УДК 004.4 (043.2)

Гамаюн В. П., д-р техн. наук

КВАЗИГРАФИЧЕСКИЙ МЕТОД ВЫЧИСЛЕНИЯ ОСТАТКА ПО МОДУЛЮ

Институт кибернетики им. В. М. Глушкова НАН Украины

Рассмотрена реализация центральной операции модульной арифметики – вычисления остатка по модулю для многоразрядных чисел. Арифметико-алгоритмический аппарат основан на разрядно-логарифмическом представлении операндов и базовых операциях маскирования-сравнения.

Введение. Постановка задачи

Для решения множества задач вычислительной, прикладной, дискретной математики применяется технология вычислений, которая использует алгоритмы разных операций над многогоразрядными числами. При реализации многоразрядной арифметики применяемые алгоритмы основаны на математических законах арифметики и алгебры, методах вычислительной математики [1 – 4]. Например, вычисление остатка по модулю является центральной операцией модульной арифметики, применяемой в различных приложениях, в том числе в системах защиты информации [1 – 4]. Эффективность по сложности алгоритма вычисления остатка по модулю определяет, наряду с другими алгоритмами, основные показатели системы защиты информации. Известный алгоритм Монтгомери определяет остаток за k^2+k вычислений (одноцифровых умножений) при выполнении ряда необходимых предвычислений (где k – разрядность числа) [3].

В компьютерной арифметике, наряду с традиционными методами и алгоритмами возможно применение такого набора алгоритмов обработки, которые базируются на «подобии» геометрической топологии представления данных – операндов, так называемые квазиграфические методы [5]. Целью настоящей работы является представление результатов по методам квазиграфического вычисления остатка, сущность которого состоит в следующем: для получения результата вычисления остатка по модулю, возможно применять в качестве базовых операций компьютерной арифметики не операции сложения – вычитания, а маскирования – сравнения.

Метод вычисления остатка с применение базовых операций маскирования – сравнения

Квазиграфические методы реализуются в компьютерной среде с получением таких же результатов преобразования, как и применении счета [5].

Деление двоичных кодов возможно рассматривать как исключение из кода делимого A разрядов делителя B, которые сдвинуты относительно позиций исключаемых разрядов.

Наиболее «успешной» операцией вычитания при делении считается та, при которой разность (остаток) равен нулю: при этом деление выполняется за минимальное количество тактов. Например, при делении числа A=11011101 на B=1101 следует дважды исключить код делителя из делимого для получения результата:

A 1101 1101 A^1 0000 1101
B 1101 1101
A^2 0000 0000: результат 10001, остаток 0000.

Как видно из примера для такого «быстрого» деления следует делимое преобразовывать к виду, при котором есть соответствие между разрядами уменьшаемого и вычитаемого. Затем применяя сравнение и маскирование получаем либо нулевой, либо определенный остаток. При этом никаких традиционных действий связанных с классически вычитанием (микрооперация определения заема) не

выполняется. Поэтому, если в классических алгоритмах компьютерной арифметики при вычислении остатка с использованием деления применяется вычитание, то тогда возможно применить сравнение и маскирование или вычеркивание тех разрядов, которые совпадают со значением модуля (делителя).

Например – вычисление остатка по модулю 1101101 из числа 1111111 равно, после вычеркивания совпадающих разрядов, следующему значению

1111111
1101101 0010010 – остаток

по модулю.

Так как значение частного не фиксируется, то получен правильный результат операции вычисления остатка. В данном примере число было подобрано равным по разрядности модулю. Рассмотрим пример вычисления остатка по модулю 1101101 из другого числа, разрядность которого больше разрядности модуля – 1 1111111. При применении кода «–1» старшую единицу числа преобразуем как 10000000 – 1111111+1

Перепишем исходное число как

1111111+1
1111111

и применим к нему вычеркивание для первого числа 1111111+1,

1101101

и для второго числа 1111111

1101101.

В результате вычеркивания получаем два кода – 0010010+1 и 0010010 , которые преобразуем в результат применением сдвига (так как коды одинаковые) и приписыванием (подстановка) единицы: 0100101. Так как полученное значение меньше заданного модуля, то 0100101 является результатом. Каждая использованная операция по времени определяется как одноцифровое умножение (задержка на 2-х входовом элементе И). Для вычисления остатка по модулю использовано операций:

– преобразование операнда в код «-1»;

– 2 операции маскирования – сравнения;

– сдвиг с добавлением (подстановкой) единицы младшего разряда.

