**Тема 5. Представление чисел в компьютере**

## Как представляются в компьютере целые числа?

## Как компьютер выполняет арифметические действия над

## целыми числами?

3. Как представляются в компьютере вещественные числа?

## 4. Как компьютер выполняет арифметические действия над нормализованными числами?

## Как представляются в компьютере целые числа?

Целые числа могут представляться в компьютере со знаком или без знака.

**Целые числа без знака**

Обычно занимают в памяти компьютера один или два байта. В однобайтовом формате принимают значения от 000000002​ до 111111112​. В двубайтовом формате — от 00000000 000000002​ до 11111111 111111112​.

**Диапазоны значений целых чисел без знака**



**Примеры:**

а) число 7210 = 10010002 в **однобайтовом** формате:

http://book.kbsu.ru/theory/chapter4/0048.gif

б) это же число в **двубайтовом** формате:

http://book.kbsu.ru/theory/chapter4/0049.gif

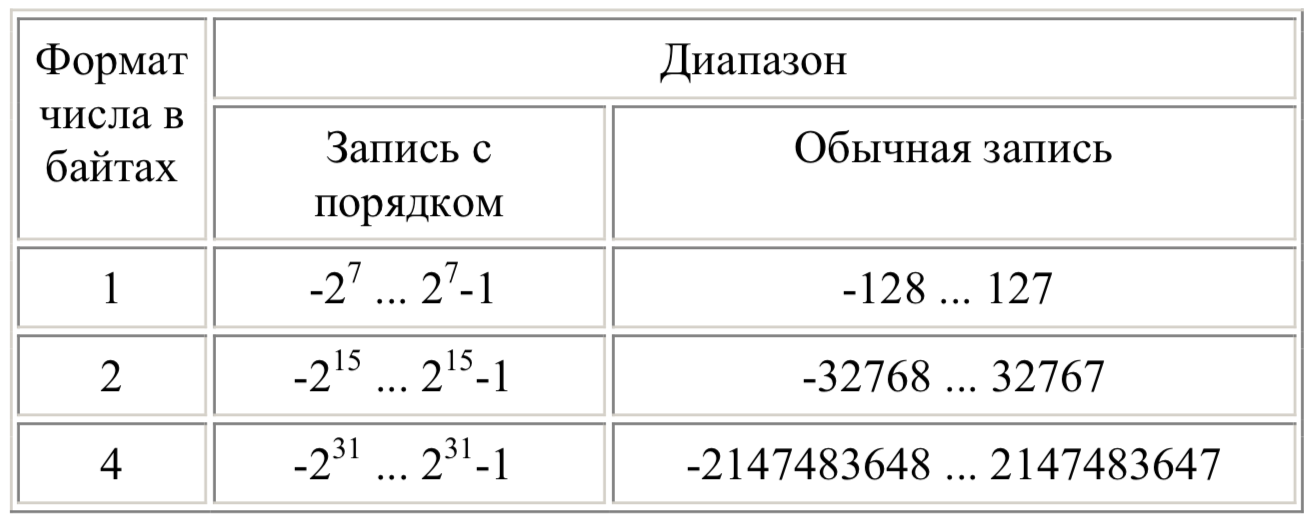
в) число 65535 в **двубайтовом** формате:

http://book.kbsu.ru/theory/chapter4/0050.gif  
 

**Целые числа со знаком**

Обычно занимают в памяти компьютера один, два или четыре байта, при этом самый левый (старший) разряд содержит информацию о знаке числа.

**Диапазоны значений целых чисел со знаком**



Рассмотрим особенности записи целых чисел со знаком на примере **однобайтового формата**, при котором для знака отводится один разряд, а для цифр абсолютной величины – семь разрядов.

В компьютерной технике применяются три формы записи (кодирования) целых чисел со знаком:

* **прямой** код,
* **обратный** код,
* **дополнительный** код.

