Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронно-вычислительных средств

М. В. Качинский, В. Б. Клюс, А. Б. Давыдов

АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ

Рекомендовано УМО по образованию
в области информатики и радиоэлектроники
в качестве учебно-методического пособия для специальности
1-40 02 02 «Электронные вычислительные средства»

Минск БГУИР 2014

УДК 004.315(076)
ББК 32.973.26-04я73
К30

Рецензенты:
кафедра информационных систем и технологий
Белорусского национального технического университета
(протокол №10 от 3.06.2013 г.);

заведующий кафедрой информатики учреждения образования
«Минский государственный высший радиотехнический колледж»,
кандидат технических наук, доцент Ю. А. Скудняков

Качинский, М. В.

К30 Арифметические основы электронных вычислительных средств : учеб.-метод. пособие / М. В. Качинский, В. Б. Клюс, А. Б. Давыдов. – Минск : БГУИР, 2014. – 64 с. : ил.

ISBN 978-985-543-001-9.

Рассматриваются вопросы представления числовой информации в ЭВС, выполнения операций двоичной арифметики с фиксированной и плавающей запятой, десятичной арифметики. Предназначено для изучения дисциплины «Основы проектирования электронных вычислительных средств».

УДК 004.315 (076)
ББК 32.973.26-04я73

ISBN 978-985-543-001-9

Содержание

ВВЕДЕНИЕ

Для современного этапа развития вычислительной техники характерным является то, что в основе построения электронных вычислительных средств (ЭВС) лежит сравнительно небольшое число принципов, а на практике применяется большое число различных технических решений, разнообразная элементная база, имеющая значительное число параметров и характеристик. В этих условиях спроектировать устройство с заданными характеристиками способен только квалифицированный специалист. С одной стороны, он должен хорошо знать элементную базу – микросхемы различных типов и уровней интеграции, а также методы логического проектирования, с другой стороны, владеть методами и алгоритмами выполнения основных арифметических операций в вычислительных устройствах.

Пособие содержит материал по представлению числовой информации в ЭВС, выполнению операций двоичной арифметики с фиксированной и плавающей запятой, десятичной арифметики. Наряду с традиционными способами выполнения арифметических операций приводятся также алгоритмы, использование которых приводит к ускорению вычислений.

Каждая тема содержит теоретический материал, а также примеры. В теоретическом материале приводятся необходимые сведения по представлению чисел с фиксированной и плавающей запятой, погрешностям представления чисел, кодированию двоичных чисел со знаком, рассматриваются особенности выполнения операций сложения, вычитания, умножения и деления двоичных чисел в форме с фиксированной и плавающей запятой, операций сложения и вычитания в двоично-десятичном коде. Следует отметить, что материал по двоичной арифметике является достаточно полным для того, чтобы изучать его без обращения к дополнительным источникам. Что касается десятичной арифметики, то в пособии не рассматривается выполнение операций умножения и деления в двоично-десятичном коде, так как указанные операции в настоящее время не используются на практике. Рассматриваемый теоретический материал сопровождается достаточным количеством примеров, что делает его более понятным для изучения.

Основная цель пособия – помочь студентам специальности «Электронные вычислительные средства», изучающим дисциплину «Основы проектирования электронных вычислительных средств», приобрести теоретические и практические навыки выполнения основных арифметических операций. Методическое пособие, посвященное арифметическим основам ЭВС, было выпущено кафедрой достаточно давно (1992 г.), материал по выполнению арифметических операций в нем рассматривался в достаточно сжатом виде.

1 ПРЕДСТАВЛЕНИЕ ЧИСЛОВОЙ ИНФОРМАЦИИ В ЭВС

1.1 Представление чисел с фиксированной и плавающей запятой

В ЭВС числа и нечисловая информация представляются совокупностью двоичных разрядов. Совокупность двоичных разрядов, предназначенных для представления (записи) данных, называется *разрядной сеткой*.

В ЭВС применяют две формы представления чисел – с фиксированной запятой (точкой) и с плавающей запятой (точкой) и называют соответственно *естественной* и *нормальной*.

1.1.1 Представление чисел с фиксированной запятой

При *естественной форме* число записывается в естественном, натуральном виде с выделением в общем случае следующих компонент числа: знака, запятой и цифр числа. Для сокращения длины разрядной сетки и упрощения обработки данных в конкретных ЭВС положение запятой фиксируется схемотехнически, т. е. аппаратными средствами. Эта форма представления числа называется формой с *фиксированной запятой*. Такое название связано с тем, что запятая, отделяющая дробную часть от целой, фиксируется в определенном месте относительно разрядов числа. При этом в слове данных сохраняются только два структурных компонента: один знаковый разряд и n разрядов для представления цифр числа. Для кода знака обычно выделяется крайний слева разряд. В знаковом разряде 1 соответствует минусу, а 0 – плюсу. Обычно положение запятой фиксируется либо после младшего разряда (0 разряда), либо перед старшим разрядом ($n-1$ разрядом). В первом случае числа представлены как целые, во втором – как правильные дроби. При этом запятая никак не обозначается, но в алгоритмах выполнения операций (умножение, деление) ее место учтено заранее одним из указанных способов [1, 5].

В случае целых чисел (рисунок 1) минимальным по модулю отличным от нуля числом будет $A_{\min} = 00\ldots01, = 1$, а максимальным, которому соответствуют единицы во всех n разрядах, – $A_{\max} = 11\ldots11, = 2^n - 1$, т. е. диапазон представления чисел в этом случае $1 \le |A| \le 2^n - 1$.

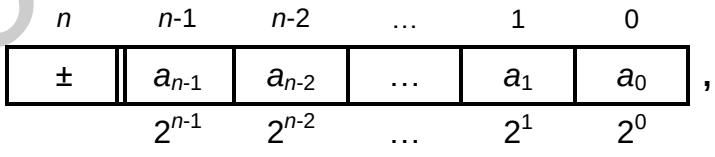


Рисунок 1 – Разрядная сетка для представления целых чисел

В случае правильных дробей (рисунок 2) минимальным по модулю отличным от нуля числом будет $A_{\min} = ,00\ldots01 = 2^{-n}$, а максимальным – $A_{\max} = ,11\ldots11 = 1 - 2^{-n}$, т. е. диапазон представления чисел в этом случае $2^{-n} \le |A| \le 1 - 2^{-n}$.

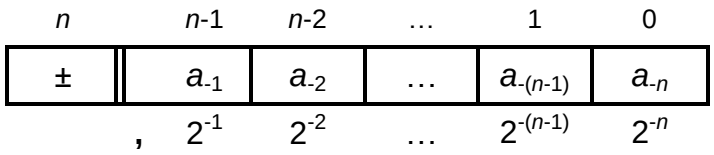


Рисунок 2 – Разрядная сетка для представления правильных дробей

Достоинство фиксированной запятой: имеется возможность построить сравнительно несложные арифметические устройства с высоким быстродействием.

1.1.2 Представление чисел с плавающей запятой

Для расширения диапазона представления чисел и уменьшения погрешности их задания используется нормальная форма записи [1, 5].

В общем случае нормальная форма записи числа может быть представлена в виде $A = \pm Mr^{\pm p}$, где M – мантисса числа; r – основание системы счисления; p – порядок числа. Мантисса $|M| < 1$, т. е. является правильной дробью, а порядок p – целое число.

Порядок (с учетом знака) показывает, на сколько разрядов и в какую сторону сдвинута запятая при замене формы записи числа с естественной на нормальную. Положение запятой в мантиссе M определяется величиной порядка p. С изменением порядка в большую или меньшую сторону запятая перемещается влево или вправо, т. е. «плавает» в изображении числа. Поэтому такую форму записи называют *представлением чисел с плавающей запятой*.

Формат числа с *плавающей запятой* (рисунок 3) включает: один разряд для представления знака порядка, q разрядов для представления порядка p, один разряд для представления знака мантиссы, m разрядов для представления мантиссы M.

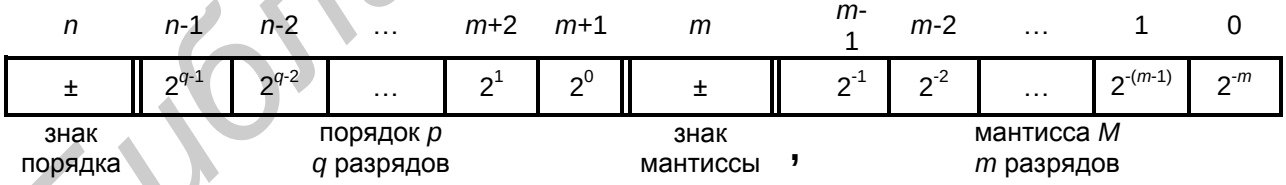


Рисунок 3 – Разрядная сетка для представления чисел с плавающей запятой

В условиях ограничения разрядной сетки максимально возможную точность представления чисел имеет нормальная форма записи числа, при которой старшая цифра мантиссы является значащей, т. е. $1/r \le |M| < 1$. Такие числа называются *нормализованными*. Для двоичной системы счисления $r = 2$ и $0{,}5 \le |M| < 1$, т. е. старший разряд мантиссы должен быть равен единице. Например: $0{,}10101100 \cdot 2^{011}$ – нормализованное число, а $0{,}00101011 \cdot 2^{101}$ – ненор-

мализованное. Нормализованное представление чисел с плавающей запятой позволяет сохранить в мантиссе большее число разрядов с единицами, что повышает точность вычислений.

При выполнении действий над числами с плавающей запятой определенные операции выполняются как над мантиссами, так и над порядками. Для упрощения операций над порядками их сводят к действиям над целыми положительными числами (числами без знака), применяя представление чисел с плавающей запятой со смещенным порядком. В случае представления числа с плавающей запятой со смещенным порядком к его порядку p прибавляется целое число – смещение 2^q, где q – число двоичных разрядов, используемых для представления модуля порядка, $p_{\text{см}} = p + 2^q$. Смещенный порядок будет всегда положительным. Для его представления необходимо такое же число разрядов, как и для модуля и знака порядка – q+1 разрядов. Важная особенность смещенных порядков состоит в том, что если для порядков p_1 и p_2, представляющих собой целые числа со знаками, выполняется соотношение $p_1 > p_2$, то и для положительных целых чисел, соответствующих смещенных порядков $p_{\text{см1}}$ и $p_{\text{см2}}$, также будет выполняться соотношение $p_{\text{см1}} > p_{\text{см2}}$.

Для устранения неоднозначности смещенные порядки называют *характеристиками*.

Наименьшее по модулю число с плавающей запятой с ненулевой нормализованной мантиссой может быть представлено единицей в старшем разряде с весом 2^{-1} и наибольшим по абсолютной величине отрицательным порядком:

$$A_{\min} = M_{\min} 2^{-p_{\max}} = 2^{-1} 2^{-(2^q - 1)} = 2^{-2^q},$$

а наибольшее по модулю число с плавающей запятой

$$A_{\max} = M_{\max} 2^{p_{\max}} = (1 - 2^{-m}) 2^{2^q - 1}.$$

Сравним диапазон представления чисел с фиксированной и плавающей запятой. Пусть, например, n = 31 (разрядная сетка состоит из 32 разрядов). В этом случае максимальное целое число с фиксированной запятой по абсолютному значению равно $2^{31} - 1 = 2\,147\,483\,647$.

Для числа с плавающей запятой (n = 31, m = 24, q = 6 и два разряда используются для кодирования знаков мантиссы и порядка) максимальное значение числа равно $(1 - 2^{-24}) 2^{64-1} \approx 1 \times 2^{63} \approx 10^{19}$.

Из сравнения этих двух значений вытекает, что при одинаковом числе разрядов n форма с плавающей запятой обеспечивает *более широкий диапазон представления чисел*.

В то же время операции над числами с фиксированной запятой выполняются быстрее, чем операции над числами с плавающей запятой. Поэтому выбор той или иной формы представления чисел часто диктуется классом задач, временем, необходимым для их решения, точностью вычислений, диапазоном изменения исходных данных и результата.

1.1.3 Погрешности представления чисел

Представление числовой информации в ЭВС, как правило, влечет за собой появление погрешностей (ошибок), величина которых зависит от формы представления чисел и от длины разрядной сетки [1, 5].

Абсолютная погрешность представления – разность между истинным значением исходной величины A и ее значением, полученным из машинного изображения A_M, т. е. $\Delta[A] = A - A_M$.

Относительная погрешность представления – величина $\delta[A] = \frac{\Delta[A]}{A_M}$.

Все числа целого типа переводятся из одной системы счисления в другую точно (без погрешностей). Операции сложения, вычитания и умножения чисел целого типа также выполняются точно. Результат деления чисел целого типа дает целое частное и целый остаток.

Правильные дроби переводятся из одной системы счисления в другую неточно – с погрешностью. Некоторая величина в одной системе счисления может иметь конечное значение, а в другой системе счисления становится бесконечной величиной, например, дробь 1/10 имеет конечное десятичное представление, но, будучи переведена в двоичную систему счисления, становится бесконечной дробью 0,0001100110011... Однако эту погрешность нетрудно оценить, если известны истинные значения исходных чисел.

Поэтому для чисел с фиксированной запятой погрешность представления имеет место только в случае правильных дробей.

Абсолютная погрешность перевода десятичной дроби в систему с основанием r при использовании усечения (младшие цифры справа отбрасываются) определяется следующим выражением:

$$\Delta[A] = a_{-(n+1)}r^{-(n+1)} + a_{-(n+2)}r^{-(n+2)} + \cdots = \sum_{i=-(n+1)}^{-\infty} a_i r^i.$$

Для двоичной системы счисления $r = 2$, и максимальное значение этой погрешности получается при $a_i = 1$:

$$\Delta[A]_{\max} = \sum_{i=-(n+1)}^{-\infty} 1 \cdot 2^i = 2^{-n} \sum_{i=-1}^{-\infty} 2^i = 2^{-n},$$

т. е. максимальная абсолютная ошибка равна значению младшего разряда.

При использовании округления максимальная абсолютная ошибка не превышает половины значения младшего разряда:

$$\Delta[A]_{\max} = 2^{-(n+1)} = 0{,}5 \cdot 2^{-n}.$$

Для представления чисел в форме с фиксированной запятой абсолютное значение машинного изображения правильной дроби находится в диапазоне

$$2^{-n} \le |A|_{\Phi} \le 1 - 2^{-n}.$$

Следовательно, относительная погрешность представления для максимального значения числа равна

$$\delta[A]_{\Phi\min} = \frac{\Delta[A]}{|A|_{\Phi\max}} = \frac{0{,}5 \cdot 2^{-n}}{1 - 2^{-n}}.$$

В современных ЭВС, как правило, n = 16…64, поэтому $1 \gg 2^{-n}$, откуда

$$\delta[A]_{\Phi\min} = 0{,}5 \cdot 2^{-n}.$$

Аналогично для минимального значения:

$$\delta[A]_{\Phi\max} = \frac{\Delta[A]}{|A|_{\Phi\min}} = \frac{0{,}5 \cdot 2^{-n}}{2^{-n}} = 0{,}5,$$

откуда видно, что погрешности представления малых чисел в форме с фиксированной запятой могут быть очень значительными.

Все числа с плавающей запятой представляют некоторые данные приближенно.

Для представления чисел в форме с плавающей запятой абсолютное значение мантиссы находится в диапазоне

$$0{,}5 \le |M| < 1 - 2^{-n},$$

поэтому ее погрешности определяются так же, как и для чисел с фиксированной запятой.

Для нахождения погрешностей представления чисел в форме с плавающей запятой величину этой погрешности надо умножить на величину 2^{p_A}:

$$\Delta[A]_{\Pi} = 0{,}5 \cdot 2^{-m} \cdot 2^{p_A},$$

$$\delta[A]_{\Pi\min} = \frac{0{,}5 \cdot 2^{-m} \cdot 2^{p_A}}{(1 - 2^{-m}) \cdot 2^{p_A}} = \frac{0{,}5 \cdot 2^{-m}}{1 - 2^{-m}} \approx 0{,}5 \cdot 2^{-m},$$

$$\delta[A]_{\Pi\max} = \frac{0{,}5 \cdot 2^{-m} \cdot 2^{p_A}}{0{,}5 \cdot 2^{p_A}} = 2^{-m}.$$

Из полученных выражений следует, что максимальная абсолютная ошибка не превышает половины значения младшего разряда мантиссы с учетом порядка, а относительная точность представления чисел в форме с плавающей запятой почти не зависит от величины числа.

1.2 Кодирование двоичных чисел со знаком

Для машинного представления чисел со знаком используются три способа: *прямой*, *дополнительный* и *обратный коды* [1, 5]. При рассмотрении кодов будем считать, что разрядная сетка содержит $n + 1$ двоичных разрядов, из которых n младших разрядов используются для задания значения числа. При изображении целых чисел точкой условно отделяют знаковый разряд от числовой части.

1.2.1 Прямой код

Прямой код числа есть представление числа в виде абсолютного значения с кодом знака. Знак плюс кодируется двоичным 0, а минус – 1.

Правило образования прямого кода целого числа записывается в виде

$$[A]_{\text{пр}} = \begin{cases} A, \text{если } A \geq 0; \\ 2^n + |A| = 2^n - A, \text{если } A \leq 0, \end{cases}$$

где n – количество разрядов, используемых для представления значения числа.

Знаковый разряд занимает позицию с весом 2^n.

Пример 1. Записать прямой код целого числа при $n = 4$.

$$A_1 = +1011; [A_1]_{\text{пр}} = 0.1011.$$
$$A_2 = -1011; [A_2]_{\text{пр}} = 1.1011.$$

Правило образования прямого кода правильной дроби записывается в виде

$$[A]_{\text{пр}} = \begin{cases} A, \text{если } A \geq 0; \\ 1 + |A| = 1 - A, \text{если } A \leq 0. \end{cases}$$

Знаковый разряд занимает позицию с весом 2^0.