Для выполнения вычисления остатка по модулю следует выполнять (реализовывать) правило преобразования разряда кода в последовательность единиц, выполнять сравнение – маскирование (вычеркивание) совпадающих разрядов и сдвиг результата с подстановкой единицы в младший разряд. Выполненные действия аналогичны некоторым операциям, выполняемым при графическом преобразовании объекта (нанесение маски – вычеркивание), поэтому предлагаемый метод определим как квазиграфический. Значение исходного числа может быть различным и поэтому рассмотрим общий алгоритм вычисления остатка.

На начальном этапе выполним сравнение - маскирование – вычеркивание из исходного кода значений, соответствующих коду модуля. Для этого определим межразрядные расстояния или межразрядные интервалы между ненулевыми разрядами (значащими единицами) в исходном числе $\{r1_i\}$ и заданном модуле $\{m_i\}$.

Если для представления данных применяется разрядно-логарифмическое представление, то такая процедура может быть выполнена за один такт при параллельном исполнении. При получении результатов $\{r1_i\}$, $\{m_i\}$ выполняется сравнение массива $\{r1_i\}$ с массивом $\{m_i\}$. В массиве $\{r1_i\}$ последовательно, начиная со старших разрядов выделяется столько разрядов, сколько в определенном (заданном) модуле и сравнивается такая комбинация с комбинацией межразрядных интервалов в модуле. При совпадении группа разрядов из исходного числа удаляется, как не определяющая никакого остатка. При несовпадении формируется следующая группа массива $\{r1_i\}$ способом сдвига относительно предыдущей группы на один разряд влево (в сторону старших разрядов).

Если анализируемая группа разрядов содержит только единицы, то в этом случае возможно также определить остаток без выполнения арифметических операций – остаток будет равен числу, ненулевые разряды в котором определяются нулевыми разрядами модуля. Например для модуля 1101 группа разрядов исходного числа 1111 определяет остаток 0010. Аналогично возможно определить условие, что группа 1110 по такому же модулю (1101) дает остаток 0001. Неизменным остается условие невыполнения никаких арифметических операций при определении остатка.

После вычеркивания из исходного числа «модульных единиц» возможно несколько вариантов результата:

– вычеркнуты все единицы (остаток равен нулю);

– в исходном числе остались значащие единицы, но их разрядные значения меньше модуля (получено окончательное значение остатка);

– в исходном числе остались значащие единицы и их разрядные значения больше модуля (требуется дальнейшее получение окончательного значения остатка);

Значащие разряды, с большим весом, чем старший разряд модуля, преобразуются по правилу формирования кода «–1» (минус единица). Преобразование производится последовательно, начиная с разряда с большим весом, и может выполняться не до младшего разряда, а до группы следующих значащих разрядов. Такое правило следует из результатов моделирования алгоритма и определяет меньшее количество шагов при получении результата. Полученный код преобразования одного разряда, после процедуры вычеркивания (которую можно упростить), объединяется с оставшимися разрядами исходного числа и далее выполняется операция вычеркивания с предварительной процедурой упорядочения и приведения подобных в объединенной последовательности разрядов. Операции приведения подобных и сортировки возможно выполнять одновременно, так как объединяемые массивы упорядочены, приведение подобных редко приводит к распространению переноса. Как определяет моделирование время на сортировку и приведение оценивается в количество шагов, равное среднему количеству элементов в двух объединяемых последовательностях.

После получения нового варианта результата вычеркивания действия по алгоритму аналогичны.

Рассмотрим пример вычисления 7^{10}mod13. Используем разрядно-логарифмическое представление для операндов [6]: РЛ код 7^{10} равен 7^{10} = 28.23.22.20.18.17.13.12.11.9.7.6.5.4.0., а код mod 13 = 3.2.0.

Межразрядные расстояния для 7^{10} определим как 5-1-2-2-1-4-1-1-2-1-1-1-3 и для mod13= 1-2 . Применяя последнюю комбинацию (1-2) для анализа получаем, что на первом этапе вычеркивания исключаются группы разрядов

7^{10} =28.23.22.20.18.17.13.12.11.9.7.6.5.4.0. (подчеркнутые разряды),

23.22.20.,12.11.9. и 7.6.5.4., после вычеркивания которой остается 5.

