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

Кафедра программного обеспечения информационных технологий

**А.Т. Пешков**

# Организация и функционирование ЭВМ

Методическое пособие для студентов специальности
«Программное обеспечение информационных технологий»
дневной формы обучения
В 3 частях
Часть 1
Арифметические основы ЭВМ



Минск 2004

УДК 004.3(075.8)
ББК 32.973 я73
П 31

Р е ц е н з е н т:
доцент кафедры ИИТ БГУИР, канд. техн. наук В. Н. Мухаметов

**Пешков А.Т.**

П 31 Организация и функционирование ЭВМ. Метод. пособие для студ. спец. «Программное обеспечение информационных технологий» дневной формы обуч.: В 3 ч. Ч. 1. Арифметические основы ЭВМ / А.Т. Пешков. - Мн.: БГУИР, 2004. — 61 с.: ил.
ISBN 985-444-602-6 (ч.1)

Пособие предназначено для студентов специальности «Программное обеспечение информационных технологий». В нем излагается материал, связанный с арифметическими, логическими и схемотехническими вопросами построения ЭВМ. Пособие включает большое количество иллюстраций, таблиц и примеров с решениями, что способствует успешному усвоению излагаемого материала.

**УДК 004.3 (075.8)**
**ББК 32.973 я73**



**ISBN 985-444-602-6 (ч.1)**
**ISBN 985-444-601-8**

**Содержание**

# ВВЕДЕНИЕ

Начало развития машин для выполнения вычислений можно отнести к семнадцатому столетию, когда в 1642 г. выдающийся французский математик Блез Паскаль[1] изобрел и сконструировал первую суммирующую машину. Однако из-за несовершенства развития механики того времени созданная им машина не нашла практического применения. В конце этого же столетия в 1694 г. немецкий математик Гофрид Лейбниц построил вычислительную машину, которая могла выполнять не только операции сложения, но и операции умножения. В 1874 г. российский инженер В.Т. Однер сконструировал арифмометр, которой был лучшим для того времени и долгое время оставался прототипом разрабатывавшихся впоследствии машин подобного назначения.

Идея создания машины с элементами программирования, т.е. с автоматической организацией вычислительного процесса, принадлежит Чарльзу Боббиджу, предложившему в 1833 г. проект «аналитической машины», которая помимо использования принципа программного управления имела память для хранения программы, исходных данных и результатов их обработки. В предложенной машине были предусмотрены средства ввода исходной информации с перфоленты и средства вывода информации в печатной форме. Эта машина была даже построена, однако реализация достаточно сложных идей, положенных в основу этой «аналитической машины», на базе сравнительно несовершенной механики того времени привела к тому, что созданная машина оказалась настолько ненадежной, что не нашла фактически никакого практического использования.

Все разрабатываемые машины для выполнения расчетов до 40-х гг. строились на основе механических узлов. Появление первых электронных вычислительных машин можно отнести к сороковым годам. Одной из первых ЭВМ является «ENIAC», созданная в США Джоном Мокли и Дж. Преспером Эккертом. При построении машины было использовано около 20 тысяч электронных ламп. Она имела колоссальные габариты (для ее размещения было построено специальное помещение площадью более 100 метров) и имела фантастическое по тем временам быстродействие - 5 тысяч сложений в секунду.

С 40-х гг. развитие вычислительной техники с точки зрения используемой технологической базы можно подразделить на этапы создания ЭВМ следующих поколений:

- ЭВМ на электронных вакуумных приборах;
- ЭВМ на полупроводниковых приборах;
- ЭВМ на интегральных схемах;
- ЭВМ на больших интегральных схемах;
- ЭВМ на сверхбольших интегральных схемах.

[1] Именем этого математика был назван один из самых распространенных современных алгоритмических языков программирования.

При переходе от поколения к следующему поколению почти на порядок улучшались основные параметры ЭВМ, к числу которых относятся:

- быстродействие;
- емкость памяти;
- потребляемая мощность;
- габариты.

За это время существенно изменилась структурная организация ЭВМ, её внешняя память, средства ввода - вывода информации.

Современные ЭВМ при габаритах, как правило, соответствующих настольному варианту, обладают быстродействием, измеряемым десятками миллионов операций в секунду, имеют только оперативную память емкостью в десятки (а в некоторых случаях сотни) миллионов байт, оснащены разнообразными средствами ввода -вывода, позволяющими вести обмен информации с пользователем в удобной для последнего форме.

Структура современной ЭВМ включает следующие основные компоненты:

– ОП - оперативная память, используемая для хранения исполняемых в данное время программ, исходных данных, промежуточных и окончательных результатов;

– процессор - устройство, осуществляющее основную обработку информации в соответствии с исполняемой программой;

– ПУ - периферийные устройства, включающие средства ввода - вывода информации, представленной в различной форме, осуществляющие двусторонний обмен данными с пользователем, а также устройства типа внешней памяти, имеющие огромную информационную ёмкость, позволяющую хранить все исходные данные, программы, промежуточные и конечные результаты обработки информации.

В данном пособии рассматриваются основополагающие материалы, связанные с ЭВМ, а именно:

- арифметические основы ЭВМ;
- логические основы ЭВМ;
- схемотехнические основы ЭВМ.

Приведенные материала основываются на разделах «Физика», «Математика» и «Электротехника».

Полученные при их изучении знания могут быть использованы при изучении дисциплин по программированию («Конструирование программ и языки программирования», «Системное программное обеспечение» и др.) и всех дисциплин, связанных с аппаратной частью ЭВМ.

## 1. Системы счисления

Арифметическая обработка чисел во многом определяется системами счисления, представляющими собой совокупность используемых цифр и набором правил, позволяющих однозначно представлять числовую информацию.

В своей повседневной деятельности человек использует различные системы счисления, к числу которых относятся десятичная система счисления, римская система, система исчисления времени и т.д. Все системы счисления можно подразделить на позиционные и непозиционные.

В не позиционных системах счисления «доля» цифры или её вес в количественном измерении записанного числа не зависит от местоположения данной цифры в записи этого числа. Типичным примером такой системы счисления является римская система счисления. В этой системе используются цифры:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| I | V | X | L | C | D | M | и т.д. - римские цифры; |
| 1 | 5 | 10 | 50 | 100 | 500 | 1000 | - десятичные эквиваленты римским цифрам. |

При количественной оценки числа его значение определяется как сумма значений цифр, составляющих запись числа, кроме пар, состоящих из цифры меньшего веса, предшествующей цифре большего веса, значение которой определяется как разность веса большей и меньшей цифр. Например, значение числа

MMMCMLIX

определяется как сумма

1000 + 1000 + 1000 + (1000-100) + 50 + (10 - 1), что соответствует десятичному эквиваленту 3959.

Позиционная система счисления характеризуется тем, что «доля» некоторой цифры в количественной оценки записанного числа определяется не только видом цифры, но и местоположением (позицией) данной цифры в записи числа, т.е. каждая позиция (разряд) в записи числа имеет определенный вес.

Количественная оценка записанного числа в такой системе счисления определяется как сумма произведений значения цифр, составляющих запись числа, умноженных на вес позиции, в которой располагается цифра.

Примером такой системы счисления является широко используемая десятичная система счисления. Например, количественная оценка десятичного числа

$3959_{10}$

определяется как

$3 \cdot 1000 + 9 \cdot 100 + 5 \cdot 10 + 9 \cdot 1$, где 1000, 100, 10, 1 - соответственно веса четвертого, третьего, второго, первого разрядов записи оцениваемого числа.

Десятичная *система счисления* является также системой с равномерно распределенными весами, которые характеризуются тем, что соотношение весов двух любых соседних разрядов имеют для такой системы одинаковое значение. Это соотношение называется основанием системы счисления, которое в дальнейшем будем обозначать как «q».

Общая запись числа в системе с равномерно распределенными весами имеет вид

$$N_q = A_n A_{n-1} \ldots A_2 A_1 A_0. \qquad (1)$$

Значение такого числа определяется как

$$N_q = A_n \cdot q^n + A_{n-1} \cdot q^{n-1} + A_{n-2} \cdot q^{n-2} + ..... A_2 \cdot q^2 + A_1 \cdot q^1 + A_0 \cdot q^0, \quad (2)$$

где $A_i$ - цифра записи числа, удовлетворяющая условию

$0 <= A_i <= (q-1)$;

q - основание системы счисления.

При q =10 A изменяется в диапазоне от 0 до 9, т.е. до (10-1).

Запись числа N в виде (1) называется кодированной, а запись в форме (2) называется расширенной записью.

Помимо q = 10 (десятичная *система счисления*) возможны другие значения для основания системы счисления:

- двоичная *система счисления*;
- восьмеричная *система счисления*;
- шестнадцатеричная *система счисления* и т.д.

Для обозначения цифр в различных системах счисления в качестве цифр используются обозначение соответствующих цифр десятичной системы счисления - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а в случае, когда десятичных цифр «не хватает» (для систем счисления с основанием q, большим чем 10), для цифр, превышающих 9, вводятся дополнительные обозначения, например, для q =16 это будут обозначения A, B, C, D, E, F, которые соответствуют шестнадцатеричным цифрам, десятичные эквиваленты которых равны соответственно 10, 11, 12, 13, 14, 15.

В связи с тем, что в дальнейшем изложении будут использоваться различные системы счисления, примем обозначение:

$N_q$ - число N, представленное в системе счисления с основанием q.

Примеры записи чисел в различных системах счисления:

$N_2 = 10011011 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$,

$N_8 = 471025 = 4 \cdot 8^5 + 7 \cdot 8^4 + 1 \cdot 8^3 + 0 \cdot 8^2 + 2 \cdot 8^1 + 5 \cdot 8^0$,

$N_{16} = 84FE4A = 8 \cdot 16^5 + 4 \cdot 16^4 + F \cdot 16^3 + E \cdot 16^2 + 4 \cdot 16^1 + A \cdot 16^0$,

$N_{10} = 35491 = 3 \cdot 10^4 + 5 \cdot 10^3 + 4 \cdot 10^2 + 9 \cdot 10^1 + 1 \cdot 10^0$.

На основании вышеизложенного можно заключить, что запись одного и того же числа в различных системах счисления будет тем длиннее, чем меньше основание системы счисления. Например, число N, десятичное значение которого равно 2063, в различных системах счисления представляется как

$N = 2063_{10} = 100000001111_2 = 4017_8 = 80F_{16}$ .

При работе с различными системами счисления полезно помнить соотношения, приведенные в табл. 1.

Таблица 1

| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^n$ | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |

Человек в своей практической деятельности наиболее часто использует *десятичную* систему счисления. *Двоичная система счисления* является удобной для обработки информации в ЭВМ. Промежуточное место между этими системами занимает *двоично-десятичная система* счисления. Эта система в принципе является десятичной, но отдельные десятичные цифры в ней записываются в виде набора двоичных разрядов. Существуют различные двоично-десятичные системы, которые отличаются способом представления набором двоичных разрядов десятичных цифр. Наиболее широкое распространение получила двоично-десятичная система 8,4,2,1. Данная система характеризуется тем, что отдельные десятичные цифры в ней представляются их четырех битовым двоичным эквивалентом, как это показано в табл. 2.

Таблица 2

| N п.п. | q = 2 | q = 8 | q = 16 | q = 10 | Десятичный эквивалент | Двоичный эквивалент |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0000 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0001 |
| 3 | | 2 | 2 | 2 | 2 | 0010 |
| 4 | | 3 | 3 | 3 | 3 | 0011 |
| 5 | | 4 | 4 | 4 | 4 | 0100 |
| 6 | | 5 | 5 | 5 | 5 | 0101 |
| 7 | | 6 | 6 | 6 | 6 | 0110 |
| 8 | | 7 | 7 | 7 | 7 | 0111 |
| 9 | | | 8 | 8 | 8 | 1000 |
| 10 | | | 9 | 9 | 9 | 1001 |
| 11 | | | A | | 10 | 1010 |
| 12 | | | B | | 11 | 1011 |
| 13 | | | C | | 12 | 1100 |
| 14 | | | D | | 13 | 1101 |
| 15 | | | E | | 14 | 1110 |
| 16 | | | F | | 15 | 1111 |

Например, десятичное число
804714

в двоично-десятичной системе 8,4,2,1 представляется в виде

1000 0000 0100 0111 0001 0100.

В дальнейшем для сокращения будем использоваться название «двоично-десятичная система», имея в виду двоично-десятичную систему 8,4,2,1.

## 2. Перевод чисел из одной системы счисления в другую

Наличие различных систем счисления предполагает использование разных способов перевода записи числа из одной системы счисления в другую. Для этой цели применяются следующие способы преобразований:

– метод преобразования с использованием весов разрядов в исходной и в искомой записи числа;

– метод деления (умножения) на новое основание;

– метод с использованием особого соотношения заданной и искомой систем счисления.

### 2.1. Метод преобразования с использованием весов разрядов

Метод преобразования с использованием весов разрядов записи числа в исходной и в искомой системах предполагает использование расширенной записи числа (2) в некоторой системе счисления.

Метод имеет две разновидности в зависимости от того, какая система счисления (исходная или искомая) является более привычной. Если более привычной является искомая система счисления, то на основании расширенной записи исходного числа подсчитываются значения её отдельных разрядов в новой системе счисления. Далее полученные значения суммируются.

Например, при преобразовании целого двоичного числа

$N_2 = 110011010$

в десятичную систему счисления исходное число представляется в расширенной записи

$N = 2^8 + 2^7 + 2^4 + 2^3 + 2^1$

и рассчитывается вес отдельных (ненулевых) двоичных разрядов в десятичной системе счисления:

256 , 128, 16, 8, 2.

Затем искомая запись числа определяется как сумма весов всех ненулевых разрядов записи числа в заданной системе счисления:

$256 + 128 + 16 + 8 + 2 = 410.$

При преобразовании правильных дробей в принципе используется тот же подход, но при расчете весов отдельных разрядов берутся отрицательные степени основания счисления.

Кроме того, учитывая, что при преобразовании правильных дробей в общем случае результат получается неточный, перед началом преобразования необхо-

димо подсчитать количество разрядов представления числа в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была бы не более половины единицы младшего разряда в исходной записи числа.

Например, при использовании двоичной и десятичной систем счисления берется соотношение, согласно которому один десятичный разряд соответствует точности представления четырехразрядного двоичного числа.