Пример 2. Записать прямой код целого числа при $n = 4$.

$$A_1 = +0{,}1011; [A_1]_{\text{пр}} = 0{,}1011.$$
$$A_2 = -0{,}1011; [A_2]_{\text{пр}} = 1{,}1011.$$

Из определения прямого кода следует, что нуль в прямом коде имеет два изображения:

$$A = +00 \ldots 00; [A]_{\text{пр}} = 0.00 \ldots 00.$$
$$A = -00 \ldots 00; [A]_{\text{пр}} = 1.00 \ldots 00.$$

Достоинством прямого кода является удобство выполнения операций умножения и деления. К недостатку представления чисел в прямом коде следует отнести сложность правил выполнения сложения и вычитания при различных комбинациях знаков чисел.

Операции сложения и вычитания чисел с разными знаками в ЭВС сводятся к операции сложения. Для этого используются дополнительный и обратный коды представления чисел со знаком.

Дополнительный и обратный коды положительных чисел совпадают с их прямым кодом.

Основой дополнительного и обратного кодов отрицательных чисел являются дополнения чисел до:

– для целых чисел дополнения до 2^n или до $2^n - 1$ соответственно;

– для правильных дробей дополнения до 1 или до $1 - 2^{-n}$ соответственно.

Для лучшего понимания замены операции вычитания операцией сложения рассмотрим пример.

Пример 3. Вычислить разность $A - B$, для $A = 111$, $B = 110$ ($n = 3$).

Представим B в виде дополнения до 2^n:

$$B' = 2^3 - B = 1000 - 110 = 010.$$

Заменим далее операцию вычитания операцией сложения:

$$A - B = A + B' = 111 + 010 = \boxed{1}\,001.$$

отбрасывается

Результат как n-разрядная разность получился верным. Однако появившуюся единицу с весом 2^3 необходимо отбросить, так как вместо вычитания из числа A числа $B = 6$ к числу A было добавлено $B' = 2$. Таким образом, сумма получена с избытком $2^n = 8$.

1.2.2 Дополнительный код

В основе дополнительного кода n-разрядного отрицательного двоичного числа A лежит дополнение:

– для целого числа до 2^n: $A' = 2^n - |A|$;

– для правильной дроби до 1: $A' = 1 - |A|$.

Дополнительный код *целого* отрицательного n-разрядного числа A представляет собой дополнение до 2^{n+1} и определяется следующим образом:

$$[A]_{\text{доп}} = 2^{n+1} + A = 2^{n+1} - |A| = 2^n + \underbrace{2^n - |A|}_{\text{дополнение}} = 2^n + \underbrace{2^n - 1 - |A|}_{\text{инверсия}} + 1.$$

код знака

Формально переход к дополнительному коду осуществляется следующим образом:

1) в знаковый разряд записывается единица;

2) в значащих разрядах нули заменяются единицами, а единицы – нулями, т. е. значащие разряды инвертируются;

3) к младшему разряду прибавляется единица.

Пример 4. Записать дополнительный код числа $A = -101101$, $n = 6$.

```
 10.000000  2^(n+1)        1.000000   2^n          1.000000   2^n
-   101101  |A|          -   101101   |A|        -        1
  1.010011  [A]доп         0.010011   A'           0.111111
                         + 1.000000   2^n        -   101101   |A|
                           1.010011   [A]доп        0.010010  инверсия |A|
                                                 +        1
                                                    0.010011   A'
                                                 + 1.000000   2^n
                                                    1.010011  [A]доп
```

Дополнительный код отрицательной n-разрядной *правильной дроби* A представляет собой дополнение до 2 и определяется следующим образом:

$$[A]_{\text{доп}} = 2 + A = 2 - |A| = 1 + \underbrace{1 - |A|}_{\text{дополнение}} = 1 + \underbrace{1 - 2^{-n} - |A|}_{\text{инверсия}} + 2^{-n}.$$

код знака

Пример 5. Записать дополнительный код числа $A = -0{,}101101$, $n = 6$.

```
 10,000000   2          1,000000                1,000000
- 0,101101  |A|       - 0,101101  |A|         - 0,000001   2^-n
  1,010011  [A]доп      0,010011   A'           0,111111
                      + 1,000000                - 0,101101  |A|
                        1,010011  [A]доп          0,010010  инверсия |A|
                                                + 0,000001   2^-n
                                                  0,010011   A'
                                                + 1,000000
                                                  1,010011   [A]доп
```

Формально переход к дополнительному коду осуществляется следующим образом:

1) в знаковый разряд записывается единица;

2) в значащих разрядах нули заменяются единицами, а единицы – нулями, т. е. значащие разряды инвертируются;

3) к младшему разряду прибавляется единица.

Нуль в дополнительном коде имеет одно представление: 0.00…00.

1.2.3 Обратный код

В основе обратного кода n-разрядного отрицательного двоичного числа A лежит дополнение:

– для целого числа до $2^n - 1$: $A'' = 2^n - 1 - |A|$ (инверсия $|A|$);

– для правильной дроби до $1 - 2^{-n}$: $A'' = 1 - 2^{-n} - |A|$ (инверсия $|A|$).

Обратный код *целого* отрицательного n-разрядного числа A представляет собой дополнение до 2^{n+1} без единицы младшего разряда и определяется следующим образом:

$$[A]_{\text{обр}} = 2^{n+1} - 1 + A = 2^{n+1} - 1 - |A| = 2^n + \underbrace{2^n - 1 - |A|}_{\text{дополнение (инверсия)}}.$$

код знака

Пример 6. Записать обратный код числа $A = -101101$, $n = 6$.

```
 10.000000  2^(n+1)        1.000000   2^n
-       1                -       1
  1.111111                 0.111111
-  101101   |A|          -  101101   |A|
  1.010010  [A]обр         0.010010   A" = инверсия |A|
                         + 1.000000   2^n
                           1.010010   [A]обр
```

Обратный код отрицательной n-разрядной *правильной дроби* A представляет собой дополнение до 2 без единицы младшего разряда и определяется следующим образом:

$$[A]_{\text{обр}} = 2 - 2^{-n} + A = 2 - 2^{-n} - |A| = 1 + \underbrace{1 - 2^{-n} - |A|}_{\text{дополнение (инверсия)}}.$$

код знака

Пример 7. Записать обратный код числа $A = -0{,}101101$, $n = 6$.

```
 10,000000   2          1,000000
- 0,000001   2^-n      - 0,000001   2^-n
  1,111111                0,111111
- 0,101101   |A|        - 0,101101   |A|
  1,010010   [A]обр       0,010010   A" = инверсия |A|
                        + 1,000000
                          1,010010   [A]обр
```

Формально переход к обратному коду осуществляется следующим образом:

1) в знаковый разряд записывается единица;

2) в значащих разрядах нули заменяются единицами, а единицы – нулями, т. е. значащие разряды инвертируются.

Нуль в обратном коде имеет два представления:

$A = +00 \ldots 00;\ [A]_{\text{обр}} = 0.00 \ldots 00.$

$A = -00 \ldots 00;\ [A]_{\text{обр}} = 1.11 \ldots 11.$

1.3 Двоично-кодированное представление десятичных чисел

Для сочетания простоты двоичной системы с удобствами десятичной применяют схемы, в которых цифры десятичного числа кодируются группами двоичных разрядов. При этом каждая такая группа рассматривается как одно целое. Схема кодирования, при которой каждая десятичная цифра представляется двоично-кодированной группой, называется *двоично-кодированной десятичной схемой* [1].

Наиболее часто используется такое *двоично-кодированное представление десятичного числа*, в котором каждая десятичная цифра изображается группой из четырех двоичных разрядов (тетрадой):

$$A_{\text{Д}} = a_{n-1} \dots a_1 a_0 = \{\alpha_3^{n-1}\alpha_2^{n-1}\alpha_1^{n-1}\alpha_0^{n-1}\}_{n-1} \dots \{\alpha_3^1\alpha_2^1\alpha_1^1\alpha_0^1\}_1\{\alpha_3^0\alpha_2^0\alpha_1^0\alpha_0^0\}_0,$$

где a_j – j-я десятичная цифра;

α_i^j – i-й разряд j-й тетрады;

n – количество десятичных цифр.

Количество различных кодов при двоично-кодированном представлении десятичного числа определяется количеством возможных сочетаний по 10 из 16 комбинаций, которые допускает тетрада.

При двоично-кодированном представлении десятичного числа следует исходить из общих требований, предъявляемых к системам счисления:

1) различным десятичным цифрам должны соответствовать различные тетрады;

2) большая десятичная цифра должна изображаться большей тетрадой, если разряды тетрады имеют веса по двоичной системе счисления;

3) для двух десятичных цифр $a = \{\alpha_3\alpha_2\alpha_1\alpha_0\}$ и $b = \{\beta_3\beta_2\beta_1\beta_0\}$, связанных соотношением $a + b = 9$, должно удовлетворяться условие

$$\beta_i = \begin{cases} 0, \text{ если } \alpha_i = 1; \\ 1, \text{ если } \alpha_i = 0; \end{cases} \quad i = 0, 1, 2, 3.$$

Коды, обладающие таким свойством, называют *самодополнительными*. Они применяются при выполнении арифметических действий над десятичными числами, представленными в обратном или дополнительном коде;

4) для однозначности перевода чисел желательно, чтобы разряды тетрад имели определенный вес. Тогда значение десятичной цифры a соответствует выражению

$$a = \alpha_3\sigma_3 + \alpha_2\sigma_2 + \alpha_1\sigma_1 + \alpha_0\sigma_0,$$

где σ_3, σ_2, σ_1, σ_0 – веса соответствующих разрядов тетрады.

Существует несколько способов кодирования десятичных цифр. В таблице 1 приведено кодирование десятичных цифр в различных схемах представления десятичных чисел.

Наиболее известна *схема 8421 BCD* (8421 Binary-Coded Decimal) – *двоично-десятичный код*. В коде 8421 разрешенные комбинации соответствуют двоичным эквивалентам десятичных цифр с весами разрядов, равными степеням основания 2. Каждая цифра десятичного числа представляется в двоичной форме и изображается соответствующим 4-разрядным двоичным числом (тетрадой). Код 8421 является взвешенным кодом. Для этого кода не выполняется условие 3, так как цифры, являющиеся дополнением до 9, не получаются простым инвертированием тетрад.

Таблица 1

Десятичная цифра	Двоичные коды				
	8421	2421	8421+3	7421	2 из 5
0	0000	0000	0011	0000	11000
1	0001	0001	0100	0001	01100
2	0010	0010	0101	0010	00110
3	0011	0011	0110	0011	00011
4	0100	0100	0111	0100	10001
5	0101	1011	1000	0101	10100
6	0110	1100	1001	0110	01010
7	0111	1101	1010	1000	00101
8	1000	1110	1011	1001	10010
9	1001	1111	1100	1010	01001

Важно понимать, что двоично-десятичный код не является еще одной системой счисления, такой, как двоичная, восьмеричная, десятичная и шестнадцатеричная системы. Это фактически та же десятичная система, в которой каждый десятичный разряд представлен двоичным эквивалентом. Также важно понимать, что двоично-десятичное число – не то же самое, что обычное двоичное число. Обычный двоичный код представляет в двоичной системе какое-либо десятичное число полностью, а двоично-десятичный код преобразовывает в двоичную систему каждый десятичный разряд по отдельности.

Код 8421 является естественным представлением десятичных цифр в двоичной системе и удобен для ввода/вывода в цифровых устройствах десятичных чисел. Однако использование этого кода связано со сложностью выполнения арифметических операций над десятичными числами.

Для *кода 2421* веса разрядов тетрады соответственно равны 2, 4, 2, 1. Таблица кодирования делится на две части: от 0 до 4 – тетрады повторяют двоичные эквиваленты, от 5 до 9 – по сравнению с двоичной системой каждая тетрада содержит избыток +0110 (+6). Это дает возможность любую цифру одной части таблицы превратить в ее дополнение до 9 простым инвертированием разрядов, т. е. код 2421 является *самодополнительным*. Код 2421 применяется для выполнения арифметических действий над десятичными числами, представленными в обратном или дополнительном коде.

Код 8421+3 называют BCD кодом с избытком 3. В этом коде все тетрады имеют значения на три единицы большие, чем тетрады кода 8421 (отсюда название кода). Для него не существует целочисленных значений веса, которые удовлетворяли бы условию 4, т. е. код 8421+3 не является взвешенным. В коде с избытком 3 комбинация, соответствующая любой из десятичных цифр, представляет собой инверсию кодовой комбинации, соответствующей ее дополнению до девяти, т. е. код 8421+3 также является *самодополнительным*.

Особенность *кода 7421* заключается в том, что любая кодовая комбинация содержит не более двух единиц.

Код 2 из 5 представляет собой *невзвешенный* код с 5 битами в каждой группе. В коде 2 из 5 все кодовые комбинации содержат только две единицы. Это свойство используется для обнаружения ошибочных комбинаций.

2 ДВОИЧНАЯ АРИФМЕТИКА С ФИКСИРОВАННОЙ ЗАПЯТОЙ

2.1 Формальные правила двоичной арифметики

Правила выполнения арифметических действий в двоичной системе счисления представлены в таблице 2 [1].

Таблица 2

Операция		
Сложение	Вычитание	Умножение
0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = (1)0 Перенос в старший разряд	0 − 0 = 0 1 − 0 = 1 1 − 1 = 0 0 − 1 = (1)1 Заем из старшего разряда	0 × 0 = 0 0 × 1 = 0 1 × 0 = 0 1 × 1 = 1

Правила поразрядных действий при сложении двух двоичных чисел $A + B$ представлены в таблице 3, где a_{i-1} и b_{i-1} – i-й разряд соответственно числа A и B; c_{i-1} – перенос из $(i - 1)$-го разряда, c_i – перенос в $(i + 1)$-й разряд, s_i – i-й разряд суммы.

Правила поразрядных действий при вычитании двух двоичных чисел $A - B$ представлены в таблице 4, где z_i – заем из i-го разряда, z_{i+1} – заем из $(i + 1)$-го разряда, r_i – i-й разряд разности.

Заем равносилен вычитанию единицы из старшего разряда, поэтому в таблице 4 заем показан со знаком минус.

Таблица 3

a_i	b_i	c_{i-1}	s_i	c_i
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Таблица 4

a_i	b_i	z_i	r_i	z_{i+1}
0	0	0	0	0
1	0	0	1	0
1	1	0	0	0
0	1	0	1	-1
0	0	-1	1	-1
1	0	-1	0	0
1	1	-1	1	-1
0	1	-1	0	-1

2.2 Сложение и вычитание чисел со знаком в форме с фиксированной запятой

2.2.1 Сложение в прямых кодах

При сложении в *прямых кодах* отсутствует цепь поразрядного переноса между старшим значащим и знаковым разрядами. В прямых кодах можно складывать только числа, имеющие одинаковые знаки, т. е. прямой код не подходит для выполнения операции алгебраического сложения. При этом сумма чисел имеет знак любого из слагаемых, а значащая часть результата получается путем сложения значащих частей слагаемых.

Пример 8. Сложить два числа в прямых кодах: A = +1011, B = +100 (n = 4).

Решение. $[A]_{\text{пр}} = 0.1011; [B]_{\text{пр}} = 0.0100$

```
  1011    |A|
+ 0100    |B|
  1111    |A + B|
```

$[A + B]_{\text{пр}} = 0.1111.$

Ответ. $A + B = +1111.$

2.2.2 Сложение в дополнительных кодах

Сложение в *дополнительных кодах* характеризуется наличием цепи поразрядного переноса из старшего разряда значащей части в знаковый разряд.

Определим правила сложения чисел в дополнительном коде [1].

Теорема. *Сумма дополнительных кодов чисел есть дополнительный код суммы чисел.*

Доказательство. Предположим, что числа представлены в форме с фиксированной запятой, стоящей после младшего разряда, т. е. слагаемые являются целыми числами. Рассмотрим возможные случаи.

1. $A > 0,\ B > 0,\ A + B < 2^n$.

Так как $[A]_{\text{доп}} = A$, $[B]_{\text{доп}} = B$, то $[A]_{\text{доп}} + [B]_{\text{доп}} = \underbrace{A + B}_{\text{положительное число}} = [A + B]_{\text{доп}}$.

2. $A < 0,\ B > 0,\ |A| > B$.

Так как $[A]_{\text{доп}} = 2^{n+1} + A,\ [B]_{\text{доп}} = B,$ то $[A]_{\text{доп}} + [B]_{\text{доп}} = 2^{n+1} + \underbrace{(A + B)}_{\text{отрицательное число}} =$
$= [A + B]_{\text{доп}}$.

3. $A < 0,\ B > 0,\ |A| \le B$.

Так, как и в предыдущем случае, здесь

$$[A]_{\text{доп}} = 2^{n+1} + A,\ [B]_{\text{доп}} = B,\ [A]_{\text{доп}} + [B]_{\text{доп}} = 2^{n+1} + (A + B).$$

Однако в этом случае (A + B) число положительное, и поэтому

$$2^{n+1} + (A + B) > 2^{n+1}.$$

Так как значение этой суммы больше 2^{n+1}, то появляется единица переноса из знакового разряда, что равносильно изъятию из суммы 2^{n+1} единиц. Если отбросить эту единицу переноса из знакового разряда, то получим

$$[A]_{\text{доп}} + [B]_{\text{доп}} = \underbrace{2^{n+1}}_{\substack{\text{единица переноса из знакового разряда, которая отбрасывается}}} + \underbrace{(A + B)}_{\text{положительное число}} = A + B = [A + B]_{\text{доп}}.$$

4. $A < 0,\ B < 0,\ |A + B| < 2^n$.

Так как $[A]_{\text{доп}} = 2^{n+1} + A$, $[B]_{\text{доп}} = 2^{n+1} + B$, то

$$[A]_{\text{доп}} + [B]_{\text{доп}} = \underbrace{2^{n+1}}_{\substack{\text{единица переноса из знакового разряда, которая отбрасывается}}} + 2^{n+1} + \underbrace{(A + B)}_{\text{отрицательное число}} = 2^{n+1} + (A + B) = [A + B]_{\text{доп}}.$$

В этом случае получается отрицательный результат, и также появляется единица переноса из знакового разряда, которая отбрасывается.