Таким образом, на первом этапе вычеркиванием получаем из исходного числа следующий код

28.18.17.13.5.0.

Старший значащий разряд равный 28. преобразуем в код «–1» до разряда «17» включительно, что дает следующий результат

27.26.25.24.

23.22.21.20.

19.18.17.16.+16

После вычеркивания модуля из этой последовательности, получаем 25.21.17.16. Далее возможно преобразовать еще и 25. в код «–1». При преобразовании 25. до «17» включительно применим априорные знания о модуле и окончательно получим 22.18.17. Объединив полученные коды, сформируем последовательность

18.17.13.5.0. 21.17.16.22.18.17.

Выполнив приведение подобных и упорядочивая данные, получаем новый массив

22.21.19.18.17.16.13.5.0.

Далее, применяя процедуру маскирования – вычеркивания, исключаем группы разрядов 22.21.19. и 18.17.16.(15.15.) и формируем новый массив 15.13.5.0. Преобразуя значение старшего разряда 15. по коду «–1» в последовательность меньших разрядов, определим новую последовательность

14.13.12.11.10.13.10.5.0.

Исключение значений разрядов 14.13.11. и 13.12.10. приводит к новому массиву. В остатке получается последовательность 10.5.0.

Далее преобразуется 10. в коды 9.8.7.6.6. и формируем новую последовательность 7.6.5.0., после применения к которой маскирования – вычеркивания получаем 4.0. Дальнейшее преобразование 4. дает результат 3.2.1.0.+0. После маскирования – вычеркивания получаем 1.0., что вместе с 0. определяет конечный результат 2. в РЛ представлении или 4 в десятичной системе счисления.

Таким образом, в результате преобразований в код «–1» и выполнения операций сравнения – маскирования получаем результат $7^{10} \bmod 13 = 4$.

Определим метод вычисления остатка с вычеркиванием следующим образом:

– задаем число X и модуль;

– определим межразрядные расстояния между значащими (ненулевыми) разрядами исходного числа и модуля в виде двух последовательностей $\{r1_i\}^1$ и $\{m_i\}$.

– для последовательности $\{r1_i\}^1$ применяем сравнение с $\{m_i\}$ начиная со старших разрядов, при совпадении выполняем исключение соответствующих разрядов в $\{r1_i\}^1$, при несовпадении сдвиг в сторону младших разрядов;

– результат предыдущего этапа возможен в вариантах:

а) вычеркнуты все единицы (остаток равен нулю);

б) в исходном числе остались значащие единицы, но их разрядные значения меньше модуля (получено окончательное значение остатка);

в) в исходном числе остались значащие единицы и их разрядные значения больше модуля (требуется дальнейшее получение окончательного значения остатка);

– преобразуем старшую значащую единицу в код «–1» с одновременным вычеркиванием по модулю, сформируем новый массив $\{r1_i\}^2$ объединением полученных кодов и тех кодов разрядов, которые не преобразовывались;

– применим вычеркивание к полученному массиву (начиная со второго этапа) и повторяем последующие этапы до получения вариантов а) или б).

Граф алгоритма по предложенному методу для вычисления остатка из числа X по модулю представлен на рис. 1.

По рассмотренному методу возможно разработать различные алгоритмы вычисления остатка. В каждом конкретном варианте необходимо учитывать значение модуля, значение межразрядных интервалов, по которым формируется основное правило сравнения-вычеркивания-маскирования.

Для модуля $13_{10}=1101_2$ правило проверки определяется как

$R1(i)=1$ *and* $R1(i+1)=2$ и

$R1(i)=1$ *and* $R1(i+1)=1$ *and* $R1(i+2)=1$.

Для другого модуля правило проверки изменяется соответственно.