При преобразовании правильных дробей сначала ищется предварительное значение представления заданного числа в новой системе счисления с количеством разрядов, на единицу большим, чем расчетная разрядность представления числа в новой системе счисления. Дополнительный разряд в предварительном результате преобразования используется для округления, позволяющего с рассчитанным числом разрядов найти окончательный результат.

*Пример*

Представить правильную двоичную дробь 0.101 в десятичной системе счисления.

Перед началом преобразования определяется, что разрядность записи заданного числа в новой системе счисления должна быть равна 1, поэтому сначала ищется предварительная запись заданного числа в новой системе счисления с двумя десятичными разрядами

$$0.101_2 = 2^{-1} + 2^{-3} = 0.5 + 0.13 = 0.63,$$

и после округления

$$0.101_2 = 0.6_{10}.$$

Если более привычной является исходная система счисления, то запись заданного числа в новой системе счисления определяется разряд за разрядом, начиная со старшего. Первым значащим разрядом будет являться разряд с максимально возможным весом, но не превышающим значения преобразуемого числа. При этом, определив старшую цифру (старший разряд) с ненулевым значением, из исходного числа вычитаем вес этого разряда, таким образом формируя остаток, который должен быть представлен еще не найденным младшим разрядом искомой записи числа в новой системе счисления. Далее, используя полученный остаток, аналогичным приемом ищем второй старший разряд записи числа в новой системе счисления, определяем новый остаток и переходим к определению следующего разряда и т.п. Процесс этот напоминает процедуру взвешивания некоторого тела посредством уравновешивания его веса с помощью эталонных гирь.

*Примеры*

Найти двоичный эквивалент десятичного числа:

$436_{10} =$____?____ 2.

*Решение*

Первый (старший) разряд, имеющий значение 1 в искомой двоичной записи числа, будет разряд весом $2^8 = 256$. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 180 (180 - остаток, полученный как 436-256).

Второй разряд с весом $2^7$ - 128 будет иметь в искомой двоичной записи числа значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 52 ( 52 - остаток, полученный как 188 - 128).

Третий разряд с весом $2^5$ - 64 будет иметь в искомой двоичной записи числа значение 0.

Четвертый разряд с весом $2^5$ - 32 будет иметь в искомой двоичной записи числа значение 1, а остаток - 20.

Пятый разряд с весом $2^4$ - 16 будет иметь в искомой двоичной записи числа значение 1, а остаток - 4.

Шестой разряд с весом $2^3 = 8$ будет иметь в искомой двоичной записи числа значение 0.

Седьмой разряд с весом $2^2 = 4$ будет иметь в искомой двоичной записи числа значение 1, а остаток - 0.

Восьмой разряд с весом $2^1 = 2$ будет иметь в искомой двоичной записи числа значение 0.

Девятый разряд с весом $2^0 = 1$ будет иметь в искомой двоичной записи числа значение 0.

Таким образом,

$436_{10} = 110110100_2$.

*Пример*

Найти двоичный эквивалент числа $0.7_{10}$ =_____?_____ $_2$ .

Предварительный результат ищется с точностью до пяти двоичных разрядов, причем пятый разряд используется только для округления при переходе к четырехразрядному окончательному результату.

Первый (старший) разряд с весом $2^{-1} = 0.5$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 0.2 (0.2 - остаток, полученный как 0.7 - 0.5 = = 0.2).

Второй (старший) разряд с весом $2^{-2} = 0.25$ в искомой двоичной записи числа будет иметь значение 0.

Третий разряд с весом $2^{-3} = 0.13$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 0.07 (0.07 - остаток, полученный как 0.2 - 0.13).

Четвертый разряд с весом $2^{-4} = 0.06$ в искомой двоичной записи числа будет иметь значение 1, а остаток - 0.01.

Пятый разряд с весом $2^{-5} = 0.03$ искомой двоичной записи числа будет иметь значение 0.

Таким образом, десятичное число $0.7_{10} = 0.10110_2$.

После округления имеет место
$0.7_{10} = 0.1011_2$.

### 2.2. Метод деления (умножения) на новое основание

Метод деления (умножения) имеет две разновидности соответственно для преобразования целых и дробных чисел.

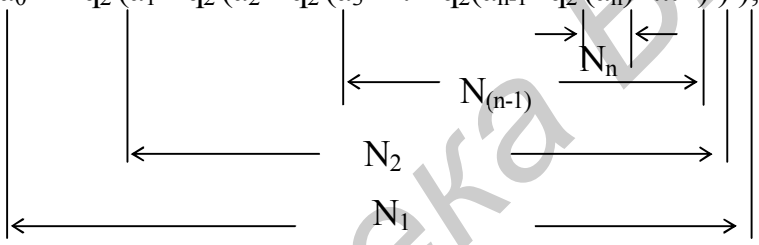
**Преобразование целых чисел**

Задачу представления числа N, заданного в системе $q_1$, в системе счисления с основанием $q_2$ можно рассматривать как задачу поисков коэффициентов полинома, представляющего собой расширенную запись числа N в системе счисления $q_2$:

$$N_{q1} = a_0 + a_1 \cdot q_2^{1} + a_2 \cdot q_2^{2} + \ldots + a_{n-2} \cdot q_2^{n-2} + a_{n-1} \cdot q_2^{n-1} + a_n \cdot q_2^{n} = Nq_2. \quad (3)$$

Введем скобочную форму для выражения (3):

$$N_{q1} = N_{q2} = a_0 + q_2 (a_1 + q_2 (a_2 + q_2 (a_3 + \ldots + q_2 (a_{n-1} + q_2 (a_n) \ldots ) ) ) );$$




Обозначим выражение в первой скобке как $N_1$, выражение во второй скобке как $N_2$, выражение в третьей скобке - как $N_3$ и т.д., выражение в (n-1)-й скобке - как $N_{(n-1)}$, выражение в n-й скобке - как $N_n$. Теперь, основываясь на выражении (3), можно утверждать, что при делении $N_{q1}/ q_2$ будет получена целая часть частного $\text{int}(N_{q1}/q_2)$ и остаток $\text{rest}(N_{q1}/q_2)$. Это можно записать как

$N_{q1}/q_2 \rightarrow \text{int}(N_{q1}/q_2)$,- целая часть частного $N_1$, и остаток $\text{rest}(N_{q1}/q_2)$, равный $a_0$;

Аналогично для остальных скобок будем иметь:

$N_1/q_2 \rightarrow \text{int}(N_1/q_2)$, равно $N_2$ и остаток $\text{rest}(N_1/q_2)$, равный $a_1$;

$N_2/q_2 \rightarrow \text{int}(N_2/q_2)$, равно $N_3$, и остаток $\text{rest}(N_2/q_2)$, равный $a_2$;

.....

$N_{(n-2)}/q\,2 \rightarrow \text{int}(N_{(n-2)}/q_1)$, равно $N_{(n-1)}$), остаток $\text{rest}(N(n=2)/q_1)$, равный $a_{(n-2)}$;

$N_{(n-1)}/q_2 \rightarrow \text{int}(N_{(n-1)}/q_2) = N_n = a_n$, и остаток $\text{rest}(N_{(n-1)}/q_2)$, равный $a_{(n-1)}$; при этом $N_n < q_2$.

Отсюда вытекает правило формирования коэффициентов полинома (3) или разрядов записи заданного числа N в системе счисления с основанием $q_2$:

– необходимо разделить исходное число $N_{q1}$ на новое основание $q_2$, при этом получив целое частное и остаток;

– полученный остаток снова необходимо разделить на $q_2$, процесс деления продолжается до тех пор, пока частное будет не меньше нового основания $q_2$. Если очередное сформированное частное будет меньше, чем $q_2$, то процесс формирования записи заданного числа в новой системе с основанием $q_2$ считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом:

– в качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.

*Пример*

Найти запись в двоичной форме десятичного числа $N_{10} = 436$.

*Решение*

Делим сначала исходное число $N_{10}$, а затем получаемые частные на значение нового основания 2 до получения частного со значением, меньше чем два:

436/2→int(436/2) =218 и rest (436/2) = 0;

218/2→int(218/2) =109 и rest (218/2) = 0;

109/2→int(109/2) =54 и rest (109/2) = 1;

54/2→int(54/2) = 27и rest (54/2) = 0;

27/2→int(27/2) = 13 и rest (27/2) = 1;

13/2→int(13/2) = 6 и rest (13/2) = 1;

6/2→int(6/2) = 3 и rest (6/2) = 0;

3/2→int(3/2) = 1 и rest (3/2)= 1.

Таким образом, 436 = 11 0110100.

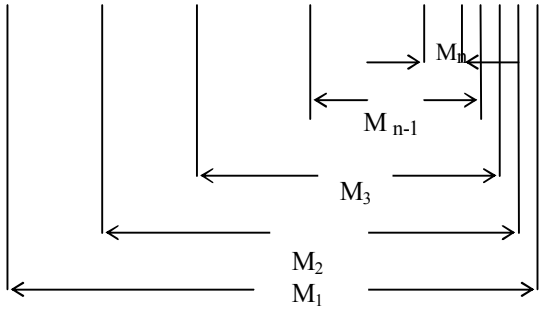
**Преобразование дробных чисел**

Задача представления дробного числа $M_{q1}$, заданного в системе $q_1$, в системе счисления с основанием $q_2$, можно рассматривать как задачу поиска коэффициентов полинома, представляющего собой расширенную запись числа M в системе счисления $q_2$:

$$b_1 \cdot q_2^{-1} + b_2^{-2} \cdot q_2 + b_3 \cdot q_2^{-3} + \ldots + b_{n-2} \cdot q_2^{-(n-2)} + b_{n-1} \cdot q_2^{-(n-1)} + b_n \cdot q_2^{-n} = M_{q2}. \qquad (4)$$

Введем скобочную форму для выражения (4):

$$M_{q2} = M_{q1} = q_2^{-1}(b_1 + q_2^{-1}(b_2 + q_2^{-1}(b_3 + \ldots + q_2^{-1}(b_{n-1} + q_2^{-1}(b_n))\ldots))));$$



Обозначим выражение в первой скобке как $M_1$, выражение во второй скобке - как $M_2$, выражение в третьей скобке - как $M_3$ и т.д., выражение в (n-1)-й скобке как $M_{n-1}$, выражение в n-й скобке как $M_n$.

Число $M_{q1}$ - правильная дробь, поэтому при умножении $M\ q_1 \cdot q_2$ будет получено произведение, в общем случае состоящее из целой части $int(M\ q_1 \cdot q_2)$ и дробной части $DF\ (M\ q_1 \cdot q_2)$. Использование введенных обозначений позволяет записать:

$$M\ q_1 \cdot q_2 = (int(M\ q_1 \cdot q_2) = b_1) + (DF(M\ q_1 \cdot q_2) = M_1);$$

аналогично для остальных скобок будем иметь:

$$M_1 \cdot q_2 = (int(M_1 \cdot q_2) = b_2) + (DF(M_1 \cdot q_2) = M_2);$$

$$M_2 \cdot q_2 = (int(M_2 \cdot q_2) = b_3) + (DF(M_2 \cdot q_2) = M_3);$$

$$M_3 \cdot q_2 = (int(M_3 \cdot q_2) = b_4) + (DF(M_3 \cdot q_2) = M_4);$$

.....

$$M_{n-2} \cdot q_2 = (int(M_{n-2} \cdot q_2) = b_{n-1}) + (DF(M_{n-2} \cdot q_2) = M_{n-1});$$

$$M_{n-1} \cdot q_2 = (int(M_{n-1} \cdot q_2) = b_n) + (DF(M_{n-1} \cdot q_2) = M_n);$$

$$M_n \cdot q_2 = (int(M_n \cdot q_2) = b_{n+1}) + (DF(M_n \cdot q_2) = M_{n+1}).$$

Отсюда вытекает следующее правило формирования коэффициентов полинома, которые одновременно являются разрядами записи заданного числа M в системе счисления с основанием $q_2$:

– определяется количество разрядов «n» в записи числа $M_{q2}$ в новой системе счисления (см. подразд.1.2.1);

– исходное число $M_{q1}$ умножается на $q_2$, при этом будет получено смешанное число;

– дробная часть полученного произведения снова умножается на $q_2$ и т.д.; процесс умножения повторяется n раз. В качестве искомых разрядов новой записи числа используются результаты выполненных операции деления следующим образом:

– в качестве первого старшего разряда искомой записи числа в новом основании берется значение целой части первого произведения, в качестве второго старшего разряда искомой записи числа в новом основании берется значение целой части второго произведения и т.д.

*Пример*

Найти запись в двоичной форме десятичного числа $M_{10}$ =0.7.

Определяем количество разрядов числа $M_2$. Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичном основании

должна содержать четыре разряда. Учитывая округление, ищем предварительный двоичный эквивалент с пятью разрядами.

Умножаем исходное число $M_{10}$, а затем дробные части последовательно получаемых произведений на новое основание 2. Выполняется пять таких операций умножения, в результате получаем:

$0.7 \cdot 2 = 1.4$ (int$(0.7 \cdot 2) = 1$ и DF $(0.7 \cdot 2) = 0.4$);

$0.4 \cdot 2 = 0.8$ (int$(0.4 \cdot 2) = 0$ и DF(rest $(0.4 \cdot 2) = 0.8$);

$0.8 \cdot 2 = 1.6$ (int$(0.8 \cdot 2) = 1$ и DF(rest $(0.8 \cdot 2) = 0.6$);

$0.6 \cdot 2 = 1.2$ (int$(0.6 \cdot 2) = 1$ и DF(rest $(0.6 \cdot 2) = 0.2$);

$0.2 \cdot 2 = 0.4$ (int$(0.2 \cdot 2) = 0$ и DF(rest $(0.2 \cdot 2) = 0.4$).

Таким образом, $0.7 = 0.10110$, а окончательный результат перехода в двоичную систему будет

$0.7_{10} = 0.1011_2$.

### 2.3. Метод с использованием особого соотношения оснований заданной и искомой систем счисления

Данный метод применим тогда, когда исходное $q_1$ и новое $q_2$ основания могут быть связаны через целую степень, т.е. когда выполняется условие

$q_1^m = q_2$ (*условие 1*) или $q_2^m = q_1$ (*условие 2*).

Если имеет место *условие 2*, то для преобразования заданного числа

$N = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$

запись его в новом основании $q_2$ ищется следующим образом:

— каждому разряду $a_i$ исходной записи числа ставится в соответствие его m-разрядный эквивалент в системе счисления с основанием $q_2$;

— исходная запись всего заданного числа формируется за счет объединения всех полученных m- разрядных групп.