Таким образом, теорема справедлива для всех случаев, в которых не возникает переполнение разрядной сетки, что позволяет складывать числа в дополнительном коде по правилам двоичной арифметики, не разделяя знак и значащую часть кода.

При сложении чисел с использованием дополнительного кода выполняется арифметическое сложение кодов слагаемых, включая их знаковые разряды, с учетом возможного переноса в знаковый разряд из старшего значащего разряда суммы. При возникновении переноса из знакового разряда суммы единица переноса отбрасывается. При сложении код знака результата получается автоматически.

Пример 9. Сложить два числа в дополнительных кодах: A = +1001, B = -101 (n = 4).

Решение. $[A]_{\text{доп}} = 0.1001$; $[B]_{\text{доп}} = 1.1011$

```
  0.1001    [A]доп
+ 1.1011    [B]доп
 10.0100    [A + B]доп = [A + B]пр
```

единица переноса из знакового разряда, которая отбрасывается

Ответ. $A + B = +100$.

2.2.3 Сложение в обратных кодах

Сложение в обратных кодах характеризуется наличием цепи поразрядного переноса из старшего разряда значащей части в знаковый разряд, а также цепи *кругового*, или *циклического*, переноса из знакового разряда в младший разряд значащей части.

Определим правила сложения чисел в обратном коде [1].

Теорема. *Сумма обратных кодов чисел есть обратный код суммы чисел*.

Доказательство. Предположим, что числа представлены в форме с фиксированной запятой, стоящей после младшего разряда, т. е. слагаемые являются целыми числами. Рассмотрим возможные случаи.

1. $A > 0,\ B > 0,\ A + B < 2^n$.

Так как $[A]_{\text{обр}} = A$, $[B]_{\text{обр}} = B$, то $[A]_{\text{обр}} + [B]_{\text{обр}} = \underbrace{A + B}_{\text{положительное число}} = [A + B]_{\text{обр}}$.

2. $A < 0,\ B > 0,\ |A| > B$.

Так как $[A]_{\text{обр}} = 2^{n+1} - 1 + A$, $[B]_{\text{обр}} = B$, то

$$[A]_{\text{обр}} + [B]_{\text{обр}} = 2^{n+1} - 1 + \underbrace{(A + B)}_{\text{отрицательное число}} = [A + B]_{\text{обр}}.$$

3. $A < 0,\ B > 0,\ |A| < B.$

Так, как и в предыдущем случае, здесь

$$[A]_{\text{обр}} = 2^{n+1} - 1 + A,\ [B]_{\text{обр}} = B,\ [A]_{\text{обр}} + [B]_{\text{обр}} = 2^{n+1} - 1 + (A + B).$$

Однако в этом случае $(A + B)$ число положительное, и поэтому

$$2^{n+1} - 1 + (A + B) > 2^{n+1}.$$

Так как значение этой суммы больше 2^{n+1}, то появляется единица переноса из знакового разряда, что равносильно изъятию из суммы 2^{n+1} единиц. Кроме того, сумма получается уменьшенной на единицу. Если отбросить единицу переноса из знакового разряда и добавить ее в младший разряд суммы, то получим

$$[A]_{\text{обр}} + [B]_{\text{обр}} = 2^{n+1} - 1 + \underbrace{(A + B)}_{\text{положительное число}} = A + B = [A + B]_{\text{обр}}$$

единица переноса из знакового разряда, которая цикличе-ски прибавляется к младшему разряду

4. $A < 0,\ B < 0,\ |A + B| < 2^n.$

Так как $[A]_{\text{обр}} = 2^{n+1} - 1 + A$, $[B]_{\text{обр}} = 2^{n+1} - 1 + B$, то

$$[A]_{\text{обр}} + [B]_{\text{обр}} = 2^{n+1} - 1 + 2^{n+1} - 1 + \underbrace{(A + B)}_{\text{отрицательное число}} = 2^{n+1} - 1 + (A + B) =$$
$$= [A + B]_{\text{обр}}.$$

единица переноса из знакового разряда, которая цикличе-ски прибавляется к младшему разряду

В этом случае получается отрицательный результат и также появляется единица переноса из знакового разряда, которая добавляется в младший разряд суммы.

5. $A < 0,\ B > 0,\ |A| = B.$

В этом случае $[\text{A}]_{\text{обр}} = 2^{n+1} - 1 + \text{A}$, $[\text{B}]_{\text{обр}} = \text{B}$, поэтому

$$[A]_{\text{обр}} + [B]_{\text{обр}} = 2^{n+1} - 1 + \underbrace{(A + B)}_{\text{равно нулю}} = 2^{n+1} - 1 = \underbrace{1.11 \ldots 11}_{n+1} = [A + B]_{\text{обр}},$$

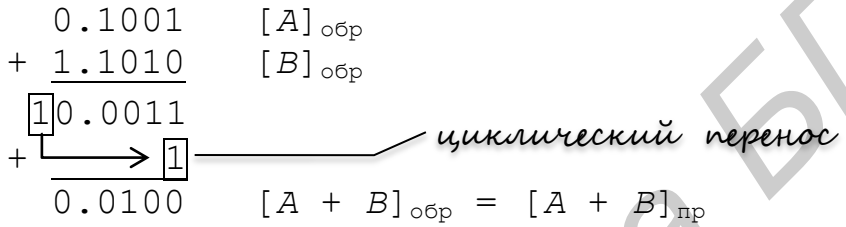
т. е. сумма равна нулю (получаем одно из изображений нуля в обратном коде).

Таким образом, теорема справедлива для всех случаев, в которых не возникает переполнение разрядной сетки, что позволяет складывать числа в обратном коде по правилам двоичной арифметики, не разделяя знак и значащую часть кода.

При сложении чисел с использованием обратного кода выполняется арифметическое сложение кодов слагаемых, включая их знаковые разряды. При возникновении переноса из знакового разряда суммы единица переноса прибавляется к младшему разряду суммы. Такой перенос называется круговым, или циклическим. При сложении код знака результата получается автоматически.

Пример 10. Сложить два числа в обратных кодах: A = +1001, B = –101 (n = 4).

Решение. $[A]_{\text{обр}} = 0.1001; [B]_{\text{обр}} = 1.1010$



Ответ. $A + B = +100$.

Таким образом, сложение с использованием обратных кодов выполняется, как и в случае дополнительных кодов, по обычным правилам сложения двоичных чисел. При этом знаковый разряд рассматривается наравне со значащими разрядами. Однако в отличие от сложения с использованием дополнительного кода единица переноса из знакового разряда не отбрасывается, а прибавляется к младшему разряду суммы.

Следует помнить, что при сложении двух одинаковых по абсолютной величине чисел с разными знаками в случае применения обратного кода получается отрицательный нуль 1.11…11. Однако отрицательный нуль в ЭВС, как правило, автоматически преобразуется к положительному, т. е. к виду 0.00…00.

Сложение правильных дробей ничем не отличается от сложения целых чисел.

Операция вычитания чисел в ЭВС сводится к операции сложения с использованием дополнительного и обратного кодов.

2.2.4 Переполнение при сложении чисел с фиксированной запятой

При сложении двух чисел с одинаковыми знаками может получиться результат, превосходящий по абсолютной величине максимально допустимое для данного формата число, т. е. для представления абсолютного значения результата потребуется n + 1 двоичный разряд. Такой результат означает, что произошло *переполнение разрядной сетки*, называемое часто просто *переполнением*.

При сложении правильных дробей переполнение означает, что результат по абсолютной величине получился большим или равным единице, а при сложении целых чисел – 2^n.

Если при сложении чисел с фиксированной запятой происходит переполнение, то результат не может быть использован в дальнейших вычислениях. Случаи переполнения в ЭВМ фиксируются с помощью специального флага.

При сложении чисел в прямом коде признаком переполнения разрядной сетки является появление единицы переноса из старшего разряда значащей части числа.

Так как переполнение возникает лишь при сложении чисел с одинаковыми знаками, то признаком переполнения при сложении чисел в дополнительном и обратном кодах может служить противоположность знака результата знакам слагаемых.

Пример 11. Сложить два отрицательных числа в дополнительных кодах: $A = -1011$, $B = -1101$ ($n = 4$).

Решение. $[A]_{\text{доп}} = 1.0101$; $[B]_{\text{доп}} = 1.0011$.

```
  1.0101    [A]доп
+ 1.0011    [B]доп
 10.1000    [A + B]доп переполнение
```

отбрасывается

Ответ. Переполнение.

В примере результат суммирования оказался неверным как по величине, так и по знаку, поскольку знаковый разряд занят старшим разрядом значащей части суммы.

В ЭВС для обнаружения переполнения анализируются переносы в знаковый разряд и из знакового разряда. Если эти переносы либо оба отсутствуют, либо оба имеются, то переполнения нет. Наличие же переноса только в знаковый разряд либо только из знакового разряда является признаком того, что имеет место переполнение. В рассмотренном выше примере при сложении отсутствовал перенос в знаковый разряд, но был перенос из знакового разряда, поэтому имело место переполнение. В рассмотренных ранее примерах сложения чисел в дополнительном и обратном кодах оба переноса присутствовали, т. е. переполнения не было.

Существует, однако, один случай, когда при суммировании чисел в дополнительном коде приведенное правило не обнаруживает переполнения. Это происходит, когда сумма модулей двух отрицательных дробных чисел равна единице.

Пример 12. Сложить два дробных отрицательных числа в дополнительных кодах: $A = -0{,}1011$, $B = -0{,}0101$ ($n = 4$).

Решение. $[A]_{\text{доп}} = 1{,}0101$; $[B]_{\text{доп}} = 1{,}1011$.

```
   1,0101    [A]доп
 + 1,1011    [B]доп
  11,0000    [A + B]доп переполнение
```

отбрасывается

Ответ. Переполнение.

В примере знак результата совпадает со знаком слагаемых, имеются переносы в знаковый разряд и из знакового разряда. Однако возникает переполнение. Для обнаружения такого переполнения требуется схема анализа на нуль значащих разрядов.

2.2.5 Модифицированные коды

Чтобы обнаружить переполнение разрядной сетки, можно ввести в знаковую часть изображения числа вспомогательный разряд, который называют разрядом *переполнения*. Такое представление числа называется *модифицированным*, а соответствующие коды – *модифицированным прямым*, *дополнительным* и о*братным кодами*.

С формальной точки зрения модификация прямого, дополнительного и обратного кодов заключается в том, что для кодирования знака числа отводится не один, а два знаковых разряда.

Пример 13. Записать в модифицированном коде:
а) положительное число $A = +1011$;
б) отрицательное число $A = -1011$.
Решение

а) $[A]^{\text{м}}_{\text{пр}} = [A]^{\text{м}}_{\text{доп}} = [A]^{\text{м}}_{\text{обр}} = 00.1011$;
б) $[A]^{\text{м}}_{\text{пр}} = 11.1011$; $[A]^{\text{м}}_{\text{доп}} = 11.0101$; $[A]^{\text{м}}_{\text{обр}} = 11.0100$.

Сложение чисел в модифицированных кодах сводится также к их арифметическому сложению, при этом оба знаковых разряда участвуют в сложении наравне со значащими разрядами. В дополнительном модифицированном коде единица переноса из старшего знакового разряда суммы теряется, а в обратном модифицированном – прибавляется к младшему разряду суммы.

Признаком переполнения при использовании модифицированных кодов является комбинация 10 (при сложении двух отрицательных чисел) или 01 (при сложении двух положительных чисел) в знаковых разрядах суммы.

Пример 14. Сложить два отрицательных числа в модифицированных дополнительных кодах: $A = -1010$, $B = -1101$ ($n = 4$).

Решение. $[A]_{\text{доп}} = 11.0110$; $[B]_{\text{доп}} = 11.0011$.

```
  11.0110    [A]доп
+ 11.0011    [B]доп
 [1]10.1001  [A + B]доп комбинация 10 указывает
                        на переполнение
  └─ отбрасывается
```

Ответ. Переполнение.

2.3 Умножение чисел с фиксированной запятой

2.3.1 Методы умножения двоичных чисел без знака

Используемая в ЭВС схема умножения похожа на известную из школьного курса процедуру умножения «столбиком» [2].

Пример на умножение «столбиком» показан на рисунке 4, а.

```
   54   Множимое
 × 14   Множитель
  216
+ 54
  756   Произведение
```

а

```
   54
 ×  4
  216  Первое частичное произведение
```

б

```
   54
 × 10
  540  Второе частичное произведение
```

в

```
  216 Первое частичное произведение
+ 540 Второе частичное произведение
  756 Произведение
```

г

а – решение примера; б – вычисление первого частичного произведения;
в – вычисление второго частичного произведения;
г – вычисление произведения

Рисунок 4 – Пример на умножение десятичных чисел

Верхнее число называется *множимым*, нижнее – *множителем*, а результат умножения – *произведением*.

По сути, умножение – просто многократно повторенная операция сложения. Вычисление произведения состоит в нахождение суммы одинаковых слагаемых, в роли которых выступает множимое. При этом количество слагаемых равно значению множителя. В нашем примере, для того чтобы получить произведение 54 × 14 = 756, пришлось бы находить сумму четырнадцати слагаемых, каждое из которых равно множимому 54. При этом процесс записи и вычисления соответствующей суммы занял бы слишком много времени. Поэтому для получения произведения используется способ умножения «столбиком». Для решения примера 54 × 14 мы сначала умножаем множимое 54 на 4. В результате получаем 216 – *первое частичное произведение* (рисунок 4, б). Затем мы умножаем множимое 54 на 1. В действительности множимое умножается на 10, как показано на рисунке 4, в. *Второе частичное произведение* равно 540. Первое и второе частичные произведения (216 и 540) складываются (рисунок 4, г). В результате мы получаем полное произведение 756. Во втором частичном произведении последний 0 обычно опускают, как показано на рисунке 4, а.

Двоичное умножение намного проще умножения десятичных чисел. В двоичной системе счисления используются только две цифры (0 и 1), поэтому правила умножения исключительно просты (см. таблицу 2). Процесс умножения двоичных чисел полностью аналогичен процессу умножения десятичных чисел. На рисунке 5 подробно иллюстрируется решение примера на умножение двоичных чисел 111 и 101.

```
Десятичные                Двоичные числа
  числа
    7                 111   Множимое
  × 5                ×101   Множитель
   35                 111   Первое частичное произведение
                    + 000    Второе частичное произведение
                      111    Третье частичное произведение
                    100011   Полное произведение
```

Рисунок 5 – Пример на умножение двоичных чисел

Сначала множимое 111 умножается на значение разряда единиц множителя. В результате получается первое частичное произведение, равное 111. Затем множимое умножается на значение разряда двоек множителя. Получаем второе частичное произведение 000. Обратите внимание, что нуль в младшем разряде второго частичного произведения отбрасывается. На третьем этапе множимое умножается на значение разряда четверок множителя. Получаем третье частичное произведение. Фактически оно равно 11100, но записывается как 111 (нули в двух младших разрядах опускаются). И наконец, первое, второе

и третье частичные произведения складываются. Их сумма равна двоичному числу 100011. Это и есть полное произведение. Для удобства слева на рисунке 5 приведено решение данного примера в десятичной системе счисления. Двоичное произведение 100011 равно десятичному произведению 35.

В общем случае [3] вычисление произведения $P\,(p_{2n-1}p_{2n-2}\ldots p_1p_0)$ двух n-разрядных двоичных чисел без знака множимого (Мн) $A\,(a_{n-1}a_{n-2}\ldots a_1a_0)$ и множителя (Мт) $B\,(b_{n-1}b_{n-2}\ldots b_1b_0)$ сводится к формированию *частичных произведений* (ЧП) W_i по одному на каждую цифру множителя, с последующим суммированием полученных ЧП. Перед суммированием каждое частичное произведение должно быть сдвинуто на один разряд относительно предыдущего согласно весу цифры множителя, которой это ЧП соответствует (рисунок 6). Перемножение двух n-разрядных двоичных чисел $P = A \times B$ приводит к получению результата, содержащего $2n$ разрядов.

	$a_3\ a_2\ a_1\ a_0$	A
	$\times b_3\ b_2\ b_1\ b_0$	B
	$w_{03}w_{02}w_{01}w_{00}$	$W_0 = Ab_0 \times 2^0$
+	$w_{13}w_{12}w_{11}w_{10}$	$W_1 = Ab_1 \times 2^1$
	$w_{23}w_{22}w_{21}w_{20}$	$W_2 = Ab_2 \times 2^2$
	$w_{33}w_{32}w_{31}w_{30}$	$W_3 = Ab_3 \times 2^3$
	$p_7\ p_6\ p_5\ p_4\ p_3\ p_2\ p_1\ p_0$	P

Рисунок 6 – Общая схема умножения для n = 4

Поскольку сомножителями являются двоичные числа, вычисление ЧП упрощается – если цифра множителя b_i равна 0, то W_i тоже равно 0, а при $b_i = 1$ частичное произведение равно множимому $W_i = A$.

Таким образом, алгоритм умножения предполагает последовательное выполнение двух операций – сложения и сдвига. Суммирование ЧП обычно производится не на завершающем этапе, а по мере их получения. Это позволяет избежать необходимости хранения всех ЧП, т. е. сокращает аппаратные затраты. Процесс получения произведения включает умножение множимого A на каждый разряд b_i множителя B. Получаемые при этом частичные произведения $W_i = Ab_i$ последовательно складываются (накапливаются), образуя *суммы частичных произведений* (частичные суммы) СЧП. Последняя сумма частичных произведений равна полному произведению.