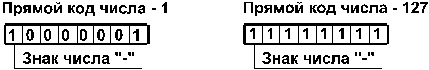
Последние две формы применяются особенно широко, так как позволяют упростить конструкцию арифметико-логического устройства компьютера путем замены разнообразных арифметических операций операцией cложения.

**Положительные числа** в прямом, обратном и дополнительном кодах изображаются одинаково  -  двоичными кодами с цифрой 0 в знаковом разряде. Например:

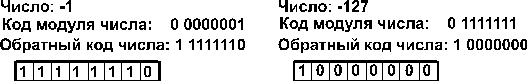


**Отрицательные числа** в прямом, обратном и дополнительном кодах имеют разное изображение.

1. **Прямой код**. В знаковый разряд помещается цифра 1, а в разряды цифровой части числа — двоичный код его абсолютной величины. Например:



2. **Обратный код**. Получается инвертированием всех цифр двоичного кода абсолютной величины числа, включая разряд знака: нули заменяются единицами, а единицы — нулями. Например:



3. **Дополнительный код**. Получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду. Например:

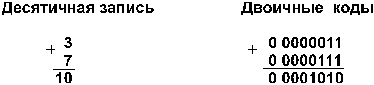
http://book.kbsu.ru/theory/chapter4/0054.gif

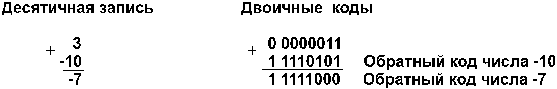
Обычно **отрицательные десятичные числа при вводе в машину автоматически преобразуются в обратный или дополнительный двоичный код** и в таком виде хранятся, перемещаются и участвуют в операциях. При выводе таких чисел из машины происходит **обратное преобразование в отрицательные десятичные числа.**

## **2. Как компьютер выполняет арифметические действия над целыми числами?**

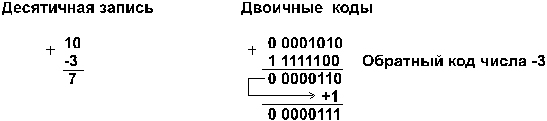
## Сложение и вычитание

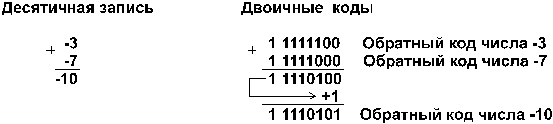
В большинстве компьютеров операция вычитания не используется. Вместо нее производится сложение обратных или дополнительных кодов уменьшаемого и вычитаемого. Это позволяет существенно упростить конструкцию АЛУ.

**1. А и В положительные.** При суммировании складываются все разряды, включая разряд знака. Так как знаковые разряды положительных слагаемых равны нулю, разряд знака суммы тоже равен нулю. Например:  
   
  
Получен правильный результат.

**2. А положительное, B отрицательное и по абсолютной величине больше, чем А.** Например:  
   
  
Получен правильный результат в обратном коде. При переводе в прямой код биты цифровой части результата инвертируются: 1 0000111 = -710.

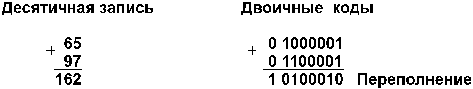
**3. А положительное, B отрицательное и по абсолютной величине меньше, чем А.** Например:

  
   
Компьютер исправляет полученный первоначально неправильный результат (6 вместо 7) **переносом единицы** из знакового разряда в младший разряд суммы.

**4. А и В отрицательные.** Например:  
   


Полученный первоначально неправильный результат (обратный код числа -1110 вместо обратного кода числа -1010) компьютер исправляет переносом единицы из знакового разряда в младший разряд суммы. При переводе результата в прямой код биты цифровой части числа инвертируются: 1 0001010 = -1010.

При сложении может возникнуть ситуация, когда старшие разряды результата операции не помещаются в отведенной для него области памяти. Такая ситуация называется **переполнением разрядной сетки формата числа.** Для обнаружения переполнения и оповещения о возникшей ошибке в компьютере используются специальные средства. Ниже приведены два возможных случая переполнения.