Если имеет место *условие 1*, то запись заданного числа

$N = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$

в новом основании $q_2$ ищется следующим образом:

— исходная запись числа разбивается на группы по m разрядов, двигаясь от точки вправо и влево (недостающие разряды в крайних группах (слева и справа) дополняются нулями;

— каждой полученной группе ставится в соответствие цифра новой системы счисления;

— искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

*Пример 1*

Найти двоичный эквивалент восьмеричного числа $67401.64_8$.

*Решение*

Основания исходной и новой систем счисления можно выразить через целую степень:

$2^3 = 8$.

Поэтому применяем третий метод для случая перехода из системы с большим основанием в систему с меньшим основанием. Ставим в соответствие каждой цифре исходной записи числа трехразрядный двоичный эквивалент:

| 6 | 7 | 4 | 0 | 1 | . | 6 | 4 |
|---|---|---|---|---|---|---|---|
| 110 | 111 | 100 | 000 | 001 | | 110 | 100 |

Формируем окончательный результат посредством объединения полученных трехразрядных двоичных чисел в единый двоичный эквивалент:

$67401.64_8 = 110111100000001.110100$.

*Пример 2*

Найти шестнадцатеричный эквивалент двоичного числа

$N = 111001011101 10.111011001_2$.

*Решение*

Основания исходной и новой систем счисления можно выразить через целую степень:

$2^4 = 16$.

Поэтому применяем третий метод для случая перехода из системы с меньшим основанием в систему с большим основанием. Разбиваем исходную запись числа на группы по четыре разряда вправо и влево от точки, в крайних левой и правой группах недостающие разряды заполняем нулями и каждой полученной группе из четырех разрядов ставим в соответствие цифру шестнадцатеричной системы счисления

| 0011 | 1001 | 0111 | 0110 | . | 1110 | 1100 | 1000 |
|---|---|---|---|---|---|---|---|
| 3 | 9 | 7 | 6 | | E | C | 8 |

Формируем окончательный результат посредством объединения полученных цифр в единый шестнадцатеричный эквивалент

$11100101110110.111011001_2 = 3\ 9\ 7\ 6.\ E\ C\ 8_{16}$.

*Пример3*

Найти шестнадцатеричный эквивалент числа $67401.64_8$, представленного в восьмеричной системе счисления.

*Решение*

Основания исходной $q_1$ и новой $q_2$ систем счисления не могут быть связаны через целую степень, поэтому напрямую третий метод перехода неприменим. Однако существует система с двоичным основанием, для которой допустим третий метод перехода и восьмеричную (исходную для данного примера), и в шестнадцатеричную (новую систему для данного примера) системы счисления, т.к.

$2^3 = 8$ и $2^4 = 16$.

Поэтому в данном случае для решения поставленной задачи целесообразно использовать два быстрых перехода из восьмеричной системы счисления в двоичную (промежуточную), а затем из двоичной системы счисления в шестнадцатеричную третьим методом. Это будет гораздо быстрее, чем использовать для заданного преобразования второй или третий метод.

Таким образом, поставленная задача решается в следующем порядке:

$67401.64_8 = 110\ 111\ 100\ 000\ 001\ .\ 110\ 100_2$;

$110\ 111\ 100\ 000\ 001\ .\ 110\ 100_2 = 0110\ 1111\ 0000\ 0001.\ 1101\ 0000_2 =$
$= 6\ F\ 0\ 1.\ D\ 0_{16}$, т.е.:

$67401.\ 64_8 = 6F01.\ D0_{16}$.

*Пример 4*

Найти двоичный эквивалент числа $6740_{10}$.

*Решение*

Основания исходной $q_1$ и новой $q_2$ систем счисления не могут быть связаны через целую степень, поэтому третий метод перехода неприменим. В принципе здесь целесообразно использовать второй метод - метод деления на новое основание. Однако в этом случае потребуется большое количество операций деления на два. Для сокращения количества операций деления может оказаться целесообразным решить эту задачу за счет перехода с использованием второго метода в промежуточную шестнадцатеричную систему счисления, а затем, используя третий метод, быстро перейти в заданную двоичную систему счисления:

– выполняем переход в промежуточную систему счисления:

6740/16 = 421 (остаток 4);

421/16 = 26 (остаток 5);

26/16 = 1 (остаток 10);

– для промежуточной системы счисления имеем:

$6740_{10} = 1A54_{16}$,

– выполняем переход из промежуточной системы счисления в заданную:

$1A54_{16} = 0001\ 1010\ 0101\ 0100_2$.

Как видно из вышеприведенного, для заданного перехода потребовалось выполнить только 4 операции деления на 16 вместо 13-ти операций деления на 2.

# 3. Арифметические операции над положительными числами

## 3.1. Операции сложения в двоичной системе счисления

При выполнении любой операции результат ищется согласно соответствующим правилам, которые удобно представлять в табличной форме, где для всех возможных комбинаций значений одноразрядных операндов приводятся значения результата.

Правила выполнения операции сложения в двоичной системе счисления задаются в виде табл. 3.

Таблица 3

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0* |

Все возможные значения первого слагаемого задаются во второй и третьей строках первой колонки. Все возможные значения второго слагаемого задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат их сложения. В таблице знаком * отмечен случай, когда в текущем разряде результата получен ноль и имеет место перенос в ближайший старший разряд.

*Пример*

$$\begin{array}{r} 1011100110 \\ +\ \underline{0101101101} \\ 10001010011 \end{array}$$

В общем случае при формировании значения в текущем разряде результата приходится дважды применять приведенную таблицу сложения: первый раз при сложении соответствующих разрядов операндов, формируя так называемую поразрядную сумму, и второй раз - при сложении разряда сформированной поразрядной суммы и переноса, пришедшего из ближайшего младшего разряда.

При машинной реализации операции сложения сначала формируется поразрядная сумма операндов без учета переноса, далее формируется код переноса, и затем с помощью специальных логических цепей учитываются возникшие переносы. При этом перенос, возникший в некотором разряде, может изменить не только ближайший старший разряд, но и целую группу старших разрядов. В худшем случае перенос, возникший в самом младшем разряде, может изменить значение старших разрядов сформированной поразрядной суммы, вплоть до самого старшего.

При формировании поразрядной суммы и учете возникших переносов используется следующая классификация разрядов складываемых операндов:

- разряд, генерирующий перенос (оба операнда в этом разряде имеют 1);
- разряд, пропускающий перенос (операнды в этом разряде имеют разные значения);
- разряд, блокирующий распространение переноса (операнды в этом разряде имеют одинаковые значения).

### 3.2. Операция вычитания

Правила выполнения операции сложения в двоичной системе счисления задаются в виде табл. 4.

Таблица 4

| - | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1* | 0 |

Все возможные значения вычитаемого задаются во второй и третьей строках первой колонки. Все возможные значения уменьшаемого задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат вычитания второго операнда из первого операнда. В таблице знаком * отмечен случай, когда в текущем разряде результата получена единица при заёме из ближайшего старшего разряда.

*Пример*

1000111001

\- 0101101101

0011001100

Как видно из таблицы и приведенного примера, реализация операция вычитания не сложнее операции сложения.

В ЭВМ никогда в перечне выполняемых операций арифметического устройства не присутствует одновременно операция сложения и операция вычитания. При этом, как правило, присутствует только операция сложения. Что же касается операции вычитания, то она реализуется за счет прибавления к уменьшаемому значения вычитаемого, взятого с противоположным знаком.

### 3.3. Операция умножения

Умножение в двоичной системе счисления задаются в виде табл. 5.

Таблица 5

| * | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Все возможные значения множимого задаются во второй и третьей строках первой колонки. Все возможные значения множителя задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат умножения первого операнда на второй операнд.

При умножении многоразрядных операндов, как правило, (особенно в десятичной системе счисления) используется метод, при котором формирование произведения выполняется за счет суммирования частичных произведений, которые формируются посредством умножения множимого на отдельные разряды множителя с учетом веса соответствующего разряда множителя.

Таблица умножения одноразрядных операндов в двоичной системе существенно упрощает задачу формирования частичного произведения:

частичное произведение для разряда множителя равняется нулю, если этот разряд равен нулю;

частичное произведение для разряда множителя равняется множимому, взятому с соответствующим весом, если разряд множителя равен единице.

При последовательном способе формирования частичных произведений, последние могут рассчитываться поочередно для отдельных разрядов множителя,

начиная с младшего или старшего разряда. При десятичном основании, как правило, формирование частичных произведений осуществляется, начиная с младшего разряда множителя.

*Пример1*

Найти произведение двоичных чисел 1011 и 1101, начиная формирование частичных произведений со старшего разряда множителя.

*Решение*

При формировании частичных произведений, начиная со старшего разряда множителя, процесс формирования произведения заданных операндов можно представить следующим образом:

```
  1 0 1
Ч 1 10 1
+1 0 1 1
+  1 0 1 1
+   0 0 0 0
+      1 0 1 1
  1 0 0 0 1 1 1 1
```

При реализации умножения рассматриваемым способом требуется использование 2n-разрядного сумматора для подсчета промежуточных и конечного произведения и 2n-разрядного сдвигающего регистра для хранения множителя.

*Пример 2*

Найти произведение двоичных чисел 1011 и 1101, начиная формирование частичных произведений с младшего разряда множителя и применяя учет сформированных частичных произведений по мере их формирования.

*Решение*

Реализация данного метода умножения требует использовать 2n-разрядный сумматор для последовательного, от такта к такту, формирования 2n-разрядного произведения и 2n- разрядный регистр для хранения и сдвига влево множимого.

```
      1 0 1 1
    Ч 1 1 0 1
      0 0 0 0
  + 1 0 1 1
      1 0 1 1
  +0 0 0 0
    0 1 0 1 1
+ 1 0 11
```

```
    1 1 0 1 1  1
+ 1 0 1 1_____
  1 0 0 0 1 1 1 1
```

Реализация данного метода умножения требует использовать 2n-разрядный сумматор для последовательного, от такта к такту формирования 2n-разрядного произведения и 2n-разрядный регистр для хранения и сдвига влево множимого.

В данном примере для того, чтобы учесть то, что очередной разряд множителя имеет вес, в два раза больший, чем предыдущий разряд, его частичное произведение учитывается со сдвигом на один разряд влево при суммировании с промежуточным результатом. В таком случае говорят, что умножение выполняется со сдвигом множимого.

Однако для того чтобы учесть, что очередное частичное произведение имеет вес, в два раза больший, чем предыдущее, можно при суммировании сдвигать вправо промежуточный результат. В таком случае говорят, что умножение выполнятся со сдвигом промежуточного результата. При использовании такого подхода, умножение чисел 1101 и 1011представляется в виде следующих действий:

```
      1 0 1 1          множимое
      Ч 1 1 0 1        множитель
      0 0 0 0          промежуточный результат (на-
1р.  + 1 0 1 1         чальное значение)
      1 0 1 1
      0 1 0 1    1     результат сдвига
2р.  + 0 0 0 0
      0 1 0 1    1
      0 0 1 0    11    результат сдвига
3р.  +1 0 1 1
      1 1 0 1    11
      0 1 1 0    111   результат сдвига
4р.  + 1 0 1 1
      1 0 0 0 1  111
      1 0 0 0    1111  результат сдвига
```

Вторая колонка отражает длину основной разрядной сетки (n = 4).

В первой колонке приведены номера отрабатываемых разрядов множителя, начиная с младшего. Эти номера отмечают строки, в которых учитывается частичное произведение, соответствующее этому разряду множителя. В этой же первой колонке расположены единицы переполнения, возникающие при суммировании промежуточного результата и очередного частичного произведения, сформированного для соответствующего отрабатываемого разряда множителя (в данном случае единица переполнения имеется при отработки четвертого (4р.), самого старшего, разряда множителя).

Вторая колонка представляет собой основную разрядную сетку.

В третьей колонке представлены разряды промежуточных и конечного произведений, «вытолкнутых» за пределы основной разрядной сетки в процессе выполнения очередного сдвига промежуточного произведения.

Из приведенного примера видно, что выталкиваемые за пределы разряды промежуточных произведений в дальнейшем не изменяются и их значение не влияет на значение суммы, формируемой в пределах основной разрядной сетки. Поэтому для реализации этого метода умножения требуется n-разрядный сумматор, обеспечивающий суммирования только в пределах основной разрядной сетки.

Аналогично умножению, начиная с младшего разряда множителя, при умножении со старших разрядов можно заменить сдвиг вправо множителя на сдвиг влево промежуточного произведения.

Операция умножения в общем случае дает точный результат – 2n-разрядное произведение, где n - разрядность операндов.

В ЭВМ при выполнении различных операций, в том числе и операции умножения, разрядность операндов и результатов одинаковая. Это означает, что при умножении правильных дробей последние, младшие, n разрядов отбрасываются а старшие n разрядов округляются с учетом отбрасываемых младших разрядов.

Например, произведение

$0.1011 \cdot 0.1101 = 0.10001111$

представляется в n-разрядном варианте как

0.1001.

В этом случае операция умножения считается приближенной.

Иногда в ЭВМ формируется точное произведение; в таких случаях специально оговаривается, что произведение формируется с удвоенной разрядной сеткой.

При операндах, представленных в виде целых чисел, формирование произведения с такой же разрядностью, что и разрядность операндов, в качестве конечного значения произведения также используются n старших разрядов, которые округляются с учетом отбрасываемых младших разрядов, но при этом масштаб представления произведения изменяется - он умножается на $2^n$. Это является одной из причин, по которой в ЭВМ, как правило, используется представление чисел в виде правильных дробей.

Возможные методы реализации операции умножения можно классифицировать по двум признакам:

– начиная с какого разряда (старшего или младшего) выполняется отработка множителя;

– что сдвигается - множимое или промежуточное произведение.

Используя эти два классификационных признака, можно выделить четыре метода умножения:

– *умножение с младших* разрядов множителя со сдвигом множителя; при реализации данного метода требуется 2n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, 2n-разрядный регистр для хранения и сдвига множимого и n- разрядный регистр для хранения множителя;

– *умножение с младших* разрядов множителя со сдвигом промежуточного произведения; при реализации данного метода требуется n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, n-разрядный регистр для хранения множимого и n-разрядный регистр для хранения множителя;

– *умножение со старшего* разряда множителя со сдвигом множимого; при реализации данного метода требуется 2n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, 2n-разрядный регистр для хранения и сдвига множимого и n- разрядный регистр для хранения множителя;

– *умножение со старшего* разряда множителя со сдвигом промежуточного произведения; при реализации данного метода требуется n-разрядный сумматор, 2n-разрядный регистр промежуточного произведения, n-разрядный регистр для хранения множимого и n-разрядный регистр для хранения множителя.