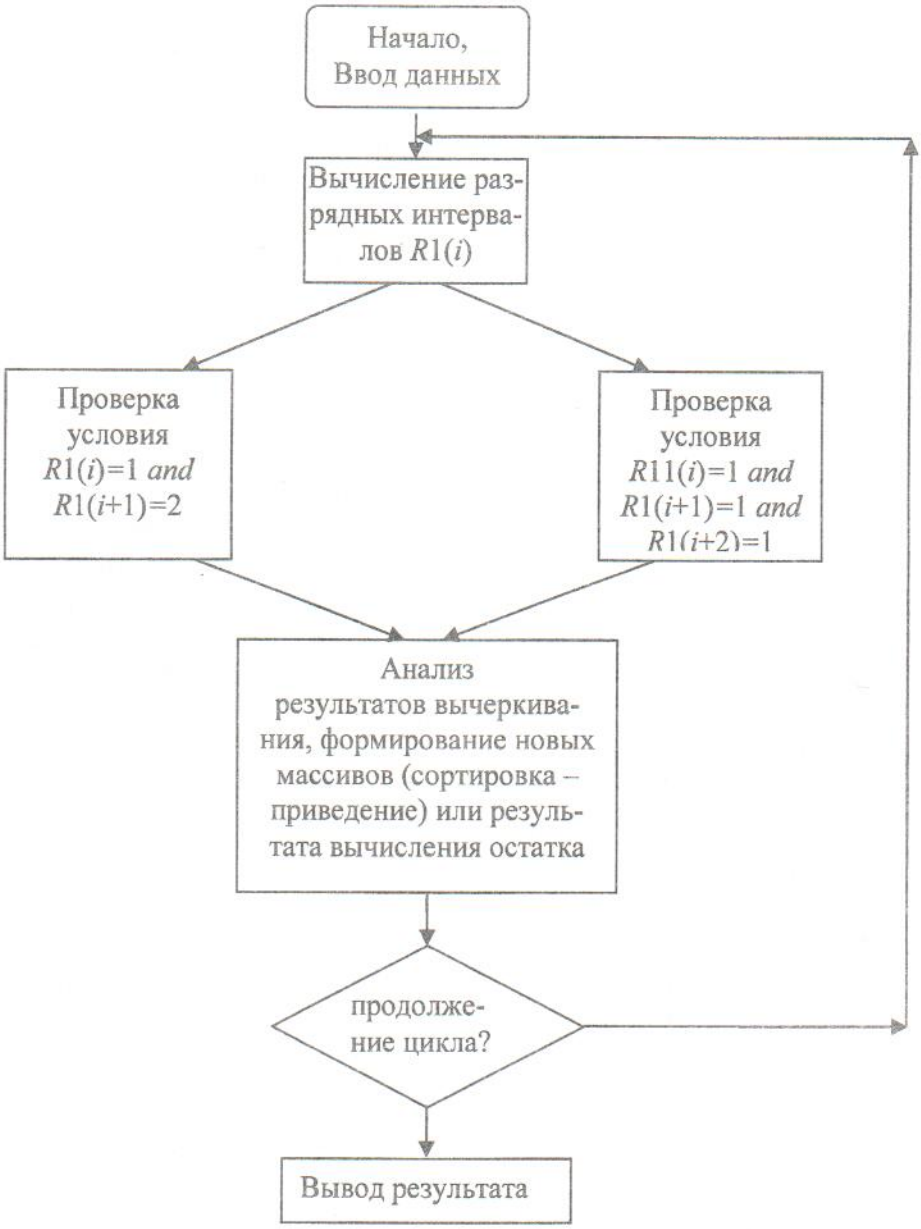

Рис. 1. Граф вычисления остатка числа по модулю 13

Алгоритм вычисления остатка по модулю 13 может быть следующим:

1) ВВОД числа X с количеством ненулевых разрядов qx и модуля M в разрядно-логарифмической форме;

2) Вычисление межразрядных интервалов в X и M:

Для i=1,…, qx–1 Выполнить

$R1(i)= X(i)–X(i+1)$

3) Для i=1 ,…,qx–1 Выполнить

Если $R1(i)$=1 и $R1(i+1)$=2 или $R1(i)$=1 $R1(i+1)$=1 $R1(i+2)$=1 То исключение в X разрядов $X_i X_{i+1} X_{i+2} X_{i+3}$ или $X_i X_{i+1} X_{i+3}$.

Иначе сдвиг $R1$ влево

4) Если X =0 «Останов» Иначе

$X <= M$ то результат в X Иначе

$X(1)_i$ заменить кодом «–1»

5) Объединить X с новым кодом (представлением) $X(1)_i$

6) Выполнить приведение и сортировку по убыванию, перейти на 2.

Программа вычисления остатка по квазиграфическому методу из числа X по модулю 13 следующая:

```
5 CLS
7 c = 0
10 DIM x(50), m(10), mr(10), r(50), r1(50),
   r2(50)
12 INPUT qx
16 FOR i = 1 TO qx: INPUT x(i): NEXT
   ввод числа
```

```
20 FOR i = 1 TO qx – 1: c = c + 1 вычисление разности
22 r1(c) = x(i) – x(i + 1): NEXT между кодами разрядов
23 l = 0: k1 = 1 начало вычеркивания-маскирования
24 FOR i = k1 TO c – 1
26 IF r1(i) = 1 AND r1(i + 1) = 2 THEN 28 ELSE 38
28 k1 = k1 + 3:
36 GOTO 24
38 IF r1(i) * r1(i + 1) * r1(i + 2) = 1 THEN 40 ELSE 52
40 l = l + 1: r(l) = x(i + 2)
42 k1 = k1 + 4
50 GOTO 24
52 l = l + 1: r(l) = x(i)
62 k1 = k1 + 1
64 NEXT
65 FOR i = 1 TO qx – k1 + 1: l = l + 1: r(l) = x(k1 + i – 1): NEXT дописывание кодов из конца записи разностей
67 FOR i = 1 TO l: PRINT r(i); : NEXT контрольная печать
68 PRINT
70 h = 0 начало вычисления – преобразования по старшему коду
72 h = h + 1: r2(h) = r(1) – 3 ( 3 – позиция нуля в модуле 13)
74 h = h + 1: r2(h) = r2(h – 1) – 4 (4 – разрядная разность нулей в модуле 13 после первого кода)
75 IF r2(h) <= r(3) THEN h = h + 1: r2(h) = r2(h – 1) – 1: GOTO 76 ELSE 74 – условие разложения до значения 3 цифры кода – может меняться
76 IF r2(h) <= 3 THEN 104 ELSE 78 – условия остановки преобразования разложения
78 PRINT: FOR i = 1 TO h: x(i) = r2(i): PRINT x(i); : NEXT
79 FOR i = 2 TO l: x(h + i – 1) = r(i): PRINT x(h + i – 1); : NEXT
80 qx = h + l – 1
82 REM приведение – сортировка
84 FOR i = 1 TO qx – 1
85 FOR j = i + 1 TO qx
86 IF x(i) = x(j) THEN x(i) = x(i) + 1 ELSE 90
87 FOR t = j TO qx – 1: x(t) = x(t + 1): NEXT: qx = qx – 1: GOTO 84
90 NEXT
92 NEXT
94 FOR i = 1 TO qx – 1
96 IF x(i) < x(i + 1) THEN gf = x(i): x(i) = x(i + 1): x(i + 1) = gf: GOTO 94 ELSE 98
98 NEXT
100 PRINT: FOR i = 1 TO qx: PRINT x(i); : NEXT: c = 0: GOTO 20
104 STOP: c = 0: GOTO 20
168 END
```

Выводы

Моделирование алгоритмов вычислении остатка по модулю, разработанных на основе квазиграфических методов подтверждает корректность предположения применения новых, альтернативных походов к используемому арифметико-алгоритмическому базису компьютерной арифметики многоразрядных чисел. Рассмотренный метод при реализации может быть распараллелен на этапах сравнения – маскирования – вычеркивания, определения межразрядных расстояний, что делает такой метод еще боле конкурентноспособным.

Список литературы

1. *Задірака В. К., Олексюк О. С.* Комп'ютерна арифметика багаторозрядних чисел: Наукове видання. – К.: 2003. – 264 с.

2. *Анісімов А. В.* Алгоритмічна теорія великих чисел: Академперіодика. – К.: 2001. – 153 с.

3. *Montgomery P. L.* Modular Multiplication Without trial division // Mathematic of Computation. – 1985. – 44. – №170. – P. 519-521.

4. *Анисимов А. В.* Быстрое прямое вычисление модулярной редукции // Кибернетика и системный анализ. – 1999. – № 4. – С. 3-12.

5. *Гамаюн В.П.* Квазиграфический метод преобразования многорядного кода // Комп'ютерні засоби, мережі та системи: зб.наук. праць. – К.: Ін-т кібернетики ім. В. М. Глушкова НАНУ 2002. – №1. – С. 53-57.

6. *Гамаюн В. П.* Макрооператорные методы вычисления многоместных произведений // Микропроцессорные системы и их применение. – К.: Ин-т кибернетики им. В.М. Глушкова АН УССР, 1990. – С.23-28.