм Евкліда.*

Gnatyuk S., Kovtun V., Berdnik O., Kovtun M. Approaches to performance increasing of extended Euclidean algorithm for double precision division on single precision large integers

Abstract. *Public key cryptography has significant processing and spatial complexity. In this regard, relevant scientific and technical challenge is to improve the performance of such transformations. In this paper proposed approaches of large integer's division operation with double precision on single precision based on the Extended Euclidean Algorithm. These approaches include handling non-zero machine words in the most frequently used operations; using an semi-exact comparison of large integer, without for-word comparison; knowledge of the of Bézout equation parameters changing. These approaches are implemented in the modified algorithm, which has been software implemented. In experiments were compared numbers with double binary size dividend and single binary size divider with ratios for various filled and total binary length. Modified algorithm showed higher performance in 1.5-3 times, with dividend and divisor increasing binary length.*

Key words: *division, remainder in division, large integer's modular reduction, extended Euclidean algorithm.*

Отримано *10 лютого 2015 року*, затверджено редколегією *27 лютого 2015 року*