Используя рассмотренные методы умножения, легко показать, что при умножении правильных дробей 0.1011 на 0.1101 будет получено произведение, равное 0.10001111.

Обобщенная структурная схема устройства для реализации операции умножения приведена на рис. 1.

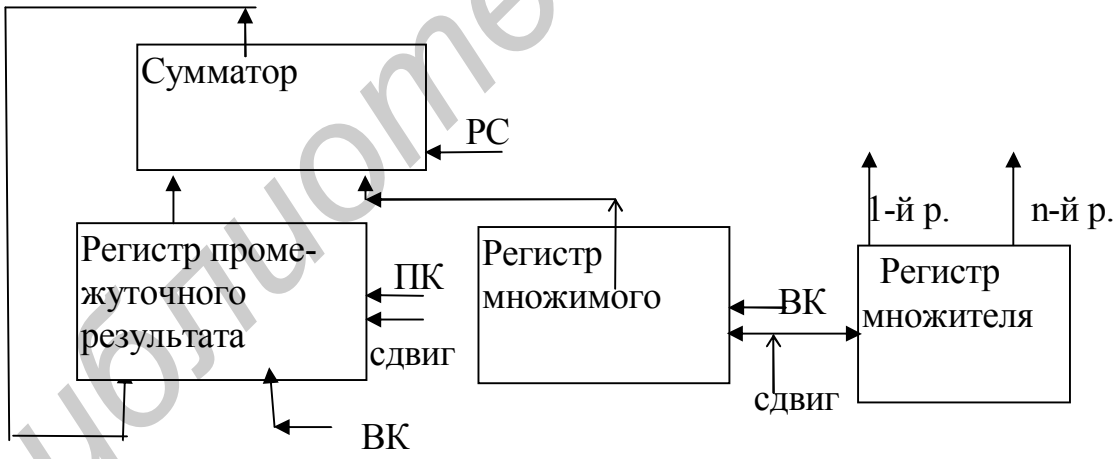



Рис. 1

На рисунке приняты следующие обозначения:

– ПК - сигнал разрешения приема кода;

– 1 р. - выход старшего (1-го) разряда множителя (используется при умножении со старших разрядов множителя);

– n-й р. - выход младшего ( n-го) разряда регистра (используется при умножении с младших разрядов множителя).

### 3.4. Деление двоичных чисел

Деление в принципе является неточной операцией, поэтому при её выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению.

Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирования частного двоичных операндов реализуется гораздо проще, чем в десятичной системе, т.к.:

− упрощается процедура подбора очередной цифры вследствие того, что в двоичной системе очередной цифрой может быть одна из двух - либо 0, либо 1;

− упрощается процедура умножения найденной цифры частного на делитель.

*Пример*

Найти частное от деления двоичных чисел 0.1001 на 0.1101.

*Решение*

По умолчанию считается, что разрядность результата и операндов одинаковая, поэтому окончательный результат должен иметь в данном случае 4 разряда. Учитывая необходимость округления, найдем дополнительный пятый разряд, на основании которого выполним округление.

```
0. 1 0 0 1        0. 1101   - делитель
 - 1 1 0 1          0. 10110  - частное
     1 0 1 0
   - 1 1 0 1
       1 1 1
     - 1 1 0 1
           1   - остаток
```

Таким образом, 0.1001/0.1101 = 0.1011

### 3.5. Арифметика с положительными двоично-десятичными числами

В ЭВМ часто предусматривается обработка чисел не только в двоичной системе счисления, но в двоично-десятичной. При этом, как правило, стремятся реализовать двоично-десятичную арифметику по правилам двоичной с введением ограниченного количества коррекций.

**Сложение двоично-десятичных чисел**

Рассмотрим на конкретном примере реализацию этой операции.

*Пример*

Найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления:

A = D + C, где
D = 3927;
C = 856.

*Решение*

Составляем двоично-десятичную запись для чисел D и C:

D =3 927 = 0011 1001 0010 0111:

C = 4856 = 0100 1000 0101 0110.

Найти значение A можно, реализовав следующую последовательность операций из двоичного сложения и операции коррекции:

```
                *          **
   0011 1001 0010 0111 - D
+  0100 1000 0101 0110 - C
   1000  0001 0111 1101  -  двоичная сумма
+       0110       0110  -  коррекция
   1000 0111 1000  0011   - двоично-десятичная сумма
```

Для получения двоично-десятичной суммы A на основании результата сложения операндов по правилам двоичной арифметики необходимо добавить шестерку (0110) в те тетрады, из которых был перенос. В данном примере это вторая тетрада (отмечена *). Необходимость такой коррекции обусловливается тем, что перенос, сформированный по правилам двоичного суммирования, унес из тетрады шестнадцать, а для десятичного сложения перенос должен был унести десять, т.е. перенос, сформированный по правилам двоичной арифметики, унес лишнюю шестерку. Кроме этого, шестерка добавляется в те тетрады, в которых получено значение, большее девяти.

Такая коррекция обусловливается тем, что по правилам десятичной арифметики в таких тетрадах должен быть выработан перенос и, чтобы его выработать по правилам двоичной арифметики, в тетраду нужно добавить шестерку. Для рассмотренного примера такой тетрадой является и четвертая тетрада (отмечена **).

*Пример*

Найти разность двух десятичных чисел с использованием двоично-десятичной системы счисления:

A = C- D, где

D = 3927;

C = 4856.

*Решение*

Составляем двоично-десятичную запись для чисел D и C:

D = 3927 = 0011 1001 0010 0111:

C = 4856 = 0100 1000 0101 0110.

Найти значение B можно, реализовав следующую последовательность операций из двоичного сложения и операции коррекции:

```
                *          *
  0100 1000 0101 0110 - C
- 0011 1001 0010 0111 - D
  0000 1111 0010 1111  -  двоичная сумма
-      0110       0110 -  коррекция
  0000 1001 0010  1001   - двоично-десятичная сумма
```

Для получения двоично-десятичной разности A на основании результата вычитания операндов по правилам двоичной арифметики необходимо вычесть шестерку (0110) из тетрад, в которые пришел заем. Это обусловливается тем, что заем, сформированный по правилам двоичного вычитания, приносит в тетраду шестнадцать, а для десятичного сложения заем должен был принести в тетраду десять, т.е. заём, сформированный по правилам двоичной арифметики, принес лишнюю шестерку. Для рассмотренного примера тетрадами, в которые пришел заем и в которых необходимо выполнить коррекцию (вычесть шестерку), являются вторая и четвертая тетрады (отмечены *).

## 4. Арифметика с алгебраическими числами

### 4.1. Кодирование алгебраических чисел

Для представления чисел со знаком используются специальные коды:

- *прямой код*;
- *дополнительный код*;
- *обратный код*.

Во всех трёх случаях используется следующий формат представления числа, содержащий два поля - поле знака и поле модуля (рис. 2).

| Поле знака | Поле модуля |
|---|---|

Рис. 2

Поле знака представлено одним разрядом, в котором устанавливается 0, если число положительное, и 1, если число отрицательное.

*Поле модуля* отражает количественную оценку числа и для каждого кода формируется по-разному. Количество разрядов поля модуля определятся диапазоном изменения отображаемых чисел или точностью их представления.

В *прямой код* запись целого числа A формируется по следующему правилу:

$$[A]_{пк} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1.\ |A|, \text{ если } A < 0. \end{cases}$$

В дополнительном коде запись целого числа А формируется по следующему правилу:

$$[A]_{дк} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1\ .\ q^n + A\ , \text{ если } A < 0, \end{cases}$$

где n - разрядность модульного поля;

q - основание системы счисления;

$q^n$ - максимальная невключенная граница диапазона изменения представляемых чисел, т.к. диапазон изменение чисел А определяется как

$$q^n > |A| >= 0\ .$$

Для случая правильной дроби запись числа В в дополнительном коде имеет вид

$$[A]_{дк} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1.\ (1 + A\ ), \text{ если } A < 0, \end{cases}$$

где 1 - максимальная невключенная граница диапазона изменения представляемых чисел, т.е. диапазон изменение чисел А определяется как

$$1 > |A| >= 0.$$

В обратном коде запись целого числа А формируется по следующему правилу:

$$[A]_{ок} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1\ .\ ((q^n - 1) + A)\ , \text{ если } A < 0, \end{cases}$$

где n - разрядность модульного поля;

q - основание системы счисления;

$(2^n - 1)$ - максимальная включенная граница диапазона изменения представляемых чисел, т.е. диапазон изменение чисел А определяется как

$$(q^n - 1) >= |A| >= 0.$$

Для случая правильной дроби запись числа В в обратном коде имеет вид

$$[A]_{ок} = \begin{cases} 0.\ B, \text{ если } B >= 0; \\ 1.\ (1 - q^{-n} + B), \text{ если } B < 0, \end{cases}$$

где $(1 - q^{-n})$ - максимальная включенная граница диапазона изменения представляемых чисел, т.е. диапазон изменение чисел А определяется как

$$(1 - q^{-n}) >= |A| >= 0.$$

Легко показать, что перевод отрицательного числа из обратного или дополнительного кода в прямой выполняется по тому же правилу, что и перевод числа из прямого кода в обратный или *дополнительный код*:

– для того чтобы перевести отрицательное число из обратного в *прямой код*, необходимо дополнить его модуль до включенной границы;

– для того чтобы перевести отрицательное число из дополнительного в *прямой код*, необходимо дополнить его модуль до невключенной границы.

### 4.2. Дополнительный и обратные коды двоичных чисел

При переводе двоичных чисел в качестве включенной и невключенной границы диапазона изменения абсолютных значений представляемых чисел используется соответственно $2^n$ и $2^n$ -1.

Представление двоичных чисел в прямом и обратном кодах поясняется следующими примерами.

*Пример*

Найти запись чисел A= 532 и B = -150 в прямом, дополнительном и обратном двоичных кодах.

*Решение*

Найдем запись заданных чисел в двоичной системе:

$A = 532_{10} = 1000010100_2$, $B = -150_{10} = -10010110_2$.

Если считать, что представляются в заданных кодах только A и B, то разрядность n - модульного поля должна соответствовать разрядности двоичной записи большего числа, т.е. n=10.

Найдем запись заданных чисел в прямом коде:

$[A]_{пр} = 0.1000010100$, $[B]_{пр} = 1.0010010110$.

Найдем запись заданных чисел в дополнительном коде:

$[A]_д = 0.1000010100$,

$[B]_д$ : для определения модульной части прибавим к невключенной границе диапазона ( $2^n = 10000000000$) число B:

10000000000 + (-10010110) = 1101101010

и тогда $[B]_д = 1.1101101010$.

Найдем запись заданных чисел в обратном коде:

$[A]_о = 0.1000010100$,

$[B]_о$ : для определения модульной части прибавим к включенной границе диапазона ($2^n$ –1 = 1111111111) число B:

1111111111 +(-10010110) = 1101101001

и тогда $[B]_о = 1.\ 1101101001$.

Анализируя запись модульной части отрицательного числа C в обратном коде, можно заключить, что она представляет собой инверсию модульной части записи этого числа в прямом коде, т.е. 0 заменяются 1, а 1 заменяются 0. Отсюда вытекает правило формирования модуля обратного кода отрицательного двоичного числа:

*чтобы сформировать модульную часть записи отрицательного числа в обратном коде, достаточно в модульной части записи этого числа в прямом коде взять обратные значения всех двоичных разрядов, т.е. необходимо проинвертировать модуль прямого кода.*

Переход от обратного кода отрицательного числа к представлению в прямом коде осуществляется по тому же правилу, т.е. необходимо проинвертировать модуль записи числа в дополнительном коде.

Если сравнить запись модульных частей дополнительного и обратного кодов отрицательного числа В, то можно заметить что они отличаются на значение, соответствующее единицы младшего разряда. Отсюда вытекает правило формирования модуля дополнительного кода отрицательного числа:

*чтобы сформировать модульную часть записи отрицательного числа в дополнительном коде, достаточно в модульной части записи этого числа в прямом коде взять обратные значения всех двоичных разрядов, т.е. необходимо проинвертировать модуль прямого кода, и к полученному коду прибавить 1 в младший разряд.*

Переход от дополнительного кода отрицательного числа к прямому осуществляется по тому же правилу, т.е. необходимо проинвертировать модуль записи числа в дополнительном коде, и к полученному коду прибавить 1 в младший разряд.

При выполнении операций над числами со знаком в ЭВМ используются прямой, обратный и дополнительный коды. Как правило, информация в памяти хранится в прямом коде, а при выполнении операций применяется или обратный, или дополнительный код.

### 4.3. Операции с двоичными числами в дополнительном коде

При использовании *дополнительного* или *обратного* кода операция вычитания заменяется операцией сложения с изменением знака второго операнда. При сложении чисел, представленных в дополнительном коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимание на границу, разделяющую знаковое и модульные поля. Переполнение знакового поля, т.е. перенос, возникший из крайнего левого разряда, игнорируется. В результате такого сложения будет получен *дополнительный код* суммы заданных операндов.

*Пример*

Найти значения для С1, С2, С3, С4, определяемых выражениями:

С1 = А+В, С2 = А-В, С3 = В-А, С4 = -А-В,

если $A = 57_{10}$, $B = -210_{10}$. При выполнении операций использовать двоичный *дополнительный* код. Результат представить в *прямом* коде.

*Решение*

Преобразуем заданные числа в двоичную систему счисления:

$A = 57_{10} = 111001_2$, $B = -210_{10} = -11010010_2$.

Определим количество разрядов для модульной части записи чисел, учитывая не только значения используемых операндов, но и ожидаемые результаты выполнения заданных операций. Исходя из абсолютного значения операндов разрядность представления модульной части n должна быть равна 8. Учитывая то, что в подлежащих реализации выражениях над числами выполняется только одна операция (или сложения, или вычитания), длину модульной части необходимо взять на один разряд больше, т.е. n = 9.

Избавляясь от операции вычитания, приводим заданные выражения к виду

C1= A+B, C2 = A+(-B), C3 = B+(-A), C4 = (-A) + (-B).