**5. А и В положительные, сумма А+В больше, либо равна 2n-1,** где n — количество разрядов формата чисел (для однобайтового формата n=8, 2n-1 = 27 = 128). Например:  
   


Семи разрядов цифровой части числового формата **недостаточно** для размещения восьмиразрядной суммы (16210 = 101000102), поэтому **старший разряд суммы оказывается в знаковом разряде.** Это вызывает **несовпадение знака суммы и знаков слагаемых**, что **является свидетельством переполнения разрядной сетки**.

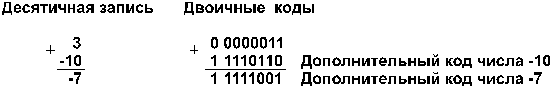
**6. А и В отрицательные, сумма абсолютных величин А и В больше, либо равна 2n-1.** Например:

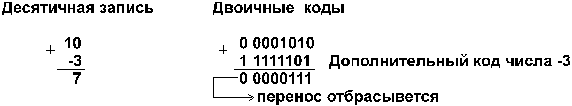


Здесь **знак суммы** тоже **не совпадает со знаками слагаемых**, что свидетельствует о **переполнении разрядной сетки**.

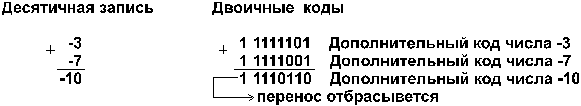
**Сложение дополнительных кодов.** Здесь также имеют место рассмотренные выше шесть случаев:

**1. А и В положительные.** Здесь нет отличий от случая 1, рассмотренного для обратного кода.

**2. А положительное, B отрицательное и по абсолютной величине больше, чем А.** Например:  
   
  
   
Получен правильный результат в дополнительном коде. При переводе в прямой код биты цифровой части результата инвертируются и к младшему разряду прибавляется единица: 1 0000110 + 1 = 1 0000111 = -710.

**3. А положительное, B отрицательное и по абсолютной величине меньше, чем А.** Например:  
   
  
Получен правильный результат. Единицу переноса из знакового разряда компьютер отбрасывает.

**4. А и В отрицательные.**

Например:  
   
  
Получен правильный результат в дополнительном коде. **Единицу переноса** из знакового разряда компьютер **отбрасывает**.

**Случаи переполнения** для дополнительных кодов рассматриваются по аналогии со случаями 5 и 6 для обратных кодов.

**Сравнение рассмотренных форм кодирования целых чисел со знаком показывает:**

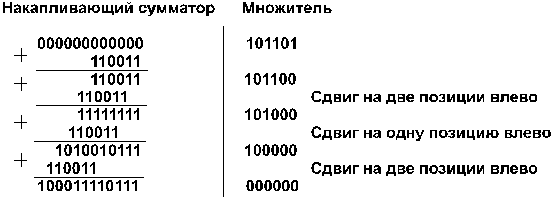
* **на преобразование отрицательного числа в обратный код компьютер затрачивает меньше времени, чем на преобразование в дополнительный код,** так как последнее состоит из двух шагов — образования обратного кода и прибавления единицы к его младшему разряду;
* **время выполнения сложения для дополнительных кодов чисел меньше, чем для их обратных кодов,** потому что в таком сложении нет переноса единицы из знакового разряда в младший разряд результата.

**Умножение и деление**

Во многих компьютерах **умножение** производится как последовательность сложений и сдвигов. Для этого в АЛУ имеется **регистр**, называемый **накапливающим сумматором**, который до начала выполнения операции **содержит число ноль**. В процессе выполнения операции в нем поочередно размещаются **множимое** и **результаты промежуточных сложений**, а по завершении операции — **окончательный результат**.

Другой регистр АЛУ, участвующий в выполнении этой операции, **вначале содержит множитель**. Затем по мере выполнения сложений содержащееся в нем **число уменьшается**, **пока не достигнет нулевого значения.**

Для иллюстрации умножим 1100112 на 1011012.



**Деление** для компьютера является трудной операцией. Обычно оно реализуется путем многократного прибавления к делимому дополнительного кода делителя.

**3. Как представляются в компьютере вещественные числа?**

**Форматы хранения вещественных чисел**

Вещественные числа в математических вычислениях не имеют ограничений на диапазон и точность представления чисел. Однако в компьютерах числа хранятся в регистрах и ячейках памяти с ограниченным количеством разрядов. Поэтому *точность представления вещественных чисел*, представимых в машине, *является конечной, а диапазон ограничен*.

При написании вещественных чисел в программах вместо привычной запятой принято ставить точку. Любое вещественное число можно представить в форме записи чисел с порядком основания системы счисления.

**Пример 4.4.** Десятичное число 1.756 в форме записи чисел с порядком основания системы счисления можно представить так:

1,756 · 100 = 0,1756 · 101 = 0,01756 · 102 = ...

или так:

17,56 · 10-1 = 175,6 · 10-2 = 1756,0 · 10-3 = ... .

Представлением числа с плавающей точкой называется представление числа ***N*** в системе счисления с основанием ***q*** в виде:

***N = m·q****p*,

где ***m*** - множитель, содержащий все цифры числа (мантисса), ***p*** - целое число, называемое порядком.

Если плавающая точка расположена в мантиссе перед первой значащей цифрой, то при фиксированном количестве разрядов, отведенных под мантиссу, обеспечивается запись максимального количества значащих цифр числа, т. е. максимальная точность представления числа в машине.

Если в мантиссе первая цифра после точки (запятой) отлична от нуля, то такое число называется нормализованным.

Мантиссу и порядок *q*-ичного числа принято записывать в системе с основанием *q*, а само основание — в десятичной системе.

**Пример 4.5.** Приведем примеры нормализованного представления числа в десятичной системе:

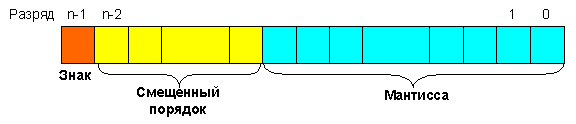
2178,01 = 0,217801 · 104

0,0045 = 0,45 · 10-2

Примеры в двоичной системе:

10110,01 = 0,1011001 · 2101 (порядок 1012 = 510)

Современными компьютерами поддерживаются несколько международных стандартных форматов хранения вещественных чисел с плавающей точкой, различающихся по точности, но все они имеют одинаковую структуру. Вещественное число хранится в трех частях: знак мантиссы, смещенный порядок и мантисса:



Смещенный порядок *n*-разрядного нормализованного числа вычисляется следующим образом: если для задания порядка выделено *k* разрядов, то к истинному значению порядка, представленного в дополнительном коде, прибавляют смещение, равное (2*k*-1 - 1).

Таким образом, порядок, принимающий значения в диапазоне от -128 до +127, преобразуется в смещенный порядок в диапазоне от 0 до 255. Смещенный порядок хранится в виде беззнакового числа, что упрощает операции сравнения, сложения и вычитания порядков, а также упрощает операцию сравнения самих нормализованных чисел.

Количество разрядов, отводимых под порядок, влияет на диапазон от наименьшего отличного от нуля числа до наибольшего числа, представимого в машине при заданном формате. Очевидно, что чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа. В связи с тем, что у нормализованных вещественных чисел старший бит мантиссы всегда равен 1, этот старший бит не хранится в памяти.

Любое двоичное целое число, содержащее не более *m* разрядов, может быть без искажений преобразовано в вещественный формат.

Таблица 4.3. **Стандартные форматы представления вещественных чисел**

| **Формат** | **Что хранится** | **Кол-во битов, отводимых под смещенный порядок** | **Кол-во битов, отводимых под мантиссу** |
| --- | --- | --- | --- |
| Одинарный | 32-разрядное нормализованное число со знаком | 8 | 23 |
| Двойной | 64-разрядное нормализованное число со знаком | 11 | 52 |
| Расширенный | 80-разрядное число со знаком (возможно ненормализованные) | 15 | 64 |

**Пример 4.6.** Представление нормализованных чисел в одинарном формате.

Проиллюстрируем, как будет храниться число 37,1610. При переводе в двоичное число не получается точного перевода: 100101,(00101000111101011100) - дробная часть, заключенная в скобках, повторяется в периоде.

Переводим число в нормализованный вид:

0,100101(00101000111101011100) · 2110

Представим вещественное число в 32-разрядном формате:

1. Знак числа «+», поэтому в знаковый разряд (31) заносим 0.
2. Для задания порядка выделено 8 разрядов, к истинному значению порядка, представленного в дополнительном коде, прибавляем смещение (27 - 1) = 127. Так как порядок положительный, то прямой код порядка совпадает с дополнительным, вычислим смещенный порядок: 00000110 + 01111111 = 10000101. Заносим полученный смещенный порядок.
3. Заносим мантиссу, при этом старший разряд мантиссы убираем (он всегда равен 1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Знак | Смещенный порядок | | | | | | | | Мантисса | | | | | | | | | | | | | | | | | | | | | | |

В данном примере мы смогли перенести не все разряда, они были утеряны с потерей точности представления числа.

## **4. Как компьютер выполняет арифметические действия над нормализованными числами?**

К началу выполнения арифметического действия операнды операции помещаются в соответствующие регистры АЛУ.

### Сложение и вычитание

При сложении и вычитании сначала производится подготовительная операция, называемая **выравниванием порядков**.

|  |
| --- |
| **В процессе выравнивания порядков мантисса числа с меньшим порядком сдвигается в своем регистре вправо на количество разрядов, равное разности порядков операндов. После каждого сдвига порядок увеличивается на единицу.** |

В результате выравнивания порядков одноименные разряды чисел оказываются расположенными в соответствующих разрядах обоих регистров, после чего **мантиссы складываются или вычитаются**. В случае необходимости полученный результат нормализуется путем сдвига мантиссы результата влево. После каждого сдвига влево порядок результата уменьшается на единицу.

**Пример 1.** Сложить двоичные нормализованные числа 0.10111**.**2-1 и 0.11011**.**210. Разность порядков слагаемых здесь равна трем, поэтому перед сложением мантисса первого числа сдвигается на три разряда вправо:

http://book.kbsu.ru/theory/chapter4/0068.gif

**Пример 2.** Выполнить вычитание двоичных нормализованных чисел 0.10101**.**210 и 0.11101**.**21. Разность порядков уменьшаемого и вычитаемого здесь равна единице, поэтому перед вычитанием мантисса второго числа сдвигается на один разряд вправо:

http://book.kbsu.ru/theory/chapter4/0069.gif

Результат получился **не нормализованным**, поэтому его **мантисса сдвигается влево на два разряда** с соответствующим уменьшением порядка на две единицы: 0.1101**.**20.

### Умножение

|  |
| --- |
| **При умножении двух нормализованных чисел их порядки складываются, а мантиссы перемножаются.** |

**Пример 3.** Выполнить умножение двоичных нормализованных чисел:

(0.11101**.**2101)**.**(0.1001**.**211) = (0.11101**.**0.1001)**.**2(101+11) = 0.100000101**.**21000.

### Деление

|  |
| --- |
| **При делении двух нормализованных чисел из порядка делимого вычитается порядок делителя, а мантисса делимого делится на мантиссу делителя. Затем в случае необходимости полученный результат нормализуется.** |

**Пример 4.** Выполнить деление двоичных нормализованных чисел:

0.1111**.**2100 : 0.101**.**211 = (0.1111 : 0.101) **.**2(100-11) = 1.1**.**21 = 0.11**.**210.

Использование представления чисел с плавающей точкой существенно усложняет схему арифметико-логического устройства.