Таким образом, процесс умножения n-разрядного двоичного множимого A на n-разрядный двоичный множитель B состоит из повторяющейся n раз последовательности умножения A на каждую очередную цифру B. Такая последовательность называется *циклом умножения*. На каждом цикле умножения сначала определяется очередное ЧП_i, далее ЧП_i прибавляется к СЧП_{i-1}, в результате чего определяется очередная сумма частичных произведений СЧП_i, и так

до получения СЧП$_n$, которое и представляет собой полное произведение, как это показано на рисунке 7.

```
   111   Множимое
  ×101   Множитель
 ------
     0   Нулевая частичная сумма
  +111   Первое частичное произведение
 ------
   111   Первая частичная сумма
 +000    Второе частичное произведение
 ------
  0111   Вторая частичная сумма
 +111    Третье частичное произведение
 ------
 100011  Полное произведение
```

Рисунок 7 – Пример двоичного умножения с образованием частичных сумм

Во всех рассмотренных примерах умножение начиналось с младших разрядов множителя. В общем случае умножение можно начинать как с младших разрядов множителя, так и со старших. На рисунке 8 подробно иллюстрируется процесс умножения двоичных чисел со старших разрядов множителя.

```
  111    Множимое
 ×101    Множитель
 ------
    0    Нулевая частичная сумма
 +111    Первое частичное произведение
 ------
  111    Первая частичная сумма
 + 000   Второе частичное произведение
 ------
  1110   Вторая частичная сумма
 +  111  Третье частичное произведение
 ------
 100011  Полное произведение
```

Рисунок 8 – Пример умножения двоичных чисел
со старших разрядов множителя

Для правильного накопления сумм частичных произведений в каждом цикле умножения множимое (очередное частичное произведение) должно сдвигаться влево (при умножении с младших разрядов множителя) или вправо (при умножении со старших разрядов множителя), при этом сумма частичных произведений должна быть неподвижна. Правильное накопление сумм частичных произведений будет происходить также в случае, если множимое (очередное частичное произведение) на каждом цикле умножения добавлять в одни и те же разряды, т. е. сделать его неподвижным, а после каждого очередного суммирования сдвигать очередную сумму частичных произведений вправо (при умножении с младших разрядов множителя) или влево (при умножении со старших разрядов множителя).

Таким образом, возможны четыре варианта реализации схемы умножения (метода умножения), которые представлены на рисунке 9.

Рисунок 9 – Методы умножения двоичных беззнаковых чисел

Методы *умножения двоичных беззнаковых чисел* основаны на представлении произведения в виде полинома [1, 6].

1. *Умножение, начиная с младших разрядов множителя, при сдвиге множимого влево и неподвижной сумме частичных произведений*.

$$P = A \times B = A(b_{n-1}2^{n-1} + \cdots + b_1 2^1 + b_0 2^0) = \\ = A2^{n-1}b_{n-1} + \cdots + A2^1 b_1 + A2^0 b_0.$$

Алгоритм умножения включает следующие шаги:

1) обнуление СЧП_0;

2) анализ младшего разряда множителя: если $b_0 = 1$, выполняется сложение $\text{ЧП}_0 = A$ с СЧП_0 и переход к шагу 3; если $b_0 = 0$, сразу переход к шагу 3;

3) сдвиг множимого влево;

4) анализ первого разряда множителя: если $b_1 = 1$, выполняется сложение $\text{ЧП}_1 = A2^1$ с СЧП_1 и переход к шагу 5; если $b_1 = 0$, непосредственно переход к шагу 5;

5) сдвиг множимого влево;

6) анализ очередного b_i разряда множителя и т. д. После анализа старшего b_{n-1} разряда множителя осуществляется последнее сложение $\text{ЧП}_{n-1} = A2^{n-1}$ с суммой частичных произведений СЧП_{n-1} (если $b_{n-1} = 1$), и процесс прекращается. Результирующая СЧП_n является искомым произведением.

2. *Умножение, начиная со старших разрядов множителя, при сдвиге суммы частичных произведений влево и неподвижном множимом.*

Применив схему Горнера, выражение для произведения можно записать следующим образом:

$$P = \left(\ldots\left(\left((0)2^1 + Ab_{n-1}\right)2^1 + Ab_{n-2}\right)2^1 + \cdots + Ab_1\right)2^1 + Ab_0.$$

Выражения в скобках в формуле представляют собой последовательные значения СЧП_i, определяемые рекуррентной формулой:

$$\text{СЧП}_i = \text{СЧП}_{i-1}2^1 + Ab_{n-i},\ \text{СЧП}_0 = 0,\ i = 1, \ldots, n.$$

Алгоритм умножения включает следующие шаги:

1) обнуление СЧП_0;

2) сдвиг СЧП_0 влево;

3) анализ старшего разряда множителя: если $b_{n-1} = 1$, выполняется сложение $\text{ЧП}_0 = A$ с $\text{СЧП}_0 2^1$ и переход к шагу 4; если $b_{n-1} = 0$, сразу переход к шагу 4;

4) сдвиг СЧП_1 влево;

5) анализ очередного b_{n-2} разряда множителя и т. д. После анализа младшего b_0 разряда множителя выполняется последнее сложение $\text{ЧП}_{n-1} = A$ с $\text{СЧП}_{n-1}2^1$ (если $b_0 = 1$), и процесс прекращается.

3. *Умножение, начиная со старших разрядов множителя, со сдвигом множимого вправо и при неподвижной сумме частичных произведений.*

$$P = A2^n 2^{-1} b_{n-1} + \cdots + A2^n 2^{-(n-1)} b_1 + A2^n 2^{-n} b_0 = \\ = V2^{-1} b_{n-1} + \cdots + V2^{-(n-1)} b_1 + V2^{-n} b_0,$$

где $V = A2^n$ – множимое, сдвинутое влево на n разрядов.

Алгоритм умножения включает следующие шаги:

1) обнуление СЧП_0;

2) сдвиг множимого вправо;

3) анализ старшего разряда множителя: если $b_{n-1} = 1$, выполняется сложение $\text{ЧП}_0 = V2^{-1}$ с СЧП_0 и переход к шагу 4; если $b_{n-1} = 0$, сразу переход к шагу 4;

4) сдвиг множимого вправо;

5) анализ очередного разряда b_{n-2} множителя и т. д. После анализа младшего разряда b_0 множителя осуществляется последнее сложение $\text{ЧП}_{n-1} = V2^{-n}$ с СЧП_{n-1} (если $b_0 = 1$), и процесс прекращается.

4. *Умножение, начиная с младших разрядов множителя, со сдвигом суммы частичных произведений вправо и при неподвижном множимом.*

Применив схему Горнера к методу 3, выражение для произведения можно записать следующим образом:

$$P = 2^{-1}\left(Vb_{n-1} + 2^{-1}\left(Vb_{n-2} + \cdots + 2^{-1}\left(Vb_1 + 2^{-1}(Vb_0 + 0)\right)\ldots\right)\right);$$

$$\text{СЧП}_i = \text{СЧП}_{i-1}2^{-1} + Vb_{i-1},\ \text{СЧП}_0 = 0,\ i = 1, \ldots, n.$$

Алгоритм умножения включает следующие шаги:

1) обнуление СЧП_0;

2) анализ младшего разряда множителя: если $b_0 = 1$, сложение $\text{ЧП}_0 = V$ с СЧП_0 и переход к шагу 3; если $b_0 = 0$, непосредственно переход к шагу 3;

3) сдвиг СЧП_1 вправо;

4) анализ очередного разряда b_1 множителя и т. д. После анализа старшего разряда b_{n-1} множителя осуществляются последнее сложение $\text{ЧП}_{n-1} = V$ с $\text{СЧП}_{n-1}2^{-1}$ (если $b_{n-1} = 1$) и последний сдвиг СЧП_n вправо, после чего процесс прекращается.

Пример 15. Умножить по первому методу два целых числа без знака: A = 1111, B = 101 (n = 4).

Решение. Умножение с младших разрядов Мт со сдвигом Мн влево.

Шаг	Мт	Мн и СЧП	Действие
0	0101	00000000	СЧП_0=0
1	010[1]	00001111 00001111 00011110	$\text{ЧП}_0 = Ab_0 = A$ $\text{СЧП}_1 = \text{СЧП}_0 + \text{ЧП}_0$ $A2^1$ (сдвиг Мн влево)
2	01[0]1	00000000 00001111 00111100	$\text{ЧП}_1 = A2^1b_1 = 0$ $\text{СЧП}_2 = \text{СЧП}_1 + \text{ЧП}_1$ $A2^2$ (сдвиг Мн влево)
3	0[1]01	00111100 01001011 01111000	$\text{ЧП}_2 = A2^2b_2 = A2^2$ $\text{СЧП}_3 = \text{СЧП}_2 + \text{ЧП}_2$ $A2^3$ (сдвиг Мн влево)
4	[0]101	00000000 [01001011] 11110000	$\text{ЧП}_3 = A2^3b_3 = 0$ $P = \text{СЧП}_4 = \text{СЧП}_3 + \text{ЧП}_3$ $A2^4$ (сдвиг Мн влево)

Пример 16. Умножить по второму методу два целых числа без знака: A = 1111, B = 101 (n = 4).

Решение. Умножение со старших разрядов Мт со сдвигом СЧП влево.

Шаг	Мт	Мн и СЧП	Действие
0	0101	00000000	$СЧП_0=0$
1	0101	00000000 0000 00000000	$СЧП_0 2^1$ (сдвиг СЧП влево) $ЧП_0=Ab_3=0$ $СЧП_1=СЧП_0 2^1+ЧП_0$
2	0101	00000000 1111 00001111	$СЧП_1 2^1$ (сдвиг СЧП влево) $ЧП_1=Ab_2=A$ $СЧП_2=СЧП_1 2^1+ЧП_1$
3	0101	00011110 0000 00011110	$СЧП_2 2^1$ (сдвиг СЧП влево) $ЧП_2=Ab_1=0$ $СЧП_3=СЧП_2 2^1+ЧП_2$
4	0101	00111100 1111 01001011	$СЧП_3 2^1$ (сдвиг СЧП влево) $ЧП_3=Ab_0=A$ $P=СЧП_4=СЧП_3 2^1+ЧП_3$

Пример 17. Умножить по третьему методу два целых числа без знака: A = 1111, B = 101 (n = 4).

Решение. Умножение со старших разрядов Мт со сдвигом Мн вправо: $V = A2^4 = 11110000$.

Шаг	Мт	Мн и СЧП	Действие
0	0101	00000000	$СЧП_0=0$
1	0101	01111000 00000000 00000000	$V2^{-1}$ (сдвиг Мн вправо) $ЧП_0=V2^{-1}b_3=0$ $СЧП_1=СЧП_0+ЧП_0$
2	0101	00111100 00111100 00111100	$V2^{-2}$ (сдвиг Мн вправо) $ЧП_1=V2^{-2}b_2=V2^{-2}$ $СЧП_2=СЧП_1+ЧП_1$
3	0101	00011110 00000000 00111100	$V2^{-3}$ (сдвиг Мн вправо) $ЧП_2=V2^{-3}b_1=0$ $СЧП_3=СЧП_2+ЧП_2$
4	0101	00001111 00001111 01001011	$V2^{-4}$ (сдвиг Мн вправо) $ЧП_3=V2^{-4}b_0=V2^{-4}$ $P=СЧП_4=СЧП_3+ЧП_3$

Пример 18. Умножить по четвертому методу два целых числа без знака: A = 111, B = 101 (n = 3).

Решение. Умножение с младших разрядов Мт со сдвигом СЧП вправо: $V = A2^4 = 111000$.

Шаг	Мт	Мн и СЧП	Действие
0	101	000000	$СЧП_0=0$
1	10[1]	111 111000 011100	$ЧП_0=Vb_0=V$ $СЧП_1=СЧП_0+ЧП_0$ $СЧП_1 2^{-1}$ (сдвиг СЧП вправо)
2	1[0]1	000 011100 001110	$ЧП_1=Vb_1=0$ $СЧП_2=СЧП_1 2^{-1}+ЧП_1$ $СЧП_2 2^{-1}$ (сдвиг СЧП вправо)
3	[1]01	111 1000110 100011	$ЧП_2=Vb_2=V$ $СЧП_3=СЧП_2 2^{-1}+ЧП_2$ $P=СЧП_3 2^{-1}$ (сдвиг СЧП вправо)

Рассмотрим организацию устройства, реализующего умножение по данному методу (рисунок 10) [2].

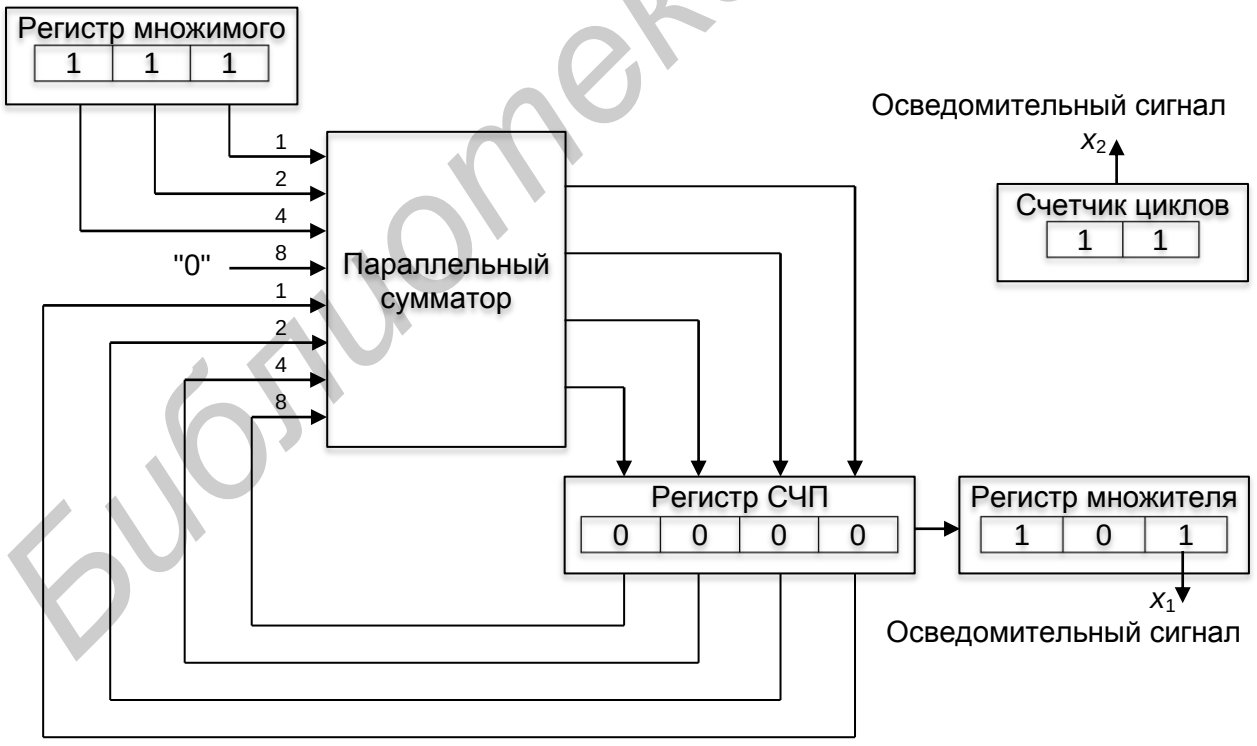


Рисунок 10 – Операционная часть устройства умножения

В исходном состоянии множимое 111 загружено в *регистр множимого*, показанный в левой верхней части рисунка, *регистр СЧП* очищен (установлен

в состояние 0000), и множитель 101 загружен в *регистр множителя*, показанный в нижней правой части рисунка. Регистр СЧП и регистр множителя при выполнении сдвига вправо рассматриваются как единый регистр. Это отражено на рисунке стрелкой, соединяющей оба регистра. Кроме того, в состав устройства умножения входит счетчик циклов, в который в исходном состоянии загружается число циклов умножения 11 (десятичное число 3). Счетчик циклов выдает осведомительный сигнал, который устанавливается в 1, когда содержимое счетчика становится равным 00.

Рассмотрим подробно процедуру умножения. На рисунке 11 поэтапно показан процесс умножения двоичного числа 111 (множимое) на двоичное число 101 (множитель).

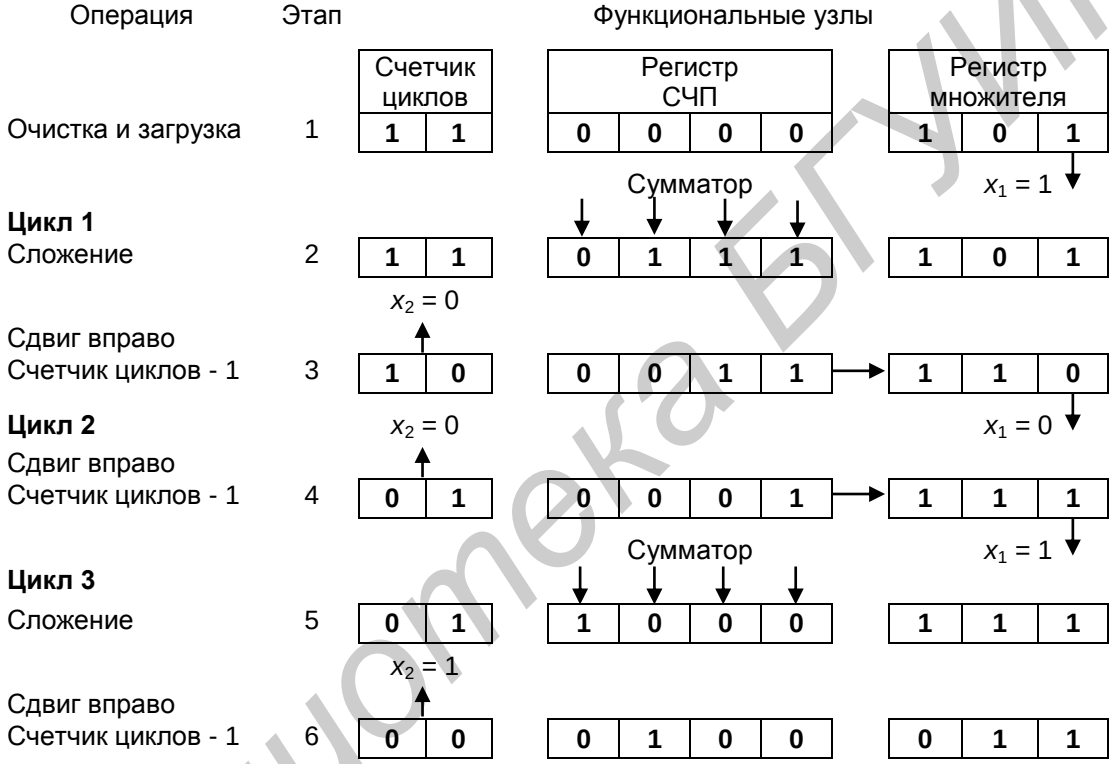


Рисунок 11 – Поэтапный процесс выполнения умножения

На *первом этапе* осуществляется очистка регистра СЧП, а также загрузка регистров множимого, множителя и счетчика циклов. Далее начинается первый цикл умножения, который включает два этапа. На *втором этапе* выполняется сложение содержимого регистра СЧП 0000 и регистра множимого, так как осведомительный сигнал x_1 (младший разряд множителя) равен единице. В результате в регистре СЧП получается первая сумма частичных произведений. На *третьем этапе* содержимое регистра СЧП и регистра множителя (при сдвиге вправо они рассматриваются как единый регистр) сдвигаются на один разряд вправо. При этом уходит из регистра и теряется единица крайнего правого разряда множителя. На этом этапе содержимое счетчика циклов становится равным 10, поэтому осведомительный сигнал $x_2 = 0$, и, следовательно, можно начинать следующий цикл умножения (т. е. процесс умножения еще не закон-

чен и его надо продолжать). Второй цикл умножения включает только один этап, на котором содержимое регистра СЧП и регистра множителя сдвигаются на один разряд вправо, а содержимое счетчика циклов уменьшается на единицу. При этом уходит из регистра и теряется бит разряда двоек множителя. В этом цикле отсутствует операция сложения содержимого регистра СЧП и регистра множителя. Это связано с тем, что при переходе к циклу 2 осведомительный сигнал $x_1 = 0$. Это означает, что второй разряд множителя равен 0. В этом случае, как это делалось в рассмотренных выше примерах, к содержимому регистра СЧП необходимо было бы добавить код 000. Но эта операция не приводит к изменению содержимого регистра СЧП, а это, в свою очередь, означает, что на самом деле никакого сложения производить не нужно. В этом цикле содержимое счетчика циклов становится равным 01, так что осведомительный сигнал $x_2 = 0$. Поэтому осуществляется переход к циклу 3, в котором выполняются операции, аналогичные первому циклу. На этом цикле выполнение умножения заканчивается, так как содержимое счетчика циклов становится равным 00 (осведомительный сигнал $x_1 = 1$). При этом младшие три разряда регистра СЧП и регистр множителя содержат полное произведение 100011 (десятичное число 35).

Варианты со сдвигом множимого на практике не используются, поскольку для их реализации регистр множимого, регистр СЧП и сумматор должны иметь разрядность $2n$.

2.3.2 Умножение чисел со знаком

Умножение чисел со знаком может выполняться как в прямом, так и в дополнительном коде [3]. Однако умножение чисел в прямом коде выполняется несколько проще, чем в дополнительном. При умножении чисел в прямом коде перемножаются модули сомножителей *A* и *B*, а знак произведения определяется путем сложения по модулю двух знаковых разрядов сомножителей и приписывается произведению после завершения перемножения модулей *A* и *B*.

Во всех ЭВМ общепринято представлять числа со знаком в форме с фиксированной запятой в дополнительном коде. По этой причине более предпочтительны варианты, не требующие преобразования сомножителей и обеспечивающие вычисления непосредственно в дополнительном коде. При перемножении чисел в дополнительном коде знак произведения формируется автоматически вместе с произведением.

Выполнение операции умножения в дополнительном коде осуществляется в соответствии с *алгоритмом Робертсона*. Алгоритм Робертсона аналогичен рассмотренному ранее алгоритму для чисел без знака по четвертому методу (метод со сдвигом суммы частичных произведений вправо), но учитывает две принципиальные особенности, обусловленные дополнительным кодом.

Первая особенность связана с тем, что в процессе умножения могут получаться как положительные, так и отрицательные суммы частичных произведений. В этом случае для получения корректного результата должна выполняться операция арифметического сдвига вправо для суммы частичных произведе-

ний – освобождающаяся при сдвиге позиция слева должна заполняться не нулем, а значением знакового разряда сдвигаемого числа. При получении СЧП может возникнуть ситуация, аналогичная переполнению при сложении чисел, т. е. старший разряд значащей части занимает позицию знака (позиция $2n$), а знак переходит в позицию с номером $2n + 1$. Поэтому при выполнении операции сдвига вправо суммы частичных произведений удобно пользоваться следующим правилом: в освобождающуюся при сдвиге позицию слева (позиция знака) заносить значение знакового разряда множимого, т. е. при положительном множимом – нуль, а при отрицательном – единицу. Однако следует учитывать, что это правило начинает действовать только с момента, когда среди анализируемых разрядов множителя появляется первая единица.

Вторая особенность заключается в том, что выполнение операции умножения в дополнительном коде распадается на два случая.

1. Множимое произвольного знака, множитель положительный.

Процедура умножения протекает аналогично умножению беззнаковых чисел с учетом сделанного замечания об арифметическом сдвиге СЧП. В случае отрицательного множимого результат умножения отрицательный, поэтому он получается в дополнительном коде.

2. Множимое произвольного знака, множитель отрицательный.

Так как множитель отрицателен, он записывается в дополнительном коде $[B]_{\text{доп}} = 2^{n+1} - |B|$, и в цифровых разрядах кода будет представлено число $2^n - |B|$. При типовом умножении, как и в случае положительного множителя (т. е. при анализе только значащих разрядов множителя), вместо истинного произведения P получаем псевдопроизведение

$$P' = A(2^n - |B|) = -A|B| + A2^n.$$

Псевдопроизведение P' больше истинного произведения P на величину $A2^n$, что и необходимо учитывать при формировании окончательного результата. Для этого выполнятся коррекция результата: после последнего сдвига из полученного псевдопроизведения вычитается избыточная величина $A2^n$.

В алгоритме Робертсона это учитывается следующим образом. Знаковый разряд участвует в операции умножения наряду со значащими разрядами. Анализ значащих разрядов множителя от младшего b_0 до старшего b_{n-1}, а также обусловленные этим анализом сложения и сдвиги выполняются стандартным образом. Разряд множителя b_n содержит знак и для отрицательного множителя равен 1. В стандартном случае это предполагает прибавление к СЧП множимого, сдвинутого влево на n разрядов, – $V = A2^n$. Вместо этого на шаге анализа знакового разряда множимое, сдвинутое влево на n разрядов, не прибавляется к СЧП, а вычитается из нее. Напомним, что такая коррекция нужна только в случае отрицательного множителя.

Таким образом, алгоритм умножения чисел со знаком можно сформулировать следующим образом:

1) множимое и множитель представляются в дополнительном коде;

2) исходное значение суммы частичных произведений СЧП_0 принимается равным нулю;

3) анализируется очередной значащий разряд множителя b_i (анализ начинается с младшего разряда). Если $\text{b}_i = 1$, то к СЧП_i прибавляется множимое, сдвинутое влево на n разрядов $\text{ЧП}_i = \text{V} = A2^n$ в том коде, в котором оно представлено. Если $\text{b}_i = 0$, то прибавление не производится. В результате получается очередная сумма частичных произведений СЧП_{i+1};

4) производится арифметический сдвиг суммы частичных произведений вправо на один разряд $\text{СЧП}_{i+1}2^{-1}$;

5) пункты 3 и 4 последовательно повторяются для всех значащих разрядов множителя. После анализа старшего значащего разряда b_{n-1} множителя осуществляются последнее сложение $\text{ЧП}_{n-1} = V$ с $\text{СЧП}_{n-1}2^{-1}$ (если $b_{n-1} = 1$) и последний сдвиг СЧП_n вправо;

6) анализируется знаковый разряд множителя. Если он равен единице (множитель отрицательный), то из СЧП_n2^{-1} вычитается $V = A2^n$ (прибавляется V с обратным знаком, представленное в дополнительном коде, $[-V]_{\text{доп}} = [-A]_{\text{доп}}2^n$). Если знаковый разряд множителя равен нулю (множитель положительный), то вычитание не производится. В результате получается произведение, представленное в дополнительном коде.

Пример 19. Умножить два целых числа со знаком: $A = -13$, $B = +10$ ($n = 4$).

Решение. $[A]_{\text{доп}} = 1.0011$; $[B]_{\text{доп}} = 0.1010$; $V = [A]_{\text{доп}}2^4 = 1.0011 \cdot 2^4$.

Шаг	Мт	Мн и СЧП	Действие
0	1010	0.00000000	СЧП_0=0
1	101**0**	0.0000 0.00000000 0.00000000	$\text{ЧП}_0 = Vb_0$=0 $\text{СЧП}_1 = \text{СЧП}_0 + \text{ЧП}_0$ $\text{СЧП}_1 2^{-1}$ (сдвиг СЧП вправо)
2	10**1**0	1.0011 1.00110000 1.10011000	$\text{ЧП}_1 = Vb_1 = V$ $\text{СЧП}_2 = \text{СЧП}_1 2^{-1} + \text{ЧП}_1$ $\text{СЧП}_2 2^{-1}$ (сдвиг СЧП вправо)
3	1**0**10	0.0000 1.10011000 1.11001100	$\text{ЧП}_2 = Vb_2$=0 $\text{СЧП}_3 = \text{СЧП}_2 2^{-1} + \text{ЧП}_2$ $\text{СЧП}_3 2^{-1}$ (сдвиг СЧП вправо)
4	**1**010	1.0011 **1**0.11111100 1.01111110 *отбрасывается*	$\text{ЧП}_3 = Vb_3 = V$ $\text{СЧП}_4 = \text{СЧП}_3 2^{-1} + \text{ЧП}_3$ $[P]_{\text{доп}} = \text{СЧП}_4 2^{-1}$ (сдвиг СЧП вправо)
		1.10000010	$[P]_{\text{пр}}$

Ответ. $P = -130$.

Обратите внимание, что в соответствии с правилами сложения в дополнительных кодах возникающая в примере 19 на шаге 4 в результате сложения единица переноса из знакового разряда отбрасывается.

Пример 20. Умножить два целых числа со знаком: A = +13, B = –10 (n = 4).

Решение
$[A]_{\text{доп}} = 0.1101$; $[B]_{\text{доп}} = 1.0110$; $V = [A]_{\text{доп}} 2^4 = 0.1101 \cdot 2^4$.
Величина коррекции $-A2^4 = [-A]_{\text{доп}} 2^4 = 1.0011 \cdot 2^4$.

Шаг	Мт	Мн и СЧП	Действие
0	0110	0.00000000	$СЧП_0$=0
1	011[0]	0.0000 0.00000000 0.00000000	$ЧП_0$=Vb_0=0 $СЧП_1$=$СЧП_0$+$ЧП_0$ $СЧП_1 2^{-1}$ (сдвиг СЧП вправо)
2	01[1]0	0.1101 0.11010000 0.01101000	$ЧП_1$=Vb_1=V $СЧП_2$=$СЧП_1 2^{-1}$+$ЧП_1$ $СЧП_2 2^{-1}$ (сдвиг СЧП вправо)
3	0[1]10	0.1101 1.00111000 0.10011100	$ЧП_2$=Vb_2=0 $СЧП_3$=$СЧП_2 2^{-1}$+$ЧП_2$ $СЧП_3 2^{-1}$ (сдвиг СЧП вправо)
4	[0]110	0.0000 0.10011100 0.01001110	$ЧП_3$=Vb_3=V $СЧП_4$=$СЧП_3 2^{-1}$+$ЧП_3$ P'=$СЧП_4 2^{-1}$ (сдвиг СЧП вправо)
Коррек-ция		1.0011 1.01111110	$[-A]_{\text{доп}} 2^4$ $[P]_{\text{доп}}$
		1.10000010	$[P]_{\text{пр}}$

Ответ. $P = -130$.

Пример 21. Умножить два целых числа со знаком: A = –13, B = –10 (n = 4).

Решение
$[A]_{\text{доп}} = 1.0011$; $[B]_{\text{доп}} = 1.0110$; $V = [A]_{\text{доп}} 2^4 = 1.0011 \cdot 2^4$.
Величина коррекции $-A2^4 = [-A]_{\text{доп}} 2^4 = 0.1101 \cdot 2^4$.

Шаг	Мт	Мн и СЧП	Действие
0	0110	0.00000000	$СЧП_0=0$
1	011[0]	0.0000 0.00000000 0.00000000	$ЧП_0=Vb_0=0$ $СЧП_1=СЧП_0+ЧП_0$ $СЧП_1 2^{-1}$ (сдвиг СЧП вправо)
2	01[1]0	1.0011 1.00110000 1.10011000	$ЧП_1=Vb_1=V$ $СЧП_2=СЧП_1 2^{-1}+ЧП_1$ $СЧП_2 2^{-1}$ (сдвиг СЧП вправо)
3	0[1]10	1.0011 [1]0.11001000 1.01100100 *отбрасывается*	$ЧП_2=Vb_2=0$ $СЧП_3=СЧП_2 2^{-1}+ЧП_2$ $СЧП_3 2^{-1}$ (сдвиг СЧП вправо)
4	[0]110	0.0000 1.01100100 1.10110010	$ЧП_3=Vb_3=V$ $СЧП_4=СЧП_3 2^{-1}+ЧП_3$ $P'=СЧП_4 2^{-1}$ (сдвиг СЧП вправо)
Коррек-ция		0.1101 [1]0.10000010 *отбрасывается*	$[-A]_{доп} 2^4$ $[P]_{доп}=[P]_{пр}$

Ответ. $P = +130$.

2.3.3 Методы ускорения умножения

Методы ускорения умножения можно условно разделить на логические и аппаратные [3]. Логические методы позволяют сократить время вычисления за счет более эффективных алгоритмов умножения, позволяющих уменьшить количество сложений в ходе умножения. Аппаратные методы направлены на схемное сокращение времени вычисления и суммирования частичных произведений.

Логические методы ускорения умножения. Логические подходы к ускорению умножения можно подразделить на две группы:

– применение избыточных систем счисления;

– применение систем счисления с основанием большим двух.

Рассмотрим первую группу логических методов – умножение с использованием избыточных систем счисления. Наиболее известным и распространенным представителем этой группы является алгоритм Бута.

Алгоритм Бута [3]. В основе алгоритма Бута лежит следующее соотношение, характерное для последовательностей двоичных цифр:

$$2^m + 2^{m-1} + \cdots + 2^k = 2^{m+1} - 2^k.$$

где m и k – номера крайних разрядов в группе из последовательных единиц. Например, $011110 = 2^5 - 2^1$. Это означает, что при наличии во множителе групп из нескольких единиц (комбинаций вида 011110), последовательное добавление к СЧП множимого с нарастающим весом (от 2^k до 2^m) можно заменить вычитанием из СЧП множимого с весом 2^k и прибавлением к СЧП множимого с весом 2^{m+1}.

Алгоритм Бута сводится к перекодированию множителя из системы $\{0, 1\}$ в избыточную систему $\{\bar{1}, 0,1\}$. Такое перекодирование часто называют перекодированием Бута (Booth recoding). В записи множителя в новой системе 1 означает добавление множимого к сумме частичных произведений, $\bar{1}$ – вычитание множимого и 0 не предполагает никаких действий. Во всех случаях после очередной итерации производится сдвиг множимого влево или суммы частичных произведений вправо. Реализация алгоритма предполагает последовательный в направлении справа налево анализ пар разрядов множителя – текущего b_i и предшествующего b_{i-1} ($b_i\ b_{i-1}$). Для младшего разряда множителя ($i = 0$) считается, что предшествующий разряд равен 0, т. е. имеет место пара $b_0 0$. На каждом шаге i ($i = 0, 1,\ldots, n - 1$) анализируется текущая комбинация $b_i\ b_{i-1}$.

Алгоритм предполагает три операции: сдвиг, сложение и вычитание. Комбинация 10 означает начало цепочки последовательных единиц, и в этом случае производится вычитание множимого из СЧП. Комбинация 01 соответствует завершению цепочки единиц, и в этом случае множимое прибавляется к СЧП. Комбинация 00 свидетельствует об отсутствии цепочки единиц, а 11 – о нахождении внутри такой цепочки. В обоих случаях никакие арифметические операции не производятся. По завершении описанных действий осуществляется сдвиг множимого влево либо суммы частичных произведений вправо, и цикл повторяется для следующей пары разрядов множителя. Если последняя пара разрядов множителя $b_{n-1}\ b_{n-2}$ равна комбинациям 10 или 11, должна быть выполнена коррекция результата, которая заключается в добавлении множимого к последней сумме частичных произведений.

Процесс умножения по алгоритму Бута рассмотрим на следующих примерах. В приведенных примерах используется вариант умножения со сдвигом множимого влево. Операция вычитания, как это принято в реальных умножителях, выполняется путем сложения с множителем, взятым с противоположным знаком и представленным в дополнительном коде. При этом сложение осуществляется по правилам сложения в дополнительных кодах, т. е. единица переноса из старшего разряда отбрасывается.

Пример 22. Умножить два целых числа без знака по алгоритму Бута: $A = 6$, $B = 7$ ($n = 4$).

Решение. $A = 00000110$, $[-A]_{\text{доп}} = 11111010$.

$B = 0111$. После перекодирования Мт 0111 приобретает вид $100\bar{1}$.

Шаг	Мт	Мн и СЧП	Действие
0	0111	00000000	$СЧП_0=0$
1	011[10] -1	11111010 11111010 00001100	Начало цепочки единиц (-Мн) $СЧП_1$ Сдвиг Мн влево
2	01[11] 0	-------- 11111010 00011000	Внутри цепочки единиц $СЧП_2$ Сдвиг Мн влево
3	0[11]1 0	-------- 11111010 00110000	Внутри цепочки единиц $СЧП_3$ Сдвиг Мн влево
4	[01]11 1	00110000 1←[00101010] отбрасывается	Конец цепочки единиц (+Мн) $P=СЧП_4=6\times7=42$

Пример 23. Умножить два целых числа без знака по алгоритму Бута: $A = 6$, $B = 14$ ($n = 4$).

Решение. $A = 00000110$, $[-A]_{доп} = 11111010$.

$B = 1110$. После перекодирования Мт 1110 приобретает вид $00\bar{1}0$.

Шаг	Мт	Мн и СЧП	Действие
0	1110	00000000	$СЧП_0=0$
1	111[00] 0	-------- 00000000 00001100	Отсутствие цепочки единиц $СЧП_1$ Сдвиг Мн влево
2	11[10] -1	11110100 11110100 00011000	Начало цепочки единиц (-Мн) $СЧП_2$ Сдвиг Мн влево
3	1[11]0 0	-------- 11110100 00110000	Внутри цепочки единиц $СЧП_3$ Сдвиг Мн влево
4	[11]10 0	-------- 11110100 01100000	Внутри цепочки единиц $СЧП_4$ Сдвиг Мн влево
Коррек-ция		01100000 [1]01010100 отбрасывается	+Мн $P=6\times14=84$

Помимо сокращения числа сложений (вычитаний) этот алгоритм имеет еще одно достоинство: он в равной степени применим как к числам без знака, так и к числам со знаком. Процесс умножения по алгоритму Бута чисел со знаком отличается тем, что знаковый разряд множителя участвует в анализе наравне со значащими разрядами. При этом, поскольку знаковый разряд участвует в процессе умножения, коррекцию результата производить не нужно.

Пример 24. Умножить два целых числа со знаком по алгоритму Бута: $A = -6$, $B = 14$ ($n = 4$).

Решение. $[A]_{\text{пр}} = 1.00000110$, $[A]_{\text{доп}} = 1.11111010$,
$[-A]_{\text{доп}} = 0.00000110$.
$[B]_{\text{пр}} = [B]_{\text{доп}} = 0.1110$. После перекодирования Мт 0.1110 приобретает вид $100\bar{1}0$.

Шаг	Мт	Мн и СЧП	Действие
0	0.1110	0.00000000	$СЧП_0$=0
1	0.111[00] 0	---------- 0.00000000 1.11110100 0.00001100	Отсутствие цепочки единиц $СЧП_1$ Сдвиг Мн влево Сдвиг -Мн влево
2	0.11[10] -1	0.00001100 0.00001100 1.11101000 0.00011000	Начало цепочки единиц (-Мн) $СЧП_2$ Сдвиг Мн влево Сдвиг -Мн влево
3	0.1[11]0 0	---------- 0.00001100 1.11010000 0.00110000	Внутри цепочки единиц $СЧП_3$ Сдвиг Мн влево Сдвиг -Мн влево
4	0.[11]10 0	---------- 0.00001100 1.10100000 0.01100000	Внутри цепочки единиц $СЧП_4$ Сдвиг Мн влево Сдвиг -Мн влево
5	[0.1]011 1	1.10100000 [1.10101100] 1.01010100	Конец цепочки единиц (+Мн) $[P]_{\text{доп}}$ $[P]_{\text{пр}}$=(-6)×14=-84

Пример 25. Умножить два целых числа со знаком по алгоритму Бута: $A = -6$, $B = -9$ ($n = 4$).

Решение. $[A]_{\text{пр}} = 1.00000110$, $[A]_{\text{доп}} = 1.11111010$,

$[-A]_{\text{доп}} = 0.00000110.$

$[B]_{\text{пр}} = 1.1001,\ [B]_{\text{доп}} = 1.0111.$ После перекодирования Мт 1.0111 приобретает вид $\bar{1}100\bar{1}$.

Шаг	Мт	Мн и СЧП	Действие
0	1.0111	0.00000000	$СЧП_0$=0
1	1.011[10] -1	0.00000110 0.00000110 1.11110100 0.00001100	Начало цепочки единиц (-Мн) $СЧП_1$ Сдвиг Мн влево Сдвиг -Мн влево
2	1.01[11] 0	---------- 0.00000110 1.11101000 0.00011000	Внутри цепочки единиц $СЧП_2$ Сдвиг Мн влево Сдвиг -Мн влево
3	1.0[11]1 0	---------- 0.00000110 1.11010000 0.00110000	Внутри цепочки единиц $СЧП_3$ Сдвиг Мн влево Сдвиг -Мн влево
4	1.[01]11 1	1.11010000 1.11010110 1.10100000 0.01100000	Конец цепочки единиц (+Мн) $СЧП_4$ Сдвиг Мн влево Сдвиг -Мн влево
5	[1.0]011 -1	0.01100000 [1]←0.00110110 *отбрасывается*	Начало цепочки единиц (-Мн) $[P]_{\text{доп}}=[P]_{\text{пр}}=(-6)\times(-9)=54$

Надо помнить, что при использовании варианта умножения со сдвигом суммы частичных произведений вправо в освобождающийся слева разряд заносится значение знакового разряда, т. е. выполняется арифметический сдвиг СЧП вправо.

Пример 26. Умножить, используя варианта умножения со сдвигом суммы частичных произведений вправо, два целых числа со знаком по алгоритму Бута: $A = -6$, $B = -9$ ($n = 4$).

Решение. $[A]_{\text{пр}} = 1.0110$, $[A]_{\text{доп}} = 1.1010$, $[-A]_{\text{доп}} = 0.0110$.

$[B]_{\text{пр}} = 1.1001,\ [B]_{\text{доп}} = 1.0111.$ После перекодирования Мт 1.0111 приобретает вид $\bar{1}100\bar{1}$.

Шаг	Мт	Мн и СЧП	Действие
0	1.0111	0.00000000	$СЧП_0=0$
1	1.011[10] -1	0.0110 0.01100000 0.00110000	Начало цепочки единиц (–Мн) $СЧП_1$ Сдвиг $СЧП_1$ вправо
2	1.01[11] 0	---------- 0.00110000 0.00011000	Внутри цепочки единиц $СЧП_2$ Сдвиг $СЧП_2$ вправо
3	1.0[11]1 0	---------- 0.00011000 0.00001100	Внутри цепочки единиц $СЧП_3$ Сдвиг $СЧП_3$ вправо
4	1.[01]11 1	1.1010 1.10101100 1.11010110	Конец цепочки единиц (+Мн) $СЧП_4$ Сдвиг $СЧП_4$ вправо
5	[1.0]011 -1	0.0110 1←[0.00110110] отбрасывается	Начало цепочки единиц (–Мн) $[P]_{доп}=[P]_{пр}=(-6)\times(-9)=54$

Модифицированный алгоритм Бута [3]. На практике большее распространение получил модифицированный алгоритм Бута, в котором производится перекодировка множителя из стандартной двоичной системы $\{0, 1\}$ в избыточную систему $\{\bar{2}, \bar{1}, 0,1,2\}$. В этой системе каждое число обозначает коэффициент, на который умножается множимое перед добавлением к СЧП: $\bar{2}$ – вычитание удвоенного множимого из СЧП, $\bar{1}$ – вычитание множимого, 2 – прибавление удвоенного множимого, 1 – прибавление множимого и 0 не предполагает никаких действий. В процессе умножения одновременно анализируются три разряда множителя $b_{i+1}b_ib_{i-1}$ – два текущих и старший разряд из предыдущей тройки. В зависимости от комбинации 0 и 1 в этих разрядах выполняется прибавление или вычитание множимого, прибавление или вычитание удвоенного множимого, либо никакие действия не производятся в соответствии с таблицей 5.

Такая модификация позволяет еще больше сократить количество операций сложения, выполняемых в процессе умножения. Например, как видно из таблицы 5, при комбинации 010 в соответствии с модифицированным алгоритмом необходимо выполнить только одну операцию сложения – прибавить к СЧП множимое. В то же время при такой комбинации стандартный алгоритм Бута предполагает выполнение двух операций: вычесть из СЧП множимое и после сдвига множимого или СЧП добавить множимое.

Таблица 5

b_{i+1}	b_i	b_{i-1}	Код {–2–1 0 1 2}	Выполняемое действие
0	0	0	0	Не выполнять никаких действий
0	0	1	1	Прибавить к СЧП множимое
0	1	0	1	Прибавить к СЧП множимое
0	1	1	2	Прибавить к СЧП удвоенное множимое
1	0	0	–2	Вычесть из СЧП удвоенное множимое
1	0	1	–1	Вычесть из СЧП множимое
1	1	0	–1	Вычесть из СЧП множимое
1	1	1	0	Не выполнять никаких действий

Пример 27. Умножить два целых числа со знаком по модифицированному алгоритму Бута: $A = 26$, $B = -18$ ($n = 5$).

Решение. $[A]_{\text{пр}} = [A]_{\text{доп}} = 0.0000011010$, $[-A]_{\text{доп}} = 1.1111100110$, $[-2A]_{\text{доп}} = 1.1111001100$.

$[B]_{\text{пр}} = 1.10010$, $[B]_{\text{доп}} = 1.01110$. После перекодирования Мт 1.01110 приобретает вид $\bar{1}0\bar{2}$.

Шаг	Мт	Мн и СЧП	Действие
0	1.01110	0.0000000000	$СЧП_0$=0
1	1.011[100] -2	1.1111001100 1.1111001100 0.0001101000	-2Мн $СЧП_1$ Сдвиг Мн влево на 2 разряда
2	1.0[111]0 0	---------- 1.1111001100 0.0110100000	Нет действия $СЧП_2$ Сдвиг Мн влево на 2 разряда
3	[1.01]110 -1	1.1001100000 [1]←[1.1000101100] отбрасывается	-Мн $СЧП_3=[P]_{\text{доп}}$
		1.0111010100	$[P]_{\text{пр}}=26\times(-18)=468$

Алгоритм Лемана [3]. Еще большее сокращение количества сложений может дать модификация алгоритма Бута, предложенная Леманом. Суть модификации заключается в следующем:

– если две группы нулей разделены единицей, стоящей в *k*-й позиции, то вместо вычитания в *k*-й позиции и сложения в (*k*+1)-й позиции достаточно выполнить только сложение в *k*-й позиции;

– если две группы единиц разделены нулем, стоящим в *k*-й позиции, то вместо сложения в *k*-й позиции и вычитания в (*k*+1)-й позиции достаточно выполнить только вычитание в *k*-й позиции.

Обработка двух разрядов множителя за шаг [3]. Из второй группы логических методов приведем метод умножения с обработкой за шаг двух разрядов множителя. В принципе это более эффективный вариант алгоритма Бута. Анализ множителя начинается с младших разрядов. В зависимости от входящей двухразрядной комбинации предусматриваются следующие действия:

00 – сдвиг на два разряда вправо суммы частичных произведений;

01 – к СЧП прибавляется множимое, после чего СЧП сдвигается на 2 разряда вправо;

10 – к СЧП прибавляется удвоенное множимое, СЧП сдвигается на 2 разряда вправо;

11 – из СЧП вычитается множимое, СЧП сдвигается на 2 разряда вправо. Полученный результат должен быть скорректирован на следующем шаге, что фиксируется специальным признаком коррекции.

Так как в случае пары 11 из СЧП вычитается одинарное множимое вместо прибавления утроенного, для корректировки результата к СЧП перед выполнением сдвига надо было бы прибавить учетверенное множимое. Но после сдвига на два разряда вправо СЧП уменьшается в четыре раза, так что на следующем шаге достаточно добавить одинарное множимое. Это учитывается при обработке следующей пары разрядов множителя, путем обработки пары 00 как 01, 01 – как 10, 10 – как 11, а 11 – как 00. В последних двух случаях фиксируется признак коррекции.

Правила обработки пар разрядов множителя с учетом признака коррекции приведены в таблице 6.

Таблица 6

Пара разрядов множителя	Признак коррекции из предыдущей пары	Признак коррекции в следующую пару	Выполняемое действие
00	0	0	–
01	0	0	+Мн
10	0	0	+2Мн
11	0	1	–Мн
00	1	0	+Мн
01	1	0	+2Мн
10	1	1	–Мн
11	1	1	–

После окончания процесса умножения требуется корректировка результата, если старшая пара разрядов множителя равна 11 или 10, и состояние признака коррекции единичное. В этом случае к полученному произведению должно быть добавлено множимое.

Данный метод применим и для варианта умножения со сдвигом множимого влево (при этом множимое сдвигается на 2 разряда).

Пример 28. Умножить два целых числа без знака методом умножения на два разряда множителя: $A = 5$, $B = 13$ ($n = 4$).

Решение. $A = 00000101$, $[-A]_{\text{доп}} = 11111011$, $B = 1101$.

Шаг	Мт	Мн и СЧП	Действие
0	1101	00000000	$СЧП_0$=0
1	11**01**	00000101 00000101 00010100	+Мн $СЧП_1$ Сдвиг Мн влево на 2 разряда
2	**11**01	11101100 11110001 01010000	-Мн (сдвинутое) $СЧП_2$ Сдвиг Мн влево на 2 разряда
Коррекция (старшая пара Мт равна 11)		01010000 1←01000001 отбрасывается	+Мн P=5×13=65

Аппаратные методы ускорения умножения. Аппаратные методы ускорения умножения сводятся:

– к уменьшению времени распространения переносов при суммировании частичных произведений;

– к параллельному вычислению частичных произведений.

Методы первой группы используют более эффективные способы суммирования ЧП, исключающие затраты времени на распространение переносов. Достигается это за счет представления ЧП в избыточной форме, благодаря чему суммирование двух чисел не связано с распространением переноса вдоль всех разрядов числа. Наиболее употребительной формой такого избыточного кодирования является так называемая форма с *сохранением переноса*. В ней каждый разряд числа представляется двумя битами *cs* – перенос (*c*) и сумма (*s*). При суммировании двух чисел в форме с сохранением переноса перенос распространяется не далее чем на один разряд. Это делает процесс суммирования значительно более быстрым, чем в случае сложения с распространением переноса вдоль всех разрядов числа.

Вторая возможность ускорения операции умножения заключается в параллельном вычислении всех частичных произведений. Если рассмотреть общую схему умножения, то нетрудно заметить, что отдельные разряды ЧП представляют собой произведения вида $a_i b_j$, т. е. произведение определенного бита множимого на определенный бит множителя. Это позволяет вычислить все биты частичных произведений одновременно с помощью n^2 схем И.

Различия схем умножения с параллельным вычислением ЧП проявляются в основном в способе суммирования полученных частичных произведений. С этих позиций используемые схемы умножения можно подразделить на *матричные* и *с древовидной структурой*. В обоих вариантах суммирование осуществляется с помощью массива взаимосвязанных одноразрядных сумматоров. В матричных умножителях сумматоры организованы в виде матрицы, а в древовидных они реализуются в виде дерева того или иного типа. Различия в рамках каждой из этих групп выражаются в количестве используемых сумматоров, их виде и способе распространения переносов, возникающих в процессе суммирования.

2.4 Деление чисел с фиксированной запятой

2.4.1 Методы деления двоичных чисел без знака

Деление – несколько более сложная операция, чем умножение, но базируется на тех же принципах. Основу составляет общепринятый способ деления с помощью операций вычитания или сложения и сдвига [3].

Задача сводится к вычислению частного Q и остатка S:

$$Q = int\left(\frac{Z}{D}\right),\ \ S = Z - QD,\ \ S < D,$$

где Z – делимое;

D – делитель.

Деление выражается как последовательность вычитаний делителя (Дт) сначала из делимого (Дм), а затем из образующихся в процессе деления *частичных остатков* (ЧО). Делимое $Z(z_{2n-1}z_{2n-2}\ldots z_1 z_0)$ обычно представляется двойным словом ($2n$ разрядов), делитель $D(d_{n-1}d_{n-2}\ldots d_1 d_0)$, частное $Q(q_{n-1}q_{n-2}\ldots q_1 q_0)$ и остаток $S(s_{n-1}s_{n-2}\ldots s_1 s_0)$ имеют разрядность n.

Операция выполняется за n итераций и может быть описана следующим образом:

$$i = 1, \ldots, n,$$

$$S^{(i)} = 2S^{(i-1)} - q_{n-i}(2^n D),$$

$$S^{(0)} = Z,$$

$$S^{(n)} = 2^n S,$$

$$q_{n-i} = \begin{cases} 1, \text{ если } \left(2S^{(i-1)} - 2^n D\right) \geq 0, \\ 0, \text{ если } \left(2S^{(i-1)} - 2^n D\right) < 0. \end{cases}$$

После n итераций получается

$$S^{(n)} = 2^n S^{(0)} - Q(2^n D) = 2^n (Z - QD) = 2^n S.$$

Частное от деления $2n$-разрядного числа на n-разрядное может содержать более чем n разрядов, что означает возникновение переполнения. В этом случае перед выполнением деления необходима проверка условия

$$Z < (2^n - 1)D + D = 2^n D.$$

Из выражения следует, что переполнения не будет, если число, содержащееся в старших n разрядах делимого, меньше делителя.

Перед выполнением деления с фиксированной запятой производится пробное вычитание делителя из старших n разрядов делимого. Если получающийся при этом остаток отрицательный, деление возможно, если же остаток от пробного вычитания положительный, то результат по абсолютной величине переполняет разрядную сетку. В этом случае должен формироваться признак переполнения и деление прекращается.

Помимо этого требования перед началом операции необходимо исключить возможность ситуации деления на 0.

Реализовать деление можно двумя основными способами:

– с неподвижным делимым и сдвигаемым вправо делителем;

– с неподвижным делителем и сдвигаемым влево делимым.

Недостатком первого способа является то, что в устройстве деления сумматор должен быть двойной длины. Второй способ позволяет строить делитель с сумматором одинарной длины. Поэтому на практике для реализации операции деления используется второй способ с *неподвижным делителем*.

В ЭВС для деления двоичных чисел можно использовать два метода: деление с восстановлением остатка и деление без восстановления остатка.

1. *Деление с восстановлением остатка* [3].

Этот алгоритм соответствует общепринятому способу деления чисел, поэтому он является наиболее очевидным алгоритмом и носит название алгоритма *деления с неподвижным делителем и восстановлением остатка*. Данный алгоритм может быть описан следующим образом:

1) исходное значение частичного остатка полагается равным старшим разрядам делимого;

2) частичный остаток сдвигается на один разряд влево;

3) из сдвинутого ЧО вычитается делитель и анализируется знак результата вычитания;

4) очередной разряд частного равен единице, когда результат вычитания положителен, и нулю, если отрицателен. В последнем случае значение остатка восстанавливается до того значения, которое было до вычитания;

5) пункты 2–4 последовательно выполняются для получения всех разрядов частного.

В общем случае для определения (*n*-*i*)-го разряда частного анализируется частичный остаток. Если $ЧО_{i-1} \geq 0$, он сдвигается влево, и из него далее вычитается Дт, в результате чего получается остаток $ЧО_i$. Если же $ЧО_{i-1} < 0$, то сначала восстанавливается предыдущий положительный (сдвинутый) остаток, для чего к $ЧО_{i-1}$ прибавляется Дт. Далее восстановленный остаток сдвигается влево, из него вычитается Дт и в результате получается остаток $ЧО_i$. При $ЧО_{i-1} \geq 0$ (*n*-*i*)-й разряд частного равен единице, при $ЧО_{i-1} < 0$ – нулю.

Пример 29. Разделить с восстановлением остатка делимое $A = 41$ на делитель $B = 7$ ($n = 4$).

Решение. $A = 00101001$; $B = 0111$; $[-B]_{доп} = 1.1001$.

Дм		0010	1001	0111 Дт
$ЧО_0$		0010	1001	0101 Частное 5
Сдвиг $ЧО_0$ влево	0	0101	0010	
-Дт	1	1001		
$ЧО_1 < 0$	1	1110	0010	
+Дт (восстановление $ЧО_0$)		0111		
Восстановленный $ЧО_0$	0	0101	0010	
Сдвиг $ЧО_0$ влево	0	1010	0100	
-Дт	1	1001		
$ЧО_2 > 0$	0	0011	0100	
Сдвиг $ЧО_2$ влево	0	0110	1000	
-Дт	1	1001		
$ЧО_3 < 0$	1	1111	1000	
+Дт (восстановление $ЧО_2$)		0111		
Восстановленный $ЧО_2$	0	0110	1000	
Сдвиг $ЧО_2$ влево	0	1101	0000	
-Дт	1	1001		
$ЧО_4 > 0$	0	0110	0000	
		0110		Остаток 6

2. *Деление без восстановления остатка* [3].

Недостаток алгоритма с восстановлением остатка заключается в необходимости выполнения на отдельных шагах дополнительных операций сложения для восстановления частичного остатка. Это увеличивает время выполнения деления, причем это время может меняться в зависимости от конкретного сочетания кодов операндов. Поэтому реальные делители строятся на основе алго-

ритма *деления с неподвижным делителем без восстановления остатка*. Данный алгоритм может быть описан следующим образом:

1) исходное значение частичного остатка полагается равным старшим разрядам делимого;

2) частичный остаток сдвигается на один разряд влево;

3) из сдвинутого частичного остатка вычитается делитель, если остаток положительный, и к сдвинутому частичному остатку прибавляется делитель, если остаток отрицательный;

4) очередной разряд частного равен единице, когда результат вычитания положительный, и нулю, если он отрицательный;

5) пункты 2–4 последовательно выполняются для получения всех разрядов частного.

Пример 30. Разделить без восстановления остатка делимое $A = 41$ на делитель $B = 7$ ($n = 4$).

Решение. $A = 00101001$; $B = 0111$; $[-B]_{\text{доп}} = 1.1001$.

Дм		0010	1001	0111 Дт
$ЧО_0$		0010	1001	0101 Частное 5
Сдвиг $ЧО_0$ влево	0	0101	0010	
–Дт	1	1001		
$ЧО_1 < 0$	1	1110	0010	
Сдвиг $ЧО_1$ влево	1	1100	0100	
+Дт		0111		
$ЧО_2 > 0$	0	0011	0100	
Сдвиг $ЧО_2$ влево	0	0110	1000	
–Дт	1	1001		
$ЧО_3 < 0$	1	1111	1000	
Сдвиг $ЧО_3$ влево	1	1111	0000	
+Дт		0111		
$ЧО_4 > 0$	0	0110	0000	
		0110		Остаток 6

2.4.2 Деление чисел со знаком

Деление чисел со знаком, как и умножение, может быть выполнено путем перехода к абсолютным значениям делимого и делителя с последующим присвоением частному знака плюс при совпадающих знаках делимого и делителя, либо минус – в противном случае. Остаток всегда имеет знак делимого.

Деление чисел, представленных в дополнительном коде, можно осуществлять, не переходя к модулям. При этом алгоритм деления видоизменяется в соответствии с особенностями использования дополнительного кода. Рассмотрим эти изменения для метода деления без восстановления остатка.

Так как делимое и делитель необязательно имеют одинаковые знаки, то действия с частичным остатком (прибавление или вычитание Дт) зависят от знаков остатка и делителя и определяются содержимым таблицы 7. Если знак остатка совпадает со знаком делителя, то очередной разряд частного равен 1, в противном случае – 0.

После завершения алгоритма может потребоваться коррекция частного, которая заключается в увеличении частного на единицу. Такая коррекция должна производиться в следующих случаях:

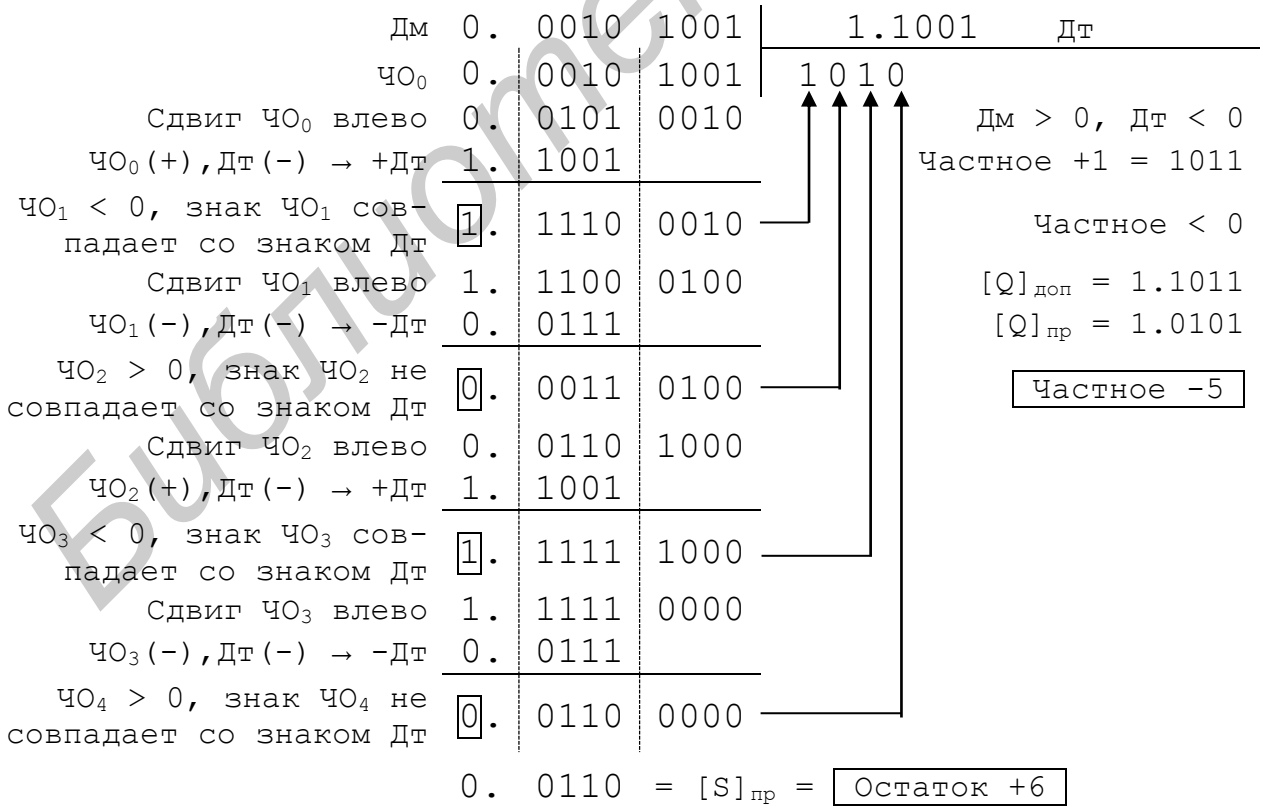
1) Дм > 0 и Дт < 0;
2) Дм < 0 и Дт > 0 при ненулевом остатке от деления;
3) Дм< 0 и Дт < 0 при нулевом остатке от деления.

Таблица 7

Знак остатка	Знак делителя	Действие
+	+	Вычитание делителя
+	–	Прибавление делителя
–	+	Прибавление делителя
–	–	Вычитание делителя

Пример 31. Разделить без восстановления остатка делимое $A = 41$ на делитель $B = –7$ ($n = 4$).

Решение. $[A]_{\text{пр}} = [A]_{\text{доп}} = 0.00101001$;
$[B]_{\text{доп}} = 1.1001$; $[-B]_{\text{доп}} = 0.0111$.



Пример 32. Разделить без восстановления остатка делимое $A = –41$ на делитель $B = +7$ ($n = 4$).

Решение. $[A]_{доп} = 1.11010111$; $[B]_{доп} = 0.0111$; $[-B]_{доп} = 1.1001$.

Дм	1.	1101	0111	0.0111	Дт
ЧО$_0$	1.	1101	0111	1010	
Сдвиг ЧО$_0$ влево	1.	1010	1110		Дм < 0, Дт > 0
ЧО$_0$(–),Дт(+) → +Дт	0.	0111			Остаток ≠ 0
ЧО$_1$ > 0, знак ЧО$_1$ совпадает со знаком Дт	[0].	0001	1110		Частное +1 = 1011
Сдвиг ЧО$_1$ влево	0.	0011	1100		Частное < 0
ЧО$_1$(+),Дт(+) → –Дт	1.	1001			$[Q]_{доп}$ = 1.1011
ЧО$_2$ < 0, знак ЧО$_2$ не совпадает со знаком Дт	[1].	1100	1100		$[Q]_{пр}$ = 1.0101
Сдвиг ЧО$_2$ влево	1.	1001	1000		Частное –5
ЧО$_2$(–),Дт(+) → +Дт	0.	0111			
ЧО$_3$ > 0, знак ЧО$_3$ совпадает со знаком Дт	[0].	0000	1000		
Сдвиг ЧО$_3$ влево	0.	0001	0000		
ЧО$_3$(+),Дт(+) → –Дт	1.	1001			
ЧО$_4$ < 0, знак ЧО$_4$ не совпадает со знаком Дт	[1].	1010	0000		
	1.	1010	= $[S]_{доп}$		
	1.	0110	= $[S]_{пр}$ =	Остаток –6	

Пример 33. Разделить без восстановления остатка делимое $A = –41$ на делитель $B = –7$ ($n = 4$).

Решение. $[A]_{доп} = 1.11010111$; $[B]_{доп} = 1.1001$; $[-B]_{доп} = 0.0111$.

Дм	1.	1101	0111	1.1001	Дт
ЧО$_0$	1.	1101	0111	0101	
Сдвиг ЧО$_0$ влево	1.	1010	1110		Дм < 0, Дт < 0
ЧО$_0$(–),Дт(–) → –Дт	0.	0111			Остаток ≠ 0
ЧО$_1$ > 0, знак ЧО$_1$ не совпадает со знаком Дт	[0].	0001	1110		Частное = 0101
Сдвиг ЧО$_1$ влево	0.	0011	1100		Частное > 0
ЧО$_1$(+),Дт(–) → +Дт	1.	1001			$[Q]_{пр}$ = 0.0101
ЧО$_2$ < 0, знак ЧО$_2$ совпадает со знаком Дт	[1].	1100	1100		Частное +5
Сдвиг ЧО$_2$ влево	1.	1001	1000		
ЧО$_2$(–),Дт(–) → –Дт	0.	0111			
ЧО$_3$ > 0, знак ЧО$_3$ не совпадает со знаком Дт	[0].	0000	1000		
Сдвиг ЧО$_3$ влево	0.	0001	0000		
ЧО$_3$(+),Дт(–) → +Дт	1.	1001			
ЧО$_4$ < 0, знак ЧО$_4$ не совпадает со знаком Дт	[1].	1010	0000		
	1.	1010	= $[S]_{доп}$		
	1.	0110	= $[S]_{пр}$ =	Остаток –6	

2.4.3 Ускорение целочисленного деления

Операция деления предоставляет относительно небольшое количество путей для своего ускорения. Возможности по ускорению целочисленного деления можно свести к следующим [3]:

1) замена делителя обратной величиной с последующим ее умножением на делимое;

2) сокращение времени вычисления частичных остатков в традиционных методах деления (с восстановлением или без восстановления остатка) за счет ускорения операций суммирования (вычитания);

3) сокращение времени вычисления за счет уменьшения количества операций суммирования (вычитания) при расчете Ч0;

4) вычисление частного в избыточной системе счисления.

За исключением первого из перечисленных подходов все прочие фактически являются модификациями традиционного способа деления.

3 ДВОИЧНАЯ АРИФМЕТИКА С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

3.1 Сложение и вычитание чисел с плавающей запятой

Операции сложения и вычитания выполняются одинаково, но в случае вычитания необходимо изменить знак второго операнда на противоположный. Таким образом, операция вычитания чисел A и B с плавающей запятой начинается с изменения знака вычитаемого B на противоположный, после чего выполняется последовательность действий сложения чисел A и $(-B)$.

Особенностью арифметики с плавающей запятой является то, что операции над двумя составляющими чисел с ПЗ – мантиссами и порядками операндов – выполняются раздельно.

При сложении двух чисел разряды мантисс слагаемых должны иметь одинаковый вес. Это требование, в свою очередь, выполняется тогда, когда слагаемые имеют одинаковые порядки. Следовательно, *чтобы сложить два числа с плавающей запятой, нужно прежде всего произвести выравнивание их порядков*.

Алгоритм сложения двух чисел с произвольными знаками сводится к выполнению следующей последовательности действий [3]:

1) производится сравнение порядков p_A и p_B слагаемых A и B путем вычитания порядка p_B второго слагаемого из порядка p_A первого слагаемого или наоборот. Для вычитания порядков знак p_B меняется на противоположный и далее порядки складываются по правилу сложения целых чисел, при этом отрицательный порядок вступает в операцию сложения в дополнительном или обратном коде. Разность $p = p_A - p_B$ указывает, на сколько разрядов надо сдвинуть вправо мантиссу числа с меньшим порядком. Если $p = p_A - p_B > 0$, то $p_A > p_B$ и для выравнивания порядков необходимо сдвинуть вправо мантиссу M_B второго слагаемого. Если же $p = p_A - p_B < 0$, то $p_A < p_B$ и для выравнивания порядков

необходимо сдвинуть вправо мантиссу M_A первого слагаемого. В случае $p = p_A - p_B = 0$ слагаемые имеют равные порядки;

2) мантисса числа с меньшим порядком сдвигается вправо на p разрядов. Если в процессе сдвига мантисса обращается в 0, в качестве результата операции берется другой операнд;

3) выполняется сложение мантисс чисел с равными порядками по правилу сложения правильных дробей, при этом отрицательная мантисса вступает в операцию сложения в дополнительном или обратном коде. В результате выполнения операции сложения может получиться нулевое значение мантиссы. В этом случае имеет место ситуация *потери значимости мантиссы*, и результат операции принимается равным нулю;

4) если при сложении мантисс произошло переполнение (в знаковом разряде оказалась старшая цифра суммы, т. е. произошло нарушение нормализации влево), производится нормализация путем сдвига мантиссы суммы вместе со знаковым разрядом вправо на один разряд с увеличением порядка на единицу. При этом в знаковый разряд мантиссы суммы записывается знак результата, т. е. выполняется арифметический сдвиг. В процессе нормализации возможно *переполнение порядка*. В этом случае операция прекращается и формируется признак переполнения. Если же после сложения мантисс мантисса суммы стала меньше 0,5 (в абсолютном значении мантиссы суммы после запятой оказались нули, т. е. произошло нарушение нормализации вправо), мантисса суммы сдвигается влево на количество разрядов, занятых нулями. Одновременно из порядка p результата вычитается число, равное этому количеству нулей. При нормализации результата возможно *исчезновение порядка* (*потеря значимости порядка*).

Пример 34. Сложить два числа в форме с плавающей запятой: $A = 45{,}25$; $B = -6{,}75$ ($m = 8$, $q = 4$).

Решение

Подготовительные операции:

1) определяем общую разрядность разрядной сетки для представления чисел с плавающей запятой n:

$$n = m + q + 2 = 8 + 4 + 2 = 14,$$

где m – разрядность значащей части мантиссы;

q – разрядность значащей части порядка, два разряда для представления знаков мантиссы и порядка;

2) записываем числа в двоичной системе счисления:

$$A = 101101{,}01,$$

$$B = -110{,}11;$$

3) представляем числа в виде мантиссы и порядка:

$$A = 0{,}10110101 \cdot 2^6 = 0{,}10110101 \cdot 2^{110},$$

$B = -0{,}11011 \cdot 2^3 = -0{,}11011 \cdot 2^{11}$;

4) записываем числа в заданной разрядной сетке (машинные изображения чисел):

$[M_A]_{\text{пр}} = [M_A]_{\text{доп}} = 0{,}10110101$, $[p_A]_{\text{доп}} = [p_A]_{\text{пр}} = 0.0110$,
$[M_B]_{\text{пр}} = 1{,}11011000$,
$[M_B]_{\text{доп}} = 1{,}00101000$, $[p_B]_{\text{доп}} = [p_B]_{\text{пр}} = 0.0011$.

Выполнение операции:

5) сравниваем порядки. Находим разность порядков $p = p_A - p_B$:

$[-p_B]_{\text{доп}} = 1.1101$,

```
  0.0110    [pA]доп
+ 1.1101    [pB]доп
  0.0011    [p]доп = [p]пр = +3 > 0
```

$p_A > p_B$;

6) мантиссу числа B сдвигаем вправо на p разрядов (сдвиг арифметический)

$\left[\vec{M}_B\right]_{\text{доп}} = 1{,}11100101$;

7) складываем мантиссы

```
  0,10110101    [MA]доп
+ 1,11100101    [MB]доп
 [1]0,10011010  [MA+B]доп = [MA+B]пр
   отбрасывается
```

Нарушения нормализации нет;

8) записываем результат:

$[M_{A+B}]_{\text{доп}} = [M_{A+B}]_{\text{пр}} = 0{,}10011010$,

$[p_{A+B}]_{\text{доп}} = [p_A]_{\text{доп}} = [p_{A+B}]_{\text{пр}} = 0.0110$;

9) проверяем полученный результат:

$A + B = +0{,}1001101 \cdot 2^{110} = +0{,}1001101 \cdot 2^6 = +100110{,}1 = +38{,}5$.

Пример 35. Сложить два числа в форме с плавающей запятой: A = -37,25; B = 6,75 (m = 8, q = 4).

Решение

Подготовительные операции:

1) определяем общую разрядность разрядной сетки для представления чисел с плавающей запятой n:

$$n = m + q + 2 = 8 + 4 + 2 = 14,$$

где m – разрядность значащей части мантиссы;

q – разрядность значащей части порядка, два разряда для представления знаков мантиссы и порядка;

2) записываем числа в двоичной системе счисления:

$A = -100101{,}01$,
$B = 110{,}11$;

3) представляем числа в виде мантиссы и порядка:

$A = -0{,}10010101 \cdot 2^{6} = -0{,}10010101 \cdot 2^{110}$,
$B = 0{,}11011 \cdot 2^{3} = 0{,}11011 \cdot 2^{11}$;

4) записываем числа в заданной разрядной сетке (машинные изображения чисел):

$[\mathrm{M}_A]_{\text{пр}} = 1{,}10010101$,

$[M_A]_{\text{доп}} = 1{,}01101011$, $[p_A]_{\text{доп}} = [p_A]_{\text{пр}} = 0.0110$,

$[M_B]_{\text{пр}} = [M_B]_{\text{доп}} = 0{,}11011000$, $[p_B]_{\text{доп}} = [p_B]_{\text{пр}} = 0.0011$.

Выполнение операции:

5) сравниваем порядки. Находим разность порядков $p = p_A - p_B$

$[-p_B]_{\text{доп}} = 1.1101$,

```
  0.0110    [pA]доп
+ 1.1101    [pB]доп
  0.0011    [p]доп = [p]пр = +3 > 0
```

$p_A > p_B$;

6) мантиссу числа B сдвигаем вправо на p разрядов (сдвиг арифметический)

$\left[\vec{M}_B\right]_{\text{доп}} = 0{,}00011011$;

7) складываем мантиссы

```
  1,01101011    [MA]доп
+ 0,00011011    [MB]доп
  1,10000110    [MA+B]доп
```

Нарушение нормализации вправо

$[M_{A+B}]_{\text{пр}} = 1{,}01111010 < 0{,}5$;

8) нормализация: сдвигаем мантиссу результата влево на один разряд и уменьшаем порядок на единицу;

9) записываем результат:

$[M_{A+B}]_{\text{пр}} = 1{,}1111010$,

$[p_{A+B}]_{\text{пр}} = 0.0101$;

10) проверяем полученный результат:

$A + B = -0{,}111101 \cdot 2^{101} = -0{,}111101 \cdot 2^5 = -11110{,}1 = -30{,}5$.

3.2 Умножение чисел с плавающей запятой

При умножении чисел *A* и *B* с плавающей запятой [3] мантисса произведения M_P определяется путем перемножения мантисс сомножителей $M_P = M_A M_B$. Порядок произведения p_P определяется путем сложения порядков сомножителей $p_P = p_A + p_B$. Если используется смещенный порядок, то в полученной сумме будет содержаться удвоенное смещение, поэтому из нее необходимо вычесть величину смещения. Результатом действий с порядками может стать как *переполнение порядка*, так и *потеря значимости*.

Перемножение мантисс сомножителей выполняется по правилу умножения правильных дробей. Знак произведения определяется путем суммирования по модулю два знаковых разрядов мантисс.

Поскольку мантиссы сомножителей нормализованы, абсолютное значение мантиссы M_P заключено в следующих пределах: $0{,}01_{(2)} \le |M_P| < 1$, т. е. может иметь место нарушение нормализации вправо не более чем на один разряд. После перемножения мантисс сомножителей результат при необходимости нормализуется путем сдвига значащих разрядов мантиссы M_P на один разряд влево и вычитания единицы из порядка p_P. При нормализации результата возможно *исчезновение порядка* (*потеря значимости порядка*).

3.3 Деление чисел с плавающей запятой

При делении чисел *A* и *B* с плавающей запятой [3] мантисса частного M_Q определяется путем деления мантиссы делимого на мантиссу делителя $M_Q = M_A/M_B$. Порядок частного определяется путем вычитания порядков операндов $p_Q = p_A - p_B$. Если используется смещенный порядок, то вычитание порядков приводит к удалению смещения из порядка результата. Следовательно, для получения смещенного порядка результата к разности должно быть добавлено смещение. После выполнения этих действий необходима проверка на *переполнение порядков* и *потерю значимости*.

Деление мантиссы делимого на мантиссу делителя выполняется по правилу деления правильных дробей. При этом положительный остаток в результате пробного вычитания модуля мантиссы делителя из модуля мантиссы делимого дает единицу в разряде целой части, которая не вызывает формирования признака переполнения разрядной сетки. Знак частного определяется путем суммирования по модулю два знаковых разрядов мантисс операндов.

Поскольку мантиссы делимого и делителя нормализованы, абсолютное значение мантиссы частного M_Q заключено в следующих пределах: $0{,}1_{(2)} < |M_Q| < 10_{(2)}$, т. е. может иметь место нарушение нормализации влево не

более чем на один разряд. После деления мантисс операндов результат при необходимости нормализуется путем сдвига мантиссы вместе со знаковым разрядом вправо на один разряд с увеличением порядка p_Q на единицу. В процессе нормализации возможно *переполнение порядка*.

4 ДЕСЯТИЧНАЯ АРИФМЕТИКА

4.1 Выполнение операции сложения в двоично-десятичном коде

Для определения формальных правил поразрядного сложения чисел, представленных в двоично-десятичном коде [1], рассмотрим особенности, которые присущи этому коду:

1) наличие разрешенных и запрещенных комбинаций. Появление запрещенной комбинации при выполнении каких-то действий над числами свидетельствует о возникновении ошибки или же о необходимости ввести корректировку результата;

2) при сложении тетрад возникает потетрадный перенос $\pi_i^* = 16$ вместо десятичного поразрядного переноса $\pi_i^* = 10$. Это приводит к необходимости коррекции результата.

При сложении двух чисел в двоично-десятичном коде $A_д = a_{n-1} \dots a_1 a_0$ и $B_д = b_{n-1} \dots b_1 b_0$ (a_i и b_i – тетрады соответствующих чисел) могут возникнуть следующие случаи:

1) $a_i + b_i + c_{i-1} < 10$, где c_{i-1} – перенос из предыдущей тетрады. При сложении в данной позиции десятичного числа образуется сумма меньше 10. Если действия над разрядами тетрады производят по правилам двоичной арифметики, то правильный результат получается без коррекции;

2) $a_i + b_i + c_{i-1} \geq 10$. При сложении возникает десятичный перенос в следующую позицию, и сумма должна быть равна $a_i + b_i + c_{i-1} - 10$. Свидетельством того, что результат неправильный, является либо появление запрещенной комбинации, если $10 \leq a_i + b_i + c_{i-1} \leq 15$, либо появление потетрадного переноса $\pi_i^* = 16$, что превышает значение десятичного переноса на 6, если $16 \leq a_i + b_i + c_{i-1} \leq 19$. Следовательно, требуется коррекция результата в данной тетраде введением поправки, равной +6.

Пример 36. Сложить две тетрады: a_i = 0011, b_i = 0101, c_{i-1} = 1.

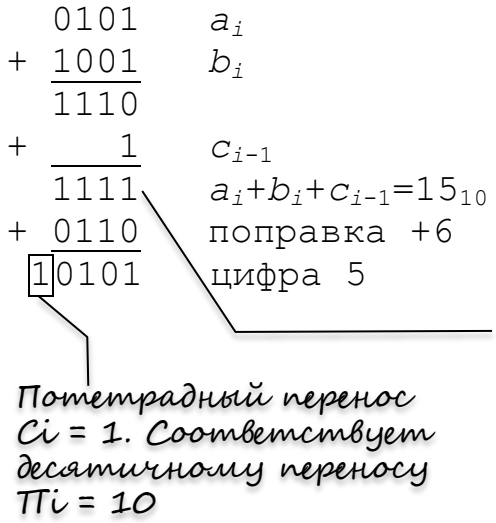
Решение

```
  0011    a_i
+ 0101    b_i
  1000
+    1    c_i-1
  1001    a_i+b_i+c_i-1=9
```

Сумма в данной тетраде меньше 10

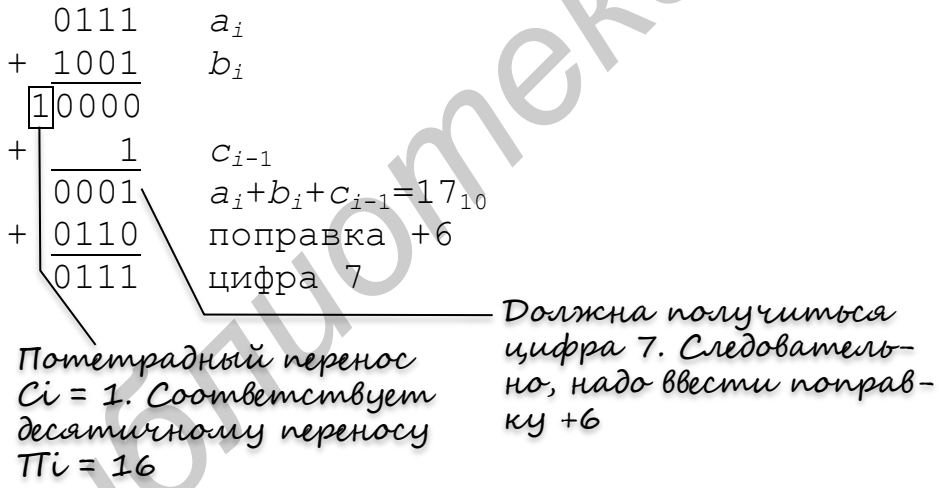
Ответ. $a_i + b_i + c_{i-1} = 1001$.

Пример 37. Сложить две тетрады: $a_i = 0101$, $b_i = 1001$, $c_{i-1} = 1$.
Решение



Ответ. $a_i + b_i + c_{i-1} = 10101$.

Пример 38. Сложить две тетрады: $a_i = 0111$, $b_i = 1001$, $c_{i-1} = 1$.
Решение



Ответ. $a_i + b_i + c_{i-1} = 10111$.

Таким образом, корректирующее слагаемое $6_{(10)} = 0110_{(2)}$ при сложении двух чисел в двоично-десятичном коде должно добавляться к каждой тетраде, в которой в процессе сложения:

1) получена недопустимая цифра (запрещенная комбинация);
2) возник перенос в следующую тетраду.

Пример 39. Сложить два числа, представленные в двоично-десятичном коде: $A = 279$; $B = 591$.

Решение

1) представляем числа в двоично-десятичном коде

$A = 0010\ 0111\ 1001$,
$B = 0101\ 1001\ 0001$;

2) выполняем сложение по правилам двоичной арифметики, фиксируя запрещенные комбинации и переносы между тетрадами

```
  0010 0111 1001   A
+ 0101 1001 0001   B
  1000 0000 1010
```

Перенос из тетрады

Запрещенная комбинация

3) выполняем коррекцию результата

```
  1000 0000 1010
+      0110 0110   коррекция +6
  1000 0111 0000
   8    7    0
```

4) проверяем полученный результат

$A + B = 279 + 591 = 870$.

4.2 Выполнение операции вычитания в двоично-десятичном коде

При выполнении вычитания в двоично-десятичном коде [1] возможен заем между тетрадами. При двоичном вычитании заем эквивалентен $16_{(10)}$, в то время как для десятичного вычитания он должен соответствовать $10_{(10)}$. Чтобы компенсировать излишек в заеме, необходимо вычесть корректирующее число $6_{(10)} = 0110_{(2)}$ из той группы, которая получила заем.

Таким образом, при вычитании чисел в двоично-десятичном коде возникает лишь одна ситуация, требующая коррекции, а именно ситуация заема между тетрадами. Коррекция при этом сводится к вычитанию $6_{(10)} = 0110_{(2)}$ из каждой тетрады разности, которая получила заем.

Пример 40. Вычесть два числа, представленные в двоично-десятичном коде: $A = 617$; $B = 385$.

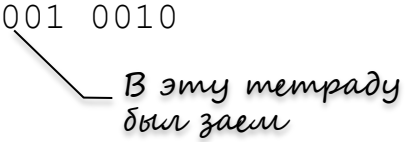
Решение

1) представляем числа в двоично-десятичном коде

$A = 0110\ 0001\ 0111$,
$B = 0011\ 1000\ 0101$;

2) выполняем вычитание по правилам двоичной арифметики, фиксируя заемы между тетрадами

```
  0110 0001 0111   A
- 0011 1000 0101   B
  0010 1001 0010
```



3) выполняем коррекцию результата

```
  0010 1001 0010
-      0110          коррекция -6
  0010 0011 0010
   2    3    2
```

4) проверяем полученный результат

$A - B = 617 - 385 = 232$.

4.3 Сложение в двоично-десятичном коде чисел со знаком

Как и в двоичной системе, для представления десятичных чисел со знаками можно использовать прямой, обратный или дополнительный код [4]. По-прежнему двоичные цифры 0 и 1 могут служить указателями знака.

Прямой код можно получить простым добавлением знакового бита к представлению абсолютной величины.

Для обратного и дополнительного кодов, если $A_д \geq 0$, знаковый разряд должен содержать 0, а остальные *n* десятичных цифр (значащая часть) – абсолютное значение $A_д$.

Для обратного кода, если $A_д \leq 0$, в старшем, знаковом разряде должна быть 1, а следующие *n* десятичных цифр должны соответствовать числу

$$10^n - 1 - |A_д|,$$

которое называют *обращением* $A_д$.

Для дополнительного кода, если $A_д < 0$, то в знаковом разряде должна быть 1, а следующие *n* десятичных цифр должны соответствовать числу

$$10^n - |A_д|,$$

которое называют *дополнением* $A_д$.

Обращение сводится к вычитанию каждой цифры из 9 и инверсии знака. Дополнение получается путем добавления 1 к результату обращения, что эквивалентно тому, что младшая десятичная ненулевая цифра вычитается из 10.

Обращение десятичного числа можно получить и другим способом. Инвертирование тетрады означает получение дополнения до $2^4 - 1 = 15$, вместо дополнения до 9. Следовательно, необходимо убрать разницу. Один из используемых при этом приемов состоит в том, что во все тетрады значащей части

числа в двоично-десятичном коде добавляется $6_{(10)} = 0110_{(2)}$ и после этого производится инвертирование разрядов тетрад. Полученное изображение представляет собой обращение числа.

Как и в двоичной системе, обращение или взятие дополнения кода соответствует изменению знака числа, представляемого этим кодом.

Сложение в двоично-десятичном коде чисел со знаком осуществляется следующим образом. Значащие части кодов складываются по рассмотренным правилам десятичной арифметики, а знаковые разряды – по правилам двоичной арифметики. В дополнительном коде перенос из знакового разряда игнорируется, а в обратном выполняется круговой перенос.

Пример 41. Выполнить десятичное сложение в дополнительном коде: $A = +73$; $B = -42$.

Решение

1) представляем числа в двоично-десятичном коде

$A = 0111\ 0011$,
$B = -0100\ 0010$;

2) получаем дополнительные коды чисел

$[A]_{доп} = 0.0111\ 0011$,

```
  0111 0011   B
+ 0110 0110   +6 в обе тетрады
  1010 1000
  0101 0111   инверсия разрядов – получаем обращение
  0000 0001   +1
  0101 1000   получаем дополнение
```

$[B]_{доп} = 1.0101\ 1000$;

3) выполняем сложение по рассмотренным выше правилам

```
  0.0111 0011   [A]доп
+ 1.0101 1000   [B]доп
  1.1100 1011
```

Запрещенная комбинация *Запрещенная комбинация*

4) выполняем коррекцию результата

```
   1.1100 1011
 +   0110 0110   коррекция +6
  10.0011 0001   [A+B]доп = [A+B]пр
   0. 3    1     +31
```

Отбрасывается

5) проверяем полученный результат

$A + B = (+73) + (-42) = +31$.

ЛИТЕРАТУРА

1 Савельев, А. Я. Основы информатики : учебник для вузов / А. Я. Савельев. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2001. – 328 с.

2 Точи, Р. Дж. Цифровые системы. Теория и практика / Р. Дж. Точи, Н. С. Уидмер. – 8-е изд. – М. : Издательский дом «Вильямс», 2004. – 384 с.

3 Орлов, С. А. Организация ЭВМ и систем: учебник для вузов / С. А. Орлов, Б. Я. Цилькер. – 2-е изд. – СПб. : Питер, 2011. – 688 с.

4 Гивоне, Д. Микропроцессоры и микрокомпьютеры. Вводный курс / Д. Гивоне, Р. Россер ; пер. с англ. – М. : Мир, 1983. – 464 с.

5 Путков, В. Н. Электронные вычислительные устройства : учеб. пособие / В. Н. Путков, И. И. Обросов, С. В. Бекетов. – Минск : Выш. шк., 1981. – 333 с.

6 Гуртовцев, А. Л. Программы для микропроцессоров : справ. пособие / А. Л. Гуртовцев, С. В. Гудыменко. – Минск : Выш. шк., 1989. – 352 с.

Св. план 2013, поз. 10

Учебное издание

Качинский Михаил Вячеславович
Клюс Владимир Борисович
Давыдов Александр Борисович

АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *И. В. Ничипор*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лысеня*

Подписано в печать 8.04.2014. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 3,84. Уч-изд. л. 3,7. Тираж 100 экз. Заказ 240.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014, №2/113 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
2200013, Минск, П. Бровки, 6