Таким образом, в подлежащих реализации выражениях в качестве операндов присутствуют следующие величины: A , -A, B, -B. Представим их в прямом и дополнительном кодах:

$[A]_{пр}$ = 0.000111001, $[A]_{дк}$ = 0.000111001,
$[-A]_{пр}$ = 1.000111001, $[-A]_{дк}$ = 1.111000111,
$[B]_{пр}$ = 1.011010010, $[B]_{дк}$ = 1.100101110,
$[-B]_{пр}$ = 0.011010010, $[-B]_{дк}$ = 0.011010010.

Используя сформированный дополнительный код, реализуем выражения для C1,C2, C3, C4.

C1: 0.0 0 0 1 1 1 0 0 1 - $[A]_{дк}$
+ 1.1 0 0 1 0 1 1 1 0 - $[B]_{дк}$
1.1 0 1 1 0 0 1 1 1 - $[C1]_{дк}$
1.0 1 0 0 1 1 0 0 1 - $[C1]_{пк}$

C2: 0.0 0 0 1 1 1 0 01 - $[A]_{дк}$
+ 0.0 1 1 0 1 0 0 1 0 - $[-B]_{дк}$
0. 1 0 0 0 0 1 0 1 1 - $[C2]_{дк}$ =$[C2]_{пк}$

C3: 1.1 1 1 0 0 0 1 1 1 - $[-A]_{дк}$
+ 1.1 0 0 1 0 1 1 1 0 - $[B]_{дк}$
11.0 1 1 1 1 0 1 0 1 - $[C3]_{дк}$
1.1 0 0 0 0 1 01 1 - $[C3]_{пк.}$.

При выполнении сложения в данном случае возникла единичка переполнения знакового поля. При работе с дополнительным кодом она игнорируется (в примере она перечеркнута).

C4: 1.1 1 1 0 0 0 1 1 1 - $[-A]_{дк}$
+ 0. 0 1 1 01 0 0 1 0 - $[-B]_{дк}$
1 0. 0 1 0 01 1 0 0 1 - $[C4]_{дк}$ =$[C4]_{пк.}$

В данном случае также возникло переполнение знакового поля, которое игнорируется.

### 4.4. Операции с двоичными числами в обратном коде

При сложении чисел, представленных в *обратном* коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимания на границу, разделяющую знаковое и модульные поля. Переполнение знакового поля, т.е. перенос, возник-

ший из крайнего левого разряда, должен быть учтен как +1 в младший разряд полученной суммы. В результате такого сложения будет получен *обратный* код суммы заданных операндов.

*Пример*

Найти значения для C1, C2, C3, C4, определяемых выражениями

C1 = A+B, C2 = A-B, C3 = B-A, C4 =-A-B,

если $A = 57_{10}$, $B = -210_{10}$. При выполнении операций использовать двоичный обратный код. Результат представить в прямом коде.

*Решение*

В данном примере используются те же выражения и те же операнды, что и в предыдущем примере, поэтому при его решение используются уже найденные ранее двоичные представления операндов и их прямые коды.

Обратные коды операндов имеют вид

$[A]_{ок} = 0.000111001$, $[-A]_{ок} = 1.111000110$,

$[B]_{ок} = 1.100101101$, $[-B]_{ок} = 0.011010010$.

Используя сформированный дополнительный код, реализуем выражения для C1, C2, C3, C4.

C1: 0.0 0 0 1 1 1 0 0 1 - $[A]_{ок}$

+ 1.1 0 0 1 0 1 1 0 1 - $[B]_{ок}$

1.1 0 1 1 0 0 1 1 0 - $[C1]_{ок}$

1.0 1 0 0 1 1 0 0 1 - $[C1]_{пк}$

C2: 0.0 0 0 1 1 1 0 01 - $[A]_{ок}$

+ 0.0 1 1 0 1 0 0 1 0 - $[-B]_{ок}$

0.1 0 0 0 0 1 0 1 1 - $[C2]_{ок} = [C2]_{пк}$

C3: 1.1 1 1 0 0 0 1 1 0 - $[-A]_{ок}$

+ 1.1 0 0 1 0 1 1 0 1 - $[B]_{ок}$

1 1.0 1 1 1 1 0 0 1 1

\+ 1

1. 0 1 1 1 1 0 1 0 0 - $[C3]_{ок.}$

1.1 0 0 0 0 1 0 1 1 - $[C3]_{пк}$

C4: 1.1 1 1 0 0 0 1 1 0 - $[-A]_{ок}$

+ 0. 0 1 1 01 0 0 1 0 - $[-B]_{ок}$

1 0. 0 1 0 01 1 0 0 0

\+ 1

0. 0 1 0 01 1 0 0 1 - $[C4]_{ок} = [C4]_{пк}$

В данном случае также возникло переполнение знакового разряда, которое должно быть учтено как +1 в младший разряд сформированной суммы.

### 4.5. Модифицированные коды

При расчете разрядности n модульного поля весьма трудно бывает учесть диапазон значений результатов, особенно когда последовательность операции, представленных в подлежащих реализации выражениях, достаточно сложны.

При несоответствии выбранной разрядности n диапазону изменения представляемых чисел при выполнении операции сложении чисел с одинаковыми знаками возможно появление ситуации переполнения, когда подлежащий представлению результат выходит за диапазон представления, определенный некорректно выбранной разрядностью n поля модуля.

Например, в случае сложения двух чисел, представленных в обратном коде:

$[D1]_{ок} = 1.00110$ и $[D2]_{ок} = 1.00110$

сумма этих чисел

F1 = D1+ D2

будет подсчитана следующим образом:

F1:

```
  1. 0 0 1 1 0
+1. 0 0 1 1 0
 10 .0 1 1 0 0
+            1
  0. 0 1 1 0 1
```

Пример, выполненный по всем формальным правилам, дал абсурдный результат, так как получена положительная сумма двух отрицательных операндов. Аналогичная ситуация может возникать и при использовании дополнительного кода.

Ситуацию переполнения можно обнаруживать по факту появления «абсурдного» результата, но для этого необходимо помнить то, что в суммировании принимают участие операнды с одинаковыми знаками и знак полученного при этом результата отличен от знака операндов.

Более просто ситуация переполнения определяется при применении *модифицированного* кода(обратного или дополнительного). Модифицированные коды отличаются от базовых кодов только тем, что поле знака операндов имеет два разряда, и эти разряды имеют одинаковые значения:

0 0 - для положительных чисел;

1 1 - для отрицательных чисел.

Если в результате сложения чисел в модифицированном коде полученный результат имеет в поле знака одинаковые значения в обоих разрядах (00 или 11), то переполнения нет, если же разряды знакового поля имеют неодинаковые значения (10 или 01), то имеет место переполнение. При этом, если в поле знака имеет ме-

сто значение 01 - результат положительный, а если 10, то полученный результат отрицательный (основным носителем знака числа является левый разряд знакового поля).

*Пример*

Найти значения выражений

C1= A+B, C2=A-B, C3= B-A, C4=-A-B,

используя *модифицированный* обратный код если

$[A]_{пк} = 0.1010011,$

$[B]_{пк} = 1.0111001.$

*Решение*

*Модифицированный обратный* код для всех операндов, используемых в приведенных выражениях, имеет вид

$[A]_{мпк} = 0\,0.1010011,\ [A]_{мок} = 0\,0.1010011,\ [-A]_{мок} = 1\,1.0101100,$

$[B]_{мпк} = 1\,1.0111001,\ [B]_{мок} = 1\,1.1000110,\ [-B]_{мок} = 0\,0.0111001.$

Выполним действия, указанные в приведенных выражениях:

C1:

$0\,0.1010011$ - $[A]_{мок}$

$+11.1000110$ - $[B]_{пок}$

$100.0011001$

$+ \quad 1$

$00.0011010$ - $[C1]_{мок}=[C1]_{мпк}$

C2:

$0\,0.1010011$ - $[A]_{мок}$

$+0\,0.0111001$ - $[-B]_{пок}$

$0\,1.0001100$

Результат положительный и имеет место переполнение.

C3:

$1\,1.0101100$ - $[-A]_{мок}$

$+1\,1.1000110$ - $[B]_{пок}$

$110.1110010$

$+1$

$10.1110011$

Результат отрицательный и имеет место переполнение.

С4:

1 1. 0101100 - $[-A]_{мок}$

+0 0. 0111001 - $[-B]_{пок}$

11. 1100101 - $[C4]_{мок}$

11. 0011010 - $[C1]_{мпк}$

При формировании С1 был получен положительный результат (без переполнения).

При формировании С4 был получен отрицательный результат (без переполнения).

Факт переполнения при формировании С3 и С2 устанавливается по наличию в разрядах знакового поля различных значений.

### 4.6. Арифметика с алгебраическими двоично-десятичными числами

Представление двоично-десятичных чисел в обратном и дополнительном кодах поясняется следующими примерами.

*Пример*

Найти *дополнительный и обратный двоично- десятичный коды* чисел

$A_{10}$ = 6917,

$A_{10}$ = -2847.

*Решение*

Двоично-десятичный код заданных чисел будет

$A_{2\text{-}10}$ = 0110 1001 0001 0111,

$B_{2\text{-}10}$ = - 0010 1000 0100 0111.

Непосредственно используя правило формирования прямого, дополнительного и обратного кодов, найдем:

$[A]_{пк} = [A]_{ок} = [A]_{дк}$ =0. 0110 1001 0001 0111, т.к. А - положительное число, поэтому записи его прямого, обратного и дополнительного кодов совпадают;

$[B_{пк}]$ = 1. 0010 1000 0100 0111,

модуль $[B]_{ок}$ : 1001 1001 1001 1001 $= (10^5 - 1)$

\- 0010 1000 0100 0111 $= B_{2\text{-}10}$

0111 0001 0101 0010 - результат вычитания записан с учетом коррекции.

Простое инвертирования модуля $[B]_{пк}$ дает значение $[B]_{ик}$ (инверсный код), равное

1101 0111 1011 1000,

которое отличается от требуемой формы модуля обратного кода тем, что каждая тетрада имеет избыточную шестерку. Обозначим такой код как «$[B]_{ок}$+6».

Таким образом, модуль обратного кода двоично-десятичного числа не может быть сформирован из модуля прямого кода за счет инвертирования, как это имеет место в случае двоичного кода. Однако инвертирование при формировании об-

ратного кода двоично-десятичного числа может быть использовано, если предварительно во все двоично-десятичные разряды (тетрады) ввести избыточную шестёрку. В этом случае для числа В имеет место:

0010 1000 0100 0111 - значение модуля записи $[B]_{пк}$

\+ 0110 0110 0110 0110 - двоичный код 6 для каждой тетрады

1000 1110 1010 1101 - значение модуля записи $[B]_{пк}$ с избыточными 6 во всех тетрадах, т.е. получено значение «$[B]_{пк}+6$»

0111 0001 0101 0010 - значение модуля записи $[B]_{ок}$ , полученое после инвертирования «$[B]_{пк}+6$».

При выполнении операций с алгебраическими двоично-десятичными числами на базе средств, реализующих двоичную арифметику, используется прием, согласно которому обработку чисел ведут по правилам двоичной арифметики с использованием ограниченного количества корректирующих операций, позволяющих получить искомый двоично-десятичный результат.

Данный подход иллюстрируется следующим примером.

*Пример*

Найти двоично-десятичные значения С1, С2, С3, С4, определяемых соответственно выражениями:

С1= А+В, С2 =А-В, С3 = В-А, С4 =-А-В, используя модифицированный обратный код,

если

А = 3 7 8 3 и В = -5 492.

Результат представить в прямом коде. При реализации операции сложения использовать модифицированный обратный код.

*Решение*

Прямой двоично - десятичный код заданных чисел имеет вид

$[A]_{пк}$ = 0. 0011 0111 1000 0011,

$[B]_{пк}$ = 1. 0101 0100 1001 0010.

Расчет выражений для С1, С2, С3, С4 осуществляется следующим образом.

$[C1]_{пк}$ : * *

00. 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 $[A]_{мок}$

\+ 11. 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0 1 $[B]_{ик} = [B]_{ок}+6$

11. 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 - сумма ($[A]_{мок}$ и $[B]_{ик}0$), сформированная по правилам суммирования двоичного

\+ 0 1 1 0 0 1 1 0 - коррекция в тедрадах, где был перенос

11. 1 1 1 0 1 0 0 0 1 1 1 1 0 1 1 0

11. 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 - $[C1]_{мпк}$

\- 1 7 0 9 - $C1_{10}$ (десятичный эквивалент С1).

При выполнении первого суммирования по правилам двоичной арифметики возникающий перенос (в тетрадах, отмеченных знаком *) унес лишнюю 6 (двоичный перенос унес из тетрады 16, а по правилам десятичного сложения он должен унести 10), что означает, что избыточная 6, введенная за счет использования инверсного кода числа вместо обратного, исчезает в тех тетрадах, где был перенос, и сохраняется в тетрадах, где перенос отсутствовал. Коррекция на +6 выполняется в тех тетрадах, где был перенос. Таким образом ,после коррекции во всех тетрадах будет иметь место избыточная 6, что позволяет перейти от такой записи к прямому коду результата за счет инвертирования записей всех тетрад модульной части.

| | |
|---|---|
| $[C4]_{пк}$ : * * | |
| 11. 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0 | - $[-A]_{ик}=[-A]_{ок}+6$ |
| + 00 . 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 | - $[-B]_{мок}$ |
| 1 00. 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 0 | - учет переноса при суммировании |
| + 1 | обратных кодов |
| 0 0. 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 | |
| + 1 0 1 0 1 0 1 0 | - коррекция в тедрадах, где не было переноса. |
| 00.0 0 01 0 1 1 1 0 0 0 0 1 0 0 1 | - $[C4]_{опк}=[C4]_{мпк}$ |
| 1 7 0 9 | - $C4_{10}$ (десятичный эквивалент). |

При выполнении первого суммирования по правилам двоичной арифметики возникающий перенос (в тетрадах, отмеченных знаком *) унес лишнюю 6 (двоичный перенос унес из тетрады 16, а по правилам десятичного сложения он должен унести 10), что означает, что избыточная 6, введенная за счет использования инверсного кода числа вместо обратного, исчезает в тех тетрадах, где был перенос, и сохраняется в тетрадах, где перенос отсутствовал.

Коррекция на 10 выполняется в тех тетрадах, где не было переноса (прибавлением 10, которая является дополнительным кодом -6, с блокировкой переноса из тетрады заменяется операция вычитания 6 из кода тетрады). Таким образом, после коррекции во всех тетрадах будет иметь место точное значение, а так как результат положительный, то эта запись соответствует искомому прямому коду С1.

Формирование С2 и С3, в выражениях которых в операции сложения принимают участие числа с одинаковыми знаками, осуществляется за счет сложения абсолютных значений операндов с последующим присвоением полученной суммы знака одного из операндов.

Таким образом, для определения С2 и С3 выполняются следующие действия:

$|C2| = |C3|$:

| | |
|---|---|
| 0. 0011 0111 1000 0011 | $-[\,\lvert A\rvert\,]_{пк}$ |
| + 0. 0101 0100 1001 0010. | $-[\,\lvert B\rvert\,]_{пк}$ |
| 0. 1000 1100 0001 0101 | |
| + 0110 0110 | |
| 0. 1001 0010 0111 0101 | $-[\,\lvert C2\rvert\,]_{пк} = [\,\lvert C3\rvert\,]_{пк}$ |

Таким образом, $[C2]_{пк}$ = 0. 1001 0110 0111 0101, $[C3]_{пк}$ = =1. 1001 0110 0111 0101.

Во всех примерах выполнения операций с использование прямого, дополнительного и обратного кода в качестве операндов рассматривались целые числа, т.е. точка, отделяющая целую от дробной части числа, предполагается расположенной после младшего разряда поля модуля записи числа. Для правильных дробей эта точка предполагается расположенной между полем знака и полем модуля. Как видно из приведенных примеров, ни при формировании заданного кода (прямого, дополнительного или обратного) на основании соответствующей записи числа, ни при выполнении операций с использованием этих кодов конкретное положение точки в записи числа не учитывалось. Поэтому формирование рассматриваемых кодов и их использование в случае дробных чисел ничем не отличаются от случая целых чисел.

### 4.7. Логические операции с двоичными кодами

Над двоичными кодами могут выполняться различные логические операции, среди которых особое место занимают:

- *логическое суммирование* (обозначения - ИЛИ, OR, «∨»);
- *логическое умножение* (обозначения - И, AND, «∧»);
- *логическое отрицание* (обозначения – НЕТ, NOT, «$\overline{x}$», т. е. штрих над отрицаемым кодом x);
- *суммирование по модулю* 2 (обозначается mod 2, « ⊕ »);
- *операции сдвига*.

Операция *логического суммирования* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором i-м разряде находится единица, если хотя бы в одном операнде в i-м разряде имеет место единица.

*Пример*

10001101 ^ 11110000 = 11111101.

Операция *логического умножения* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором i-м разряде находится единица, если оба операнда в этом i-м разряде имеются единицу, и ноль во всех других случаях.

*Пример*

10001101 ∧ 11110000 = 10000000

Операция *суммирования по модулю 2* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором i-м разряде находится единица, если два заданных операнда в i-м разряде имеется противоположные значения. Иногда эта операция называется «исключающее ИЛИ».

*Пример*

$10001101 \oplus 11110000 = 01111101$.

Операция логического отрицания выполняется над одним кодом и генерирует результирующий код той же разрядности, что и операнд, у которого в некотором i-м разряде находится значение, противоположное значению в i-м разряде отрицаемого кода.

Операции *сдвига* в свою очередь, подразделяются на:

– логические сдвиги, которые имеют разновидности - сдвиг вправо, сдвиг влево, циклический сдвиг вправо, циклический сдвиг влево;

– арифметический сдвиг вправо и в лево, выполнение которого зависит от знака и кода сдвигаемого числа.

*Логические сдвиги*

Сдвиг *влево* выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (освобождающийся самый правый т.е. самый младший, разряд заполняется 0, а «выталкиваемый» разряд пропадает). Например, код 11001110 после сдвига влево будет иметь вид 10011100.

Сдвиг *вправо* выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый, т.е. самый старший, разряд заполняется 0, «выталкиваемый» разряд пропадает). Например, код 11001110 после сдвига влево будет иметь вид 01100111.

*Циклический сдвиг влево* - выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (в освобождающийся самый правый, т.е. самый младший, разряд заносится значение старшего, т.е. самого левого разряда исходного кода). Например, код 11001110 после сдвига влево будет иметь вид 10011101.

*Циклический сдвиг вправо* - выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый т.е. самый старший, разряд заполняется значение в самом младшем разряде исходного кода). Например, код 11001110 после сдвига влево будет иметь вид 01100111.

*Арифметический сдвиг*

Арифметические сдвиги обеспечивают выполнения умножения (сдвиги влево) или операции деления (сдвиги вправо) двоичных кодов на два, точно так же, как сдвиги вправо и влево десятичного числа обеспечивают выполнение деления и умножение на 10.

Арифметические сдвиги влево двоичного прямого кода выполняются в зависимости от того, какое сдвигается число - положительное или отрицательное.

Если *сдвигается положительное* число, то сдвиг (вправо или влево) выполняется как соответствующий логический сдвиг (влево или вправо), с той лишь разницей, что предусматриваются средства определения факта переполнения при сдвиге влево, что реализуется и при всех других арифметических операциях. При любом сдвиге вправо предусматриваются средства для округления после завершения нужного количества сдвигов и средства обнаружения обнуления сдвигаемой величины после очередного сдвига.

Арифметические сдвиги влево положительных двоичных чисел выполняются независимо от используемого кода (прямого обратного, дополнительного). Его реализация иллюстрируются следующими примерами.

*Пример*

1.Найти результат арифметического сдвига влево на три разряда двоичного прямого кода числа $[A]_{пк}$ = 00. 00000101

*Решение*

Процесс выполнения заданного сдвига дает следующие промежуточные и конечное значения:

первый сдвиг: 00. 00000101→ 00. 00001010;
второй сдвиг: 00. 00001010→ 00. 00010100;
третий сдвиг: 00. 00010100→ 00. 00101000.

2. Найти результат арифметического сдвига влево на четыре разряда двоичного прямого кода числа $[A]_{пк}$= 00. 00101000.

*Решение*

Процесс заданного сдвига дает следующие промежуточные и конечное значения:

первый сдвиг: 00. 00100101→ 00. 01001010;
второй сдвиг: 00. 01001010→ 00. 10010100;
третий сдвиг: 00. 10100000→ 01. 01000000.

После третьего сдвига будет выработан сигнал переполнения, так как после очередного сдвига в разрядах знакового поля появятся разные значения. Таким образом, не считая процедуры определения переполнения, арифметический сдвиг влево выполняется точно так же, как и логический сдвиг влево.

3. Найти результат арифметического сдвига вправо на два разряда двоичного прямого кода числа $[A]_{пк}$= 00. 00000110.

*Решение*

Процесс заданного сдвига дает следующие промежуточные и конечное значение:

первый сдвиг: 00. 00000110→ 00. 00000011;
второй сдвиг: 00. 00000011→ 00. 00000001;

После выполнения заданного количества сдвигов выполняется округление на основании последнего «вытолкнутого» разряда; в данном случае последний «вытолкнутый» разряд равен 1, поэтому конечный результат выполнения заданного сдвига будет равен

00.00000010.

4. Найти результат арифметического сдвига вправо на четыре разряда двоичного прямого кода числа $[A]_{пк}$= 00. 00000110.

*Решение*

Процесс заданного сдвига дает следующие промежуточные и конечное значения:

первый сдвиг: 00. 00000110→ 00. 00000011;

второй сдвиг: 00. 00000011→ 00. 00000001;

третий сдвиг: 00. 00000001→ 00. 00000000,

после выполнения третьего сдвига будет выработан сигнал о получении нулевого результата. Оставшиеся сдвиги могут не выполняться.

*Арифметические сдвиги отрицательных двоичных чисел, представленных в прямом коде*

Арифметические сдвиги влево и вправо реализуются по-разному в зависимости как от знака числа, так и от используемого кода (прямого обратного, дополнительного).

При арифметическом сдвиге отрицательного двоичного числа, представленного в прямом коде, осуществляется соответствующий сдвиг только модульного поля записи числа.

Реализация этого типа сдвига иллюстрируется следующими примерами.

*Пример 1*

Выполнить *арифметический сдвиг* влево двоичного числа A = 11. 001010 (соответствует $10_{10}$), представленного в модифицированном прямом коде.

*Решение*

Заданный сдвиг, имеющий своей целью получение результата, в два раза превышающего по абсолютному значению значение исходного кода, дает в результате 11.010100 ($20_{10}$), которое получается за счет логического сдвига влево только модульной части исходного кода. Факт получения переполнения устанавливается по наличию единичного значения старшего разряда в сдвигаемом коде перед очередным сдвигом.

*Пример 2*

Выполнить арифметический сдвиг вправо двоичного числа A = 11. 01110 ($14_{10}$), представленного в модифицированном прямом коде.

*Решение*

Заданный сдвиг, имеющий своей целью получение кода, в два раза меньшего по абсолютному значению по отношению к значения исходного кода, дает в результате число 11.00111($7_{10}$), которое получается за счет логического сдвига влево только модульной части исходного кода.

При арифметическом сдвиге влево отрицательного двоичного числа, представленного в обратном коде, осуществляется циклический сдвиг исходного кода с контролем за переполнением, например, сдвиг влево отрицательного двоичного числа 11.1100110($25_{10}$), представленного в обратном коде, дает в результате 11.1001101($50_{10}$).

При арифметическом сдвиге вправо отрицательного двоичного числа, представленного в обратном коде, осуществляется сдвиг только модульной части записи числа с установкой единицы в освобождающийся разряд. При этом может осуществляется контроль за обнулением результата сдвиг (появление единичных значений во всех разрядах) и округление результата после выполнения заданного количества сдвигов.

*Пример3*

Выполнить сдвиг вправо на четыре разряда двоичного числа 11.1001101(десятичный эквивалент $50_{10}$), представленного в обратном коде.

Первый сдвиг дает 11.11001101($50_{10}$) → 11.11100110($25_{10}$).
Второй сдвиг дает 11.11100110($25_{10}$)→ 11.11110011($12_{10}$).
Третий сдвиг дает 11.11110011($12_{10}$)→ 11.11111001($6_{10}$).
Четвертый сдвиг дает 11.11111001($6_{10}$),.→ 11.11111100($3_{10}$).

При выполнении сдвига вправо нечетного числа результат получается с точностью до младшего разряда кода, причем ошибка отрицательная.

После выполнения последнего, четвертого сдвига выполняется округление, при котором, если последний «вытолкнутый» разряд имел значение 0, к результату последнего сдвига прибавляется -1. Данное округление можно выполнить за счет прибавления единицы к прямому коду, соответствующему результату последнего сдвига исходного обратного кода.

В рассмотренном примере корректировать на единицу результат четвертого сдвига не надо, так как «вытолкнутый» разряд при последнем (четвертом) сдвиге равен единице. В данном случае конечный результат сдвига заданного отрицательного числа, представленного в обратном коде, равен 11.11111100.

При арифметическом сдвиге влево отрицательного двоичного числа, представленного в дополнительном коде, осуществляется логический сдвиг влево модуля исходного кода (освобождающийся разряд заполняется нулем) с контролем за переполнением, например, сдвиг влево отрицательного двоичного числа 11.11001110($50_{10}$), представленного в дополнительном коде, дает в результате 11.10011100 ($100_{10}$).

При арифметическом сдвиге вправо отрицательного двоичного числа, представленного в дополнительном коде, осуществляется логический сдвиг вправо модуля записи числа с установкой единицы в освобождающийся разряд. При этом может осуществляется контроль за обнулением результата сдвига (появление единичных значений во всех разрядах).

*Пример*

Выполнить сдвиг вправо на четыре разряда двоичного числа 11.11001110 (десятичный эквивалент $50_{10}$), представленного в дополнительном коде.

*Решение*

Первый сдвиг дает 11.11001110→ 11.11100111($25_{10}$),
Второй сдвиг дает 11.11100111→ 11.11110011($13_{10}$),
Третий сдвиг дает 11.11110011→ 11.11111001($7_{10}$),
Четвертый сдвиг дает 11.11111001→ 11.11111100($4_{10}$).

При выполнении сдвига вправо нечетного целого числа результат получается с точностью до младшего разряда кода, причем ошибка положительная.

Арифметический сдвиг вправо может выполняться над отрицательными числами с переполнением (такие числа в модифицированном прямом, обратном или дополнительном коде имеют в знаковом поле 10). В этом случае после сдвига в знаковом поле будет 11, а в старшем разряде – 0, если число представлено в обратном или дополнительном коде, или 1, если число представлено в прямом коде.

*Пример1*

Выполнить сдвиг вправо на 2 разряда числа $[A]_{пк} = 10.\ 01000110$ ($A_{10} = -326$).

*Решение*

1-й сдвиг: 10. 01000110 →11.10100011 (-163);

2-й сдвиг: 11.10100011→11.01010001 (- $81_{10}$ и последний вытолкнутый разряд равен 1).

Окончательный результат с учетом округления имеем $[A/\ 2^2]_{пк} = 11.10010010$.

*Пример2*

Выполнить сдвиг вправо на 2 разряда числа $[A]_{ок} = 10.\ 10111001$ ($A_{10} = -326$).

*Решение*

1-й сдвиг: 10. 10111001 →11.01011100 (-163);

2-й сдвиг: 11.01011100 →11. 10101110 (-$81_{10}$).

*Пример3*

Выполнить сдвиг вправо на 2 разряда числа $[A]_{дк} = 10.\ 10111010$ ($A_{10} = -326$).

*Решение*

1-й сдвиг: 10. 10111010 →11.01011101 (-163);

2-й сдвиг: 11. 01011101 →11. 10101110 ( - 82 и последний вытолкнутый разряд равен 1).

Окончательный результат с учетом округления имеем $[A\ 2^2]_{ок}$= =11. 10101101.

## 5. Представление чисел с фиксированной точкой

Числовая информация представляется в машине в форме с фиксированной или с плавающей точкой. При представлении с фиксированной точкой положение последней в записи числа фиксировано.

Как правило, при использовании фиксированной точки числа представляются в виде целого числа или правильной дроби, форматы которых приведены на рис. 3.

| зн. | 1р. | 2р. | 3р. | 4р. | ... | (n-1) р. | пр. | «.» |
|---|---|---|---|---|---|---|---|---|

а

| зн. | «.» | 1р. | 2р. | 3р. | 4р. | ... | (n-1) р. | пр. |
|---|---|---|---|---|---|---|---|---|

б

Рис. 3:

а – формат целого числа; б - формат дробного числа

К заданному виду (целым числам или правильной дроби) исходные числа приводятся за счет введения масштабных коэффициентов.

Точка в записи числа не отображается, а так как она находится всегда в одном месте, то указание на её положение в записи числа отсутствует. При n-разрядном представлении модульной части форма с фиксированной точкой обеспечивает диапазон изменения абсолютного значения числа А, для которого выполняется неравенство

$2^{n.} > |A| > = 0.$

Одним из важнейших параметров представления чисел является ошибка представления. Ошибка представления может быть абсолютной (Δ) или относительной (δ). Для фиксированной точки максимальные значения этих ошибок определяются следующим образом.

В случае целых чисел:

$\Delta_{max} = 0.5$; $\delta_{max} = \Delta_{max} / A_{min} = 0.5$, где $A_{min}$ - минимальное, отличное от нуля, значение числа.

В случае дробных чисел:

$\Delta_{max} = 0.5 \cdot 2^{-n} = 2^{-(n+1)}$ ; $\delta_{max} = \Delta_{max} / A_{min} = 2^{-(n+1)} / 2^{-n} = 0.5$,

т.е. в худшем случае относительная ошибка при фиксированной точке может достигать сравнительно большого значения - 50%.

### 5.1. Арифметические операции над числами, представленными с фиксированной точкой

К числу основных арифметических операций, непосредственно реализуемых в ЭВМ, относятся операции сложения, умножения, деления. Остальные операции (например, такие, как возведение в степень, извлечение квадратного корня) реализуются программным способом.

Выполнение операций с числами, представленными с фиксированной точкой, рассмотрено в рамках материала по выполнению операций с алгебраическими числами (разд. 4).

Выполнение длинных операций, таких, как умножение и деление, реализуется в два этапа:

− на первом этапе формируется знак искомого результата,

− на втором этапе, используя абсолютные значения операндов, ищем результат (произведение или частное), которому затем присваивается предварительно определенный знак.

Операнды, как правило, представлены в прямом коде, и знак результата, не зависимо от того, частное это или произведение, ищется за счет сложения по модулю 2 знаковых разрядов операндов. В результате этого знак результата положителен, если операнды имеют одинаковые знаки, или отрицательный, если операнды имеют разные знаки.

Выполнение второго этапа, т.е. умножение положительных чисел достаточно подробно изложено в разд. 3.3.

### 5.2. Деление с фиксированной точкой

Реализация второго этапа деления, т. е. формирования частного двоичных положительных чисел в ЭВМ, представленных правильной дробью, может быть выполнена двумя методами:

- деление с восстановлением остатка;
- деление без восстановления остатка.

**Метод деления с восстановлением остатка**

Деление выполняется потактно.

На каждом такте определяется один разряд частного. В процессе деления определяются разряд целой части (этот разряд имеет единичное значение, если деление двух правильных дробей дает частное, большее или равное 1), и n разрядов дробной части частного, где n-разрядность поля модуля представления чисел. Для того чтобы точность ошибка результата не превышала половины младшего разряда модуля, ищется дополнительный (n +1)-й разряд дробной части частного, который используется только для округления результата. Таким образом, деление выполняется за (n +2) такта.

На каждом такте выполняются следующие действия:

- из остатка, полученного на предыдущем такте (на первом такте из делимого), вычитается делитель (выполняется пробное вычитание), тем самым формируется новый остаток;
- анализируется знак нового остатка и, если знак отрицательный, то осуществляется восстановление остатка, т.е. к полученному новому остатку прибавляется делитель (если знак положительный, то восстановление остатка не происходит);
- если после пробного вычитания был получен положительный результат, то в очередном разряде формируемого частного устанавливается единица;
- выполняется умножение на 2 (арифметический сдвиг влево) нового или восстановленного остатка.

На первом такте определяется разряд целой части искомого частного. Для правильной дроби этот разряд должен иметь нулевое значение, поэтому, если на первом такте будет установлено, что первый разряд, т.е. разряд целой части искомого частного, равен единице, то вырабатывается специальный сигнал о том, что искомое частное не является правильной дробью. После выполнения последнего (n +2)-го такта анализируется последний найденный разряд частного и, если он равен единице, то в n-й разряд частного прибавляется единица.

*Пример 1*

Определить частное C1 от деления числа A на B, где

$[A]_{пк} = 0.1011$,

$[B]_{пк} = 1.1101$.

При выполнении операций использовать дополнительный код.

*Решение*

Знак искомого частного C1{зн} = 1, так как сумма по модулю «2» знаковых разрядов операндов равна 1 (A{зн} = 0, а B{зн} = 1.

Определим абсолютное значение для С. В процессе поиска значений разрядов частного будут использованы числа $|A|$, $|B|$, $(-|B|)$, представление которых в модифицированном дополнительном коде имеет вид:

$[|A|]_{мдк} = 00.1011$

$[|B|]_{мдк} = 00.1101$

$[-|B|]_{мдк} = 11.0011.$

Процесс выполнения отдельных тактов представляется следующим образом, табл. 6.

Таблица 6

| Определяемый разряда частного | Выполняемые действия | Пояснение действий | Значение очередного разряда частного |
|---|---|---|---|
| 1 р. (разряд целой части) - 1-й такт | 00.1011<br>+11.0011<br>11.1110<br>+00.1101<br>00.1011<br>01.0110 | Вычитание из абсолютного значения делимого абсолютное значение делителя; остаток <0<br>Восстановленный остаток<br>Сдвинутый остаток | 0 |
| 2 р. (старший разряд модульной части)- 2-й такт | +11.0011<br>100.1001<br>00.1001<br>01.0010 | Остаток >0 (перенос игнорируется)<br><br>Сдвинутый остаток | 1 |
| 3 р. - 3-й такт | +11.0011<br>100.0101<br>00.1010 | <br>Остаток >0 (перенос игнорируется) сдвинутый остаток | 1 |
| 4 р. - 4-й такт | +11.0011<br>11.1101<br>+00.1101 | <br>Остаток <0<br>Восстановление остатка | 0 |
| | 00.1010<br>01.0100 | <br>Сдвинутый остаток | |
| 5 р. - 5-й такт | +11.0011<br>100.0111<br>00.1110 | <br>Остаток >0<br>Сдвинутый остаток | 1 |
| 6 р - 6-й такт | +11.0011<br>100.0001<br>00.0010 | <br>Остаток>0<br>Сдвинутый остаток | 1 |

Сформированное из последовательности найденных на отдельных тактах разрядов абсолютное значение частного будет равно

$|C| = 0.11011,$

после округления будет иметь место

$|C| = 0.1110.$

С учетом ранее полученного знака окончательный результат равен
$[C]_{пк} = 1.1110$.

*Метод деления без восстановления остатка*

Идея метода деления без восстановления остатка основана на следующем.

Действия на i-м и на (i +1)-м тактах зависят от знака получаемого остатка и представляются следующим образом:

$(O_{i-1} - D)\ 2 - D$ , если $(O_{i-1} - D) >= 0$;

$((O_{i-1} - D) + D)2 - D = (O_{i-1} - D) + 2D - D = (O_{i-1} - D) + D$, если $(O_{i-1} – D) < 0$,

где $O_{i-1}$ - остаток на (i-1)-м такте;

D - абсолютное значение делителя.

Отсюда следует, что в случае, если остаток на предыдущем такте был отрицательным, то можно не осуществлять операцию его восстановления, но на следующем такте нужно не вычитать, а прибавлять делитель.

Исходя из этого деление без восстановления остатка выполняется следующим образом.

1. Число тактов при рассматриваемом методе определяется точно так же, как и при делении с восстановлением остатка.
2. На каждом такте выполняются следующие действия:

– анализируется знак остатка (на первом такте анализируется знак делимого) и, если знак положительный, то из остатка вычитается делитель, в противном случае делитель прибавляется; таким образом формируется новый остаток;

– если новый остаток положительный, то в очередном разряде формируемого частного устанавливается единица;

– выполняется умножение на два нового остатка.

3. Разряд, определенный на первом такте, так же как и в предыдущем методе, является разрядом целой части и, если он ненулевой, то вырабатывается сигнал о нарушении формы представления чисел в виде правильной дроби.
4. После выполнения последнего (n + 2)-го такта выполняется округление.

*Пример 2*

Определить частное от деления числа A на B, где

$[A]_{пк} = 0.1011$,

$[B]_{пк} = 1.1101$.

При выполнении операций использовать дополнительный код.

*Решение*

Знак искомого частного C1{зн} = 1, так как сумма по модулю 2 знаковых разрядов операндов равна 1 (A{зн} = 0, а B{зн} = 1).

Определим абсолютное значение для C.

В процессе поиска значений разрядов частного будут использованы числа $|A|$, $|B|$, $(-|B|)$, представление которых в модифицированном дополнительном коде имеет вид: $[|A|]_{мдк} = 00.1011$, $[|B|]_{мдк} = 00.1101$, $[-|B|]_{мдк} = 11.0011$.

Процесс выполнения отдельных тактов представляется следующим образом, табл. 7.

Таблица 7

| Определяемый разряда частного | Выполняемые действия | Пояснение действий | Значение очередного разряда частного |
|---|---|---|---|
| р. (разряд целой части) -1-й такт | 00.1011<br>+11.0011<br>11.1110<br>11.1100 | Вычитание из абсолютного значения делимого абсолютное значение делителя остаток > 0 (перенос игнорируется)<br>сдвинутый остаток | 0 |
| 2 р. (старший разряд модуля) - 2-й такт | +00.1101<br>100.1001<br>01.0010 | Прибавление \|B\|, т.к. остаток <0<br>Остаток >0 (перенос игнорируется)<br>Сдвинутый остаток | 1 |
| 3 р. - 3-й такт | +11.0011<br>100.0101<br>00.1010 | Вычитание \|B\|, т.к. остаток >0<br>Остаток >0 (перенос игнорируется)<br>Сдвинутый остаток | 1 |
| 4 р.- 4-й такт | +11.0011<br>11.1101<br>11.1010 | Вычитание \|B\|, т.к. остаток >0<br>Новый остаток <0<br>Сдвинутый остаток | 0 |
| 5 р. - 5-й такт | +00.1101<br>100.0111<br>00.1110 | Прибавление \|B\|, т.к. остаток<0<br>Новый остаток >0<br>Сдвинутый остаток | 1 |
| 6 р. - 6-й такт | +11.0011<br>100.0001<br>00.0010 | Вычитание \|B\|, т.к. остаток >0<br>Новый остаток >0<br>Сдвинутый остаток | 1 |

Сформированное из последовательности найденных на отдельных тактах разрядов абсолютное значение частного будет равно

$|C| = 0.11011$,

после округления будет иметь место

$|C| = 0.1110$.

С учетом ранее полученного знака окончательный результат равен

$[C]_{пк} = 1.1110$.

## 6. Представление чисел с плавающей точкой

При представлении числа *с плавающей точкой* число в общем случае представляет собой смешанную дробь и имеет формат, приведенный на рис. 4.

| 1р. | 2р. | 3р. | ... | к р. | «.» | (к+1) р. | (к+2) р. | ... | (n+1) р. | n р. |
|---|---|---|---|---|---|---|---|---|---|---|

Рис. 4

Местоположение точки в записи числа может быть различным, а так как сама точка в записи числа не присутствует, то для однозначного задания числа необходима не только его запись, но и информация о том, где в записи числа располагается точка, отделяющая целую и дробную части.

Поэтому в случае с плавающей точкой число X представляется в виде двух частей:

*мантисса* ($x_м$), отображающая запись числа, представляется в виде правильной дроби с форматом фиксированной точки;

*порядок* ($x_п$), отображающий местоположения в этой записи положение точки, представляется в виде целого числа с форматом фиксированной точки.

Количественная оценка числа X определяется как

$X = q^{x_п} \cdot x_{м.}$ ,

где q - основание счисления.

Для двоичной системы счисления имеет место

$X = 2^{x_п} \cdot x_{м.}$

При s- разрядном представлении модуля записи мантиссы и k-разрядном представлении модуля записи порядка форма с плавающей точкой обеспечивает диапазон изменения абсолютного значения числа A, для которого выполняется неравенство

$2^{|x_п|_{max.}} \cdot |x|_{м\,max} = 2^P \cdot (1-2^{-S}) > = |X| > = 0$ ,

где $P = 2^k-1$.

В ЭВМ числа с плавающей точкой представляются в так называемой нормализованной форме, при которой в прямом коде мантисса нормализованного числа в старшем разряде модуля имеет ненулевое значение, а для двоичной системы счисления - нормализованная мантисса должна иметь в старшем разряде модуля прямого кода значение 1, т.е. для двоичной системы мантисса должна удовлетворять неравенству

$1> |x_m| >=0.5$.

Для плавающей точки максимальные значения абсолютной и относительной ошибок определяются следующим образом:

максимальная абсолютная ошибка представления чисел:

$\delta_{max} = \Delta_{max} / A_{min} = 2^{-(s+1)} \cdot 2^p /(x_{м\,мин} \cdot 2^p) = 2^{-(s+1)} \cdot 2^p /( 2^{-1} \cdot 2^p) =$

$= 2^{-(s+1)} /(2^{-1}) = 2^{-s}$.

Отсюда видно, что относительная ошибка при представлении чисел в форме с плавающей точкой существенно меньше, чем в случае с фиксированной точкой. Это, а также больший диапазон изменения представляемых чисел, является основным преимуществом представления чисел с плавающей точкой.

### 6.1. Арифметика с плавающей точкой

**Операция сложения**

Операция сложения чисел предполагает наличие одинаковых масштабов складываемых величин. Для случая представления чисел с плавающей точкой это предполагает наличие одинаковых порядков у операндов, подлежащих суммиро-

ванию. Поэтому при выполнении операции сложения чисел с плавающей точкой в общем случае должно быть реализовано три этапа:

- выравнивание порядков;
- сложение мантисс операндов, имеющих одинаковые порядки;
- определение нарушения нормализации и при необходимости её устранение.

*Пример*

Найти разность С1 чисел А и В, представленных с плавающей точкой, если А и В представлены в виде порядков, соответственно $[а_п]_{пк}$ и $[в_п]_{пк}$ и мантисс, соответственно $[а_м]_{пк}$ и $[в_м]_{пк}$,

где $[а_п]_{пк}$ = 1.001, $[а_м]_{пк}$ = 1.11001,

$[в_п]_{пк}$ = 0.001, $[в_м]_{пкр}$ = 0.11100.

При выполнении операций использовать дополнительный модифицированный код.

*Решение*

Начнем с выравнивания порядков.

Для этого из порядка первого числа вычитается порядок второго числа:

1.111 $[а_п]_{дк}$

+ 1.111 $[-в_п]_{дк}$

1.110 - разность порядков в дополнительном коде,

1.010 - разность порядков в прямом коде.

Так как знак разности порядков отрицательный, то в качестве общего порядка, а следовательно, и предварительного значения порядка искомого результата $С1_п`$ берется порядок второго числа ($в_п$). Для того чтобы взять в качестве порядка первого числа порядок второго числа, т.е. увеличив его порядок на 2, необходимо мантиссу этого меньшего числа умножить на $2^{-2}$, т.е. выполнить её арифметический сдвиг на два разряда вправо.

Таким образом, будем иметь после выравнивания следующую форму представления операндов:

$[а_м`]_{пк}$ = 1.00110,

$[в_м`]_{пк}$ = 0.11 1 00.

После выравнивания порядков можно определить предварительное значение мантиссы С1` как

С1` = $[а_м]_{пк}$` - $[в_м]_{пк}$`.

11.1 1 01 0 $-[а_м`]_{мдк}$

+ 1 1.0 01 0 0 - $[-в_м`]_{мдк}$

10.1 1 1 1 0 - $[С1']_{мдк}$.

10.0 0 0 1 0 - $[С1']_{мпк,}$.

Из записи $[С1`]_{дк}$, полученной после вычитания мантисс операндов с выравненными порядками, видно, что нормализация представления результата нарушена. Поэтому для данного примера необходимо выполнить этап устранения нарушения нормализации.

В данном случае нарушение нормализации слева от точки, так как получено $[С1`]_{пк}$ с ненулевой целой частью (неодинаковые разряды в поле знака использо-

ванного модифицированного дополнительного кода). Для того чтобы привести полученную предварительную мантиссу к нормализованной форме, достаточно её разделить на 2, то есть выполнить её арифметический сдвиг вправо. В результате будем иметь окончательное значение мантиссы:

$C1 = C1` \cdot 2^{-1} = 10.00010 \cdot 2^{-1} = 11.10001.$

Деление мантиссы C1` на 2 сопровождается изменением ранее найденного предварительного значения порядка результата $C1_п`$ на +1.

$$\begin{array}{ll} 0\ 0.001 & [c1_п]_{мпк}` \\ +\ 00.001 & +1 \\ \hline 00.010 & [c1_п]_{мпк} \end{array}$$

После устранения нарушения нормализации окончательный результат будет иметь вид

$C1 \rightarrow \{ [c1_п]_{пк} = 00.010, \quad [c1_м]_{пк} = 11.10001 \}.$

**Операция умножения**

С точки зрения представления чисел с плавающей точкой поиск произведения

$C2 = A \cdot B$

сводится к поиску $C2_п$ и $C2_м$, соответственно порядку и мантиссе произведения на основании порядка $a_п$ и мантиссы $a_м$ множимого и порядка $в_п$ и мантиссы $в_м$ множителя. Учитывая общую запись чисел с плавающей точкой, произведение двух операндов представляется в виде

$$C2 = A \cdot B = 2^{a_п} \cdot a_м \cdot 2^{в_п} \cdot в_м = 2^{a_п + в_п} \cdot a_м \cdot в_м = 2^{c2_п} \cdot c2_м,$$

откуда вытекает, что порядок произведения определяется как сумма порядков сомножителей, а мантисса произведения - как произведение мантисс сомножителей. Однако, учитывая то, что при умножении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значения порядка и мантиссы искомого произведения и окончательное значение произведения будет найдено только после устранения нарушения нормализации.

Таким образом, имеем:

$C2_п` = a_п + в_п;$

$C2_м` = a_м \cdot в_м.$

Отсюда последовательность действий, обеспечивающих получение произведение двух чисел, заключается в следующем:

– определяется знак произведения как сумма по модулю два знаковых разрядов мантисс сомножителей;

– определяется предварительное значение порядка произведения посредством суммирования порядков сомножителей;

– определяется предварительное значение мантиссы произведения как произведения мантисс операндов;

– устраняется нарушение нормализации мантиссы произведения (если нарушение имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого произведения.

При формировании мантиссы произведения нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации - нарушение нормализации справа от точки с появлением нуля только в старшем разряде мантиссы.

*Пример*

Найти произведение С2 числа А и В, представленных с плавающей точкой, если А и В представлены в виде порядков, соответственно $[a_п]_{пк}$ и $[в_п]_{пк}$ и мантисс, соответственно $[a_м]_{пк}$ и $[в_м]_{пк}$,

где $[a_п]_{пк}$ = 1.010, $[a_м]_{пк}$ = 1.1010,

$[в_п]_{пк}$ = 0.001, $[в_м]_{пк}$ = 0.1001.

При выполнении операций использовать *обратный код*. При умножении мантисс использовать метод умножения, начиная *со старшего разряда* множителя со сдвигом промежуточного результата.

*Решение*

Знак искомого произведения, представляемого знаком его мантиссы, отрицательный, так как знаки мантисс сомножителей неодинаковые.

Предварительное значение порядка произведения определяется следующим образом:

$С2_п$· = $a_п$ +$в_п$ :

11. 101 $[a_п]_{мок}$

+ 00.001 $[в_п]_{мок}$

11.110 $[С2_п`]_{мок}$

11.001 $[С2_п`]_{мпк}$, т.е. $[С2_п`]_{пк}$ = 1.001.

Абсолютное значение предварительного значения мантиссы произведения определяется следующим образом:

$[С2_м]$ · :

0.1010 - $|a_м|$

Ч 0.1001 - $|в_м|$

.0000 - начальное значение промежуточного произведения

+ 1010 - первый младший разряд множителя равен единице

1010 - промежуточное произведение с учетом первого разряда

01010 - сдвинутое промежуточное произведение

001010 - второй разряд множителя равен нулю, поэтому выполняется только сдвиг

0001010 - третий разряд равен нулю, поэтому выполняется только сдвиг.

+ 1010 - четвертый разряд равен единице

1011 010 - промежуточное произведение с учетом старшего разряда

01011 010 -сдвинутое промежуточное произведение.

Таким образом,

$[С2_м`]_{пк}$ = 0. 01011010,

с учетом округления имеем

$[C2_м] = 0.1\ 011.$

Мантисса произведения ненормализованная, поэтому необходимо сдвинуть мантиссу влево на один разряд, а предварительное значение порядка произведения уменьшить на единицу. После нормализации окончательное значение мантиссы и порядка произведения имеем с учетом ранее полученного знака:

$[C2_м]_п = 1.1011.$

$[C2_п]_п = 1.010.$

**Операция деления**

С точки зрения формирования частного представления чисел с плавающей точкой поиск частного С3 =А/В сводится к поиску $C3_п$ и $C3_м$, соответственно порядку и мантиссы частного на основании порядка $а_п$ и мантиссы $а_м$ делимого и порядка $в_п$ и мантиссы $в_м$ делителя. Учитывая общую запись чисел с плавающей точкой, частное двух операндов представляется в виде

$C2 = A/B = 2^{а\,п} \cdot а_м / (2^{в\,п} \cdot в_м) = 2^{ап\text{-}вп} \cdot (а_м / в_м) = 2^{сп} \cdot с_м.$

Отсюда следует, что порядок частного определяется как разность порядка делимого и делителя, а мантисса - как частное от деления мантиссы делимого на мантиссу делителя. Однако, учитывая то, что при делении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значения порядка и мантиссы искомого частного. Окончательные значения порядка и мантиссы частного будут определены после устранения нарушения нормализации в предварительном результате.

При формировании мантиссы частного нормализованных чисел с плавающей точкой возможно только один вид нарушения нормализации - нарушение нормализации слева от точки.

*Пример*

Найти частное С3 от деления чисел А на В, представленных с плавающей точкой, если А и В представлены в виде порядков, соответственно $[а_п]_{пк}$ и $[в_п]_{пк}$ и мантисс, соответственно $[а_м]_{пк}$ и $[в_м]_{пк}$,

где $[а_п]_{пк} = 1.010$, $[а_м]_{пк} = 1.1010$,

$[в_п]_{пк} = 0.001$, $[в_м]_{пк} = 0.1001$.

При выполнении операций использовать обратный код. При делении мантисс использовать метод деления без восстановления остатка. При вычитании порядков и формирования мантиссы частного использовать модифицированный обратный код.

*Решение*

Знак искомого частного, представляемого знаком его мантиссы, отрицательный, так как знаки мантисс сомножителей не одинаковые.

Предварительное значение порядка $[C3_п^*]_{ок}$ частного определяется следующим образом:

$C3_п` = а_п - в_п$ :

11.101 $[а_п]_{мок}$

<u>+ 11.110</u> $[в_п]_{мок}$

111.011
+ 1
11.100 $[СЗ_{п}`]_{мок}$, т.е. $[СЗ_{п}`]_{пк} = 1.011$.

Абсолютное значение предварительного значения мантиссы частного ищется за счет выполнения шести тактов деления следующим образом:

00.1010 - $[|a_{м}|]_{ок}$,
+11.0110 - $[-|в_{м}|]_{ок}$,
100.0000
+1 - учет переноса (переполнения знакового поля) при сложении в обратном коде, . .
00.0001 - положительный остаток первого такта,
00.0010 - сдвинутый остаток,
+11.0110 - $[-|в_{м}|]_{мок}$,
11.1000 - отрицательный остаток второго такта,
11.0001 - остаток после арифметического сдвига влево,
+ 0 0.1001 - $[|в_{м}|]_{мок}$,
11.1010 - отрицательный остаток третьего такта,
11.0101 - остаток после арифметического сдвига влево,
+ 0 0.1001 - $[|в_{м}|]_{мок}$,
11.1110 - отрицательный остаток четвертого такта,
11.1101 - остаток после арифметического сдвига влево,
+ 0 0.1001 - $[|в_{м}|]_{мок}$,
100.0110
+ 1
00.0111 - положительный остаток пятого такта
00.1110 - остаток после арифметического сдвига влево,
+11.0110 - $[-|в_{м}|]_{мок}$,
100.0100
+ 1
00.0101 - положительный остаток шестого такта,
00.1010 - остаток после арифметического сдвига влево.

Таким образом, учитывая знаки остатков, полученных на шести тактах, абсолютное предварительное значение мантиссы искомого частного равно:

$[|СЗ_{м}`|]_{п} = 1.00011$,

с учетом округления:

$[|СЗ_{м}`|]_{п} = 1.0010$.

Мантисса частного ненормализованная (нарушение нормализации слева от точки), поэтому необходимо сдвинуть мантиссу вправо на один разряд, а предварительное значение порядка частного увеличить на единицу. После нормализации окончательное значение мантиссы и порядка частного равны

$[СЗ_{м}]_{п} = 0.1001$,

$[СЗ_{п}]_{п} = 1.010$.

## 6.2. Представление данных в ЭВМ

Как правило, в качестве элементарной единицы информации для представления данных в машине используется байт, который обычно представляет восемь двоичных бит.

Можно выделить два основных вида данных:

– символьные данные;

– числовые данные.

Элементы данных, как правило, представляются в виде последовательности байт переменной длины, где на представление каждого символа отводится один *байт* (рис. 1.5,а). Исключение составляют десятичные числа, для которых может использоваться *упакованная форма*, при которой в одном байте располагается по две цифры, т.е. одна десятичная цифра в двоично-десятичной системе занимает четыре бита, т.е. тетраду (рис. 5,б).

| 5 | 4 | 8 | 6 |
|---|---|---|---|
| байт | байт | байт | байт |

а

| 5 | 4 | 8 | 6 |
|---|---|---|---|
| тетрада | тетрада | тетрада | тетрада |
| б а й т | | б а й т | |

Рис. 5:

а – посимвольная запись десятичного числа 5486;

б – упакованная запись десятичного числа 5486

Для представления двоичного числа обычно используется ограниченный набор форматов, например, один, два, четыре байта.

Пример представления чисел с плавающей точкой в 2-байтном формате приведен на рис. 6.

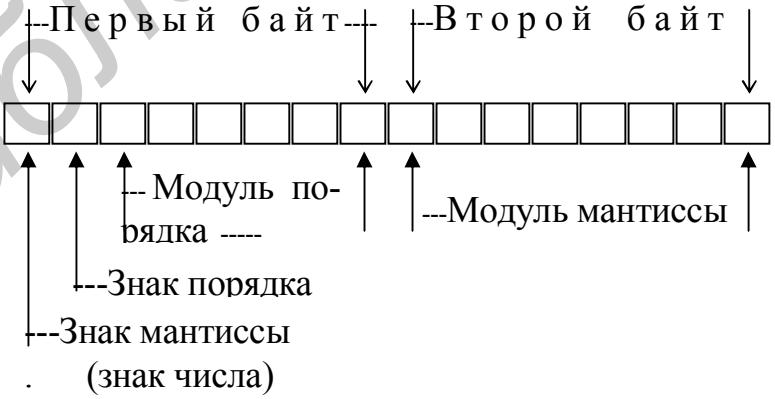



Рис. 6

При использовании 4- байтного формата вводимые дополнительные два байта, как правило, используются для расширения мантиссы представляемого числа.

Св. план 2004, поз.51

Учебное издание

**Пешков Анатолий Тимофеевич**

# Организация и функционирование ЭВМ

Методическое пособие для студентов специальности
«Программное обеспечение информационных технологий»
дневной формы обучения
В 3 частях
Часть 1
Арифметические основы ЭВМ

Редактор Е.Н. Батурчик
Компьютерная верстка М.В. Шишло

| | | |
|---|---|---|
| Подписана в печать 05. 04. 2004 | Формат 60x84 1/16. | Бумага офсетная. |
| Гарнитура «Таймс». | Печать ризографическая. | Усл.печ. л. |
| Уч.- изд. л. 3,0. | Тираж 100 экз. | Заказ 559. |

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП № 156 от 30. 12. 2002.
Лицензия ЛВ № 509 от 03. 08. 2001.
220013, Минск, П. Бровки, 6