# АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЦВМ

# 1.1. Системы счисления

В повседневной практике для представления чисел люди пользуются почти исключительно десятичной системой счисления. Лишь в редких случаях встречаются остатки других систем - римский счет, двенадцатиричная система (часы), шестидесятиричная (минуты).

Однако система изображения чисел, которая веками складывалась применительно к ручному труду, не позволяет получить наиболее эффективные методы выполнения вычислений. По этой причине в вычислительной технике применяются другие системы счисления и чаще всего - двоичная.

Введем несколько определений.

***Cистема счисления*** - совокупность символов и правил для обозначения чисел.

Разделяют системы счисления позиционные и непозиционные. Непозиционная система счисления задается перечислением изображаемых в ней значений. Позиционная система счисления характеризуется основанием и тем, что числа, как правило, представляются несколькими разрядами (являются многоразрядными), а вес любого разряда определяется его позицией в числе.

***Oснование*** позиционной системы счисления определяет количество различных цифр (символов), допустимое в системе счисления. Это же число определяет, во сколько раз вес цифры данного разряда меньше веса цифры соседнего старшего разряда.

Так, в десятичной системе счисления, основание которой равно 10, различают 10 арабских цифр - 0, 1, 2, ..., 9. Следовательно, при ее использовании для записи числа, не превышающего девяти, достаточно одной цифры, и такое число записывается как одноразрядное. А в случае записи числа, большего девяти, оно представляется как многоразрядное. При этом вес каждого более старшего (расположенного слева от текущего) разряда в десять (основание системы счисления) раз больше текущего.

Так, например, число 359 - трехразрядное, и в нем 9 - цифра разряда единиц, 5 - цифра разряда десятков, 3 - цифра разряда сотен (в 10 раз превышает вес разряда десятков). При этом значение трехразрядного числа 359 получается суммированием трех слагаемых : 3 сотни + 5 десятков + 9 единиц.

Общее правило определения веса разряда многоразрядного числа таково:

 ***Если пронумеровать разряды целого числа справа налево, начиная от 0 для разряда единиц, то вес любого разряда получается возведением основания системы счисления в степень, значение которой равно номеру разряда.***

Так, вес самого младшего разряда целых чисел равен 1, поскольку номер разряда равен 0, а любое число, в том числе и число 10, возведенное в нулевую степень, дает в результате единицу. Вес следующего слева разряда равен 10 в степени 1, т.е. равен десяти, и т.д.

Это же правило справедливо и для записи дробных чисел. При этом разрядам справа от разряда единиц, имеющего номер 0, присваиваются отрицательные значения: -1, -2, и т.д., а их веса получаются также при возведении основания 10 в соответствующую степень. Так, например, вес третьего разряда в дробной части числа 42,9724 будет равен 10 в степени (-3), т.е. равен одной тысячной.

Указанное правило можно проиллюстрировать следующим образом:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число | 7 | 5 | 0 | 6 | 8 | **,** 2 | 5 | 9 |
| Номер разряда | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 |
| Вес разряда | 10000 | 1000 | 100 | 10 | 1 | 0,1 | 0,01 | 0,001 |

Как видно из примера, в позиционной системе счисления достаточно знать значение основания системы счисления, символы, изображающие отдельные цифры, и указанное правило, чтобы представить любое число.

В вычислительной технике широко применяют двоичную, восьмеричную и шестнадцатиричную систему счисления.

Двоичная система счисления имеет основание 2, и, следовательно, две разных цифры - 0 и 1; восьмеричная - восемь разных цифр - 0, 1, 2, 3, 4, 5, 6, 7, а шестнадцатиричная - шестнадцать цифр - десять арабских цифр от 0 до 9 и еще шесть символов -

 А (цифра, изображающая десять), D (цифра тринадцать),

 В (цифра одиннадцать), E (цифра четырнадцать),

 С (цифра двенадцать), F (цифра пятнадцать).

Проще всего сопоставить запись одних и тех же чисел в этих системах счисления можно с использованием таблицы 1, приведенной на следующей странице.

Мы уже говорили о том, что современные цифровые ЭВМ все используют в качестве основной двоичную систему счисления. К ее достоинствам относится:

* простота выполнения арифметических и логических операций, что влечет за собой простоту устройств, реализующих эти операции;
* возможность использования аппарата алгебры логики (булевой алгебры) для анализа и синтеза операционных устройств ЭВМ.

К неудобствам двоичной системы счисления относится необходимость перевода чисел из десятичной в двоичную и наоборот, а также то, что запись числа в двоичной системе громоздка (требует большего числа разрядов, чем привычная для человека десятичная). По этой и ряду других причин, кроме двоичной применяются восьмеричная и шестнадцатиричная системы счисления.

 Таблица 1.1

|  |
| --- |
| С и с т е м а с ч и с л е н и я |
| **10**  |  **2** | **8** | **16** |
| 0 |  0 |  0 |  0 |
| 1 |  1 |  1 |  1 |
| 2 |  0 1 |  2 |  2 |
| 3 |  1 1 |  3 |  3 |
| 4 |  1 0 0 |  4 |  4 |
| 5 |  1 0 1 |  5 |  5  |
| 6 |  1 1 0 |  6 |  6 |
| 7 |  1 1 1 |  7 |  7 |
| 8 |  1 0 0 0 | 1 0 |  8 |
| 9 |  1 0 0 1 | 1 1 |  9 |
| 10 |  1 0 1 0 | 1 2 |  A |
| 11 |  1 0 1 1 | 1 3 |  B |
| 12 |  1 1 0 0 | 1 4 |  C |
| 13 |  1 1 0 1 | 1 5 |  D |
| 14 |  1 1 1 0 | 1 6 |  E |
| 15 |  1 1 1 1 | 1 7 |  F |
| 16 | 1 0 0 0 0 | 2 0 | 1 0 |

Совместное использование указанных систем обусловлено двумя причинами:

* в восьмеричной и шестнадцатиричной системах любое число записывается более компактно, нежели двоичное;
* простотой преобразования из двоичной в восьмеричную (шестнадцатирич-ную) систему счисления и наоборот.

Приведем правила перевода чисел из двоичной системы в восьмеричную (шестнадцатиричную) и наоборот.

### П1 .Правило перевода “8с/с -> 2c/c”

При переводе многоразрядного числа каждую цифру исходного восьмеричного числа представить всегда точно тремя двоичными цифрами, взятыми из приведенной выше таблицы. При этом, если для записи соответствующей восьмеричной цифры в виде двоичной требуется менее трех двоичных цифр, двоичный эквивалент дополняется слева нулями (незначащие нули не исказят значения числа). Таким образом, например, при записи четырехразрядного восьмеричного числа должно получиться двенадцатиразрядное двоичное. После окончания такого преобразования можно отбросить старшие для всего числа незначащие двоичные цифры.

Отметим, что три цифры принято называть ***триадой***. Поэтому можно сказать, что при описываемом переводе каждая восьмеричная цифра заменяется соответствующей ей триадой двоичных цифