Н.П. БРУСЕНЦОВ

АЛГОРИТМЫ ДЕЛЕНИЯ ДЛЯ ТРОИЧНОГО КОДА С ЦИФРАМИ 0, 1, −1

Правила выполнения в троичном коде с цифрами 0,1, −1 операций сложения, вычитания и умножения нередко приводят для иллюстрации замечательных арифметических свойств этого кода. Вместе с тем, операция деления обычно упускается из вида, хотя как раз деление в данном коде наиболее своеобразно. Это своеобразие, как и особенности других арифметических операций в данном коде, обусловлено наличием в нем цифр положительного и отрицательного веса.

Мы кратко рассмотрим отличия операции деления в троичном коде с цифрами 0, 1, −1 от операции деления в коде с неотрицательными цифрами и дадим примеры алгоритмов ее реализации. Троичный код с цифрами 0, 1, −1 будем называть (троичным) симметричным кодом.

Одно из очевидных отличий деления в симметричном коде состоит в том, что если для кода с неотрицательными цифрами в случае равенства троичных порядков делимого и делителя порядок частного равен 0 или −1, то для симметричного кода при этом уже условии он может быть равен 0, или −1, или 1.

Например:

$$1\bar{1} : 1\bar{1} = 1{,}0$$
$$1\bar{1} : 11 = 0{,}111\ldots$$
$$11 : 1\bar{1} = 1\bar{1}{,}0$$

Таким образом, если в троичном коде с неотрицательными цифрами при равенстве порядков делимого и делителя деление начинают с определения цифры в разряде 3^0 частного, то в симметричном коде необходимо начинать с определения цифры в разряде 3^1.

Другое отличие заключается в несовпадении критериев того, является или не является нулем определяемая цифра частного. В коде с неотрицательными цифрами определяемая цифра частного есть нуль, если делимое меньше делителя по абсолютной величине. В случае симметричного кода возможны два варианта рассматриваемого критерия:

1) цифра частного есть нуль, если делимое не больше 3/2 делителя по абсолютной величине;

2) цифра частного есть нуль, если делимое меньше 3/2 делителя по абсолютной величине.

В тех случаях, когда частное не имеет точного представления в троичном коде, и точное представление возможно лишь с использованием величины вида $1/2\cdot 3^n$, где n − некоторое целое число, приближенное значение частного получается в зависимости от используемого варианта критерия либо с избытком, либо с недостатком. Например, вычисляя частное 3:2 с точностью до второго знака после запятой соответственно с использованием первого и второго вариантов критерия, получим:

$$1)\ 10 : 1\bar{1} \simeq 1{,}11 \quad (=3/2-1/2\cdot 3^{-2}),$$
$$2)\ 10 : 1\bar{1} \simeq 1\bar{1}{,}\bar{1}\bar{1} \quad (=3/2+1/2\cdot 3^{-2}).$$

Первый вариант критерия заслуживает предпочтения, потому что в ряде случаев, подобных данному примеру, приводит более компактной записи частного с требуемой точностью. Далее всюду, за исключением алгоритмов *division* 4, используется первые вариант критерия, т. е. считается, что определяемая цифра частного есть нуль, если $2|c| \leq 3|d|$, где c – делимое или в i раз утроенный i-й остаток, d – делитель.

Приводимые ниже в виде АЛГОЛ-процедур алгоритмы деления в троичном симметричном коде предназначены для получения частного при условии, что значения делимого и делителя удовлетворяют требованию, которое предъявляется к мантиссам нормализованных троичных чисел [1], т. е. что абсолютные величины этих значений $|m|$ принадлежат интервалу $0{,}5 < |m| < 1{,}5$ или $m = 0$.

Алгоритмы *division* 1, *division* 2, *division* 3 выдают результат в виде пары чисел – нормализованной мантиссы e и порядка q, – соответствующих представлению частного в виде $e \times 3^q$. Точность, с которой вычисляется мантисса e, задает целочисленный параметр k, указывающий количество троичных разрядов в представлении e. Поскольку мантисса е предполагается нормализованной, то в ее разряде 3^1 должна содержаться цифра нуль, а в разряде 3^0 – цифра, отличная от нуля. Но, как указывалось выше, при равных порядках делимого и делителя троичный порядок частного может быть от −1 до 1, т. е. непосредственным результатом деления может быть мантисса, содержащая цифру, отличную от нуля в разряде 3^1, или содержащая нули в разрядах 3^1 и 3^0 одновременно. В обоих случаях требуется нормализация результата, однако в трех рассматриваемых алгоритмах частное сразу получается нормализованным.

В алгоритмах *division* 1 и *division* 2 это достигнуто применением стандартной процедуры *prepare*, которая заранее определяет значение порядка q и корректирует масштаб делимого с так, что вычисляемая затем мантисса частного е получается нормализованной. При этом собственно деление всегда начинается с определения цифры разряда 3^0 мантиссы частного. В алгоритме *division* 3 функции, выполняемые процедурой *prepare*, реализованы непосредственно.

Алгоритм *division* 1 является наиболее простым и наименее эффективным. Он состоит из цикла, вытолняющегося столько раз, сколько требуется получить разрядов мантиссы частного. Каждый раз производится определение знака очередного остатка c и проверяется условие $2|\mathrm{c}| > |\mathrm{d}|$. Если это условие выполнено, то искомой цифрой частного является 1 в случае, когда знак остатка совпадает со знаком делителя, и −1 в случае, когда знаки не совпадают. В первом случае делитель вычитается из остатка, во втором прибавляется к остатку. При этом поскольку алгоритм работает с удвоенным остатком, то каждое вычитание или прибавление производится дважды. Если же условие $2|\mathrm{c}| > |\mathrm{d}|$ не выполнено, то искомая цифра частного есть нуль, и ни вычитание, ни прибавление делителя не производится. Перед каждым повторением цикла текущее значение остатка утраивается.

Алгоритм *division* 2 отличается от первого тем, что в нем определение знака остатка производится не при каждом повторении цикла, а только в тех случаях, когда остаток изменится в результате вычитания или прибавления делителя. Кроме того, в этом алгоритме повторения цикла автоматически прекращаются, если остаток стал равным нулю.

Алгоритм *division* 3 еще более усложнен с целью увеличить скорость. Убыстрение достигнуто в основном за счет того, что в циклической части условие $g > 0$, проверка которого связана с необходимостью анализа, вообще говоря, всех разрядов слова g, заменено условием $abs(c) > 1{,}5$, для проверки которого достаточно проанализировать в слове с только один разряд 3^1. При этом в тех циклах, в которых определяемой цифрой частного оказывается нуль, над остатком не производится никаких операций кроме сдвига влево, т. е. увеличения в три раза.

Алгоритм *division* 4 вообще не содержит в цикле условий, проверка которых связана с анализом более чем одного разряда остатка.

Но этот алгоритм не всегда определяет цифру частного сразу точно: первоначально определенная цифра может быть затем уточнена по мере определения цифр младших разрядов, причем требуется, чтобы регистр частного был сумматором. В отличие от предыдущих алгоритмов, которые характеризуются ошибкой, не превышающей половины единицы младшего разряда, т. е. выдают правильно все k цифр частного, алгоритм *division* 4 работает с ошибкой, достигающей единицы младшего разряда. Частное он выдает в виде ненормализованной мантиссы e, содержащей не менее k значащих цифр.

procedure *prepare* (*c, d, e, q, k, sd*);
  value *k*; **integer** *k, q, sd*; **integer array** *e*; **real** *c, d*;
  comment *Оператор stop прерывает процедуру, если* $d = 0$;
  begin integer *p*;
    if $d > 0$ **then** $sd := 1$ **else if** $d < 0$ **then** $sd := -1$ **else**
    stop;
    $d := abs\ (d)$; $c := c + c$;
    for $p := 1$ **step** 1 **until** k **do**
    $e[p] := 0$;
    if $abs(c) - d \times 3 > 0$ **then begin** $q := 1$; $d := d \times 3$ **end**
    else if $abs(c) - d > 0$ **then** q := 0
    else begin $q := -1$; $c := c \times 3$ **end**
  end *prepare*;

procedure *division* 1 (*c, d, e, q, k*);
  value *k*; **integer** *k, q*; **real** *c, d*; **integer array** *e*;
  comment *c – делимое, d – делитель, принадлежат интервалу* $0.5 < |m| < 1.5$ *или* $m = 0$. *Частное представляется посредством мантиссы e* $[1 : k]$ *и порядка q в виде* $3^q \sum_{p=1}^{k} e_p \times 3^{1-p}$.
  Оператор stop прерывает процедуру, если $d = 0$;
  begin integer *p, sc, sd*; **real** *g*;
    prepare (*c, d, e, q, k, sd*);
    for $p := 1$ **step** 1 **until** k **do**
    begin if $c > 0$ **then** $sc := 1$ **else** $sc := -1$; $g := c \times sc - d$;
    if $g > 0$ **then begin** $e\ [p] := sc \times sd$; $c := (g - d) \times sc$
    end;
    $c := c \times 3$
    end
  end *division* 1;

```
procedure division 2 (c, d, e, q, k);
      value k; integer k, q; real c, d; integer array e;
      comment см. комментарий division 1;
      begin integer p, sc, sd; real g;
         prepare (c, d, e, q, k, sd);
         if c > 0 then sc := 1 else sc := −1;
         for p := 1 step 1 until k do
         begin g := c × sc − d;
            if g > 0 then
            begin e [p] := sc × sd; c := (g − d) × sc;
               if c > 0 then sc := 1
               else if c > 0 then sc := −1
               else go to exit
            end; c: = c × 3
         end;
exit:
      end division 2;

procedure division 3 (c, d, e, q, k);
      value k; integer k, q; real c, d; integer array e;
      comment см. комментарий division 1;
      begin integer p, s1, s2, sd; real d2, d3;
         if d > 0 then sd := 1 else if d < 0 then sd := −1 else
         stop;
         d := abs (d); d2 := d + d; d3 := d + d2; c := c + c;
         for p := 1 step 1 until k do
         e [p] := 0;
         if abs (c) > 1.5 then q := 0 else begin q := l; c := c × 3 end;
         if abs (c) > d3 then begin p := 1; q := q + 1 end else
         p := 0;
L2:      if c > 0 then s1 := 1 else s1 := 1;
         c := c − d3 × s1 ;
         if c > 0 then s2 := 1 else s2:= −1;
         if s1 = s2 then begin c := c − d3 × s1; e [p] := s1 × sd;
         go to L0 end;
         c := c − d × s2;
L1:      p := p + 1; e [p] := s1 × sd; if p = k ∨ c = 0 then go to
         exit;
         c := c × 3;
         if abs (c) > 1.5 then
         begin if c > 0 then s1 := 1 else s1 := −1;
            c := c − d2 × s1; go to L1
         end;
L0:      p := p + 1; if p = k ∨ c = 0 then go to exit;
         c := c × 3;
         if abs (c) > 1.5 then go to L2 else go to L0;
exit: end division 3;
```

```
procedure division 4 (c, d, e, k);
      value k; integer k; real c, d, e;
      comment c – делимое, d – делитель, принадлежат интервалу
      0.5 < |m| < 1.5 или m = 0, e – частное, вычисляемое с ошибкой,
      не превосходящей 0.5 × 3 ↑ (–k...) Оператор stop прерывает
      процедуру, если d = 0;
      begin integer p, s1, s2, sd;
         if d > 0 then sd := 1 else
         if d < 0 then sd := –1 else stop;
         e := 0; p := 0; d := abs (d);
L2:      if c > 0.5 then s1 := 1 else
         if c < –0.5 then s1 := –1 else go to L0;
L1:      c := c – d × s1; e := e + s1 × 3 ↑ (–p);
         if c > 0.5 then s2 := 1 else
         if c < –0.5 then s2 := –1 else go to L0;
         if s1 ≠ s2 then
         begin p := p + 1; if p > k + 1 then go to exit;
            s1 := s2; c := c × 3
         end;
         go to L1;
L0:      p := p + 1; if p > k + 1 ∨ c = 0 then
         go to exit; c := c × 3; go to L2;
exit:    e := e × sd;
      end division 4;
```

Автор выражает благодарность Н.Б. Лебедевой, испытавшей данные алгоритмы на ряде примеров и указавшей на замеченные при этом ошибки.

ЛИТЕРАТУРА

1. Брусенцов Н.П., Маслов С.П. и др. Малая цифровая вычислительная машина «Сетунь». Изд-во Моск. ун-та, 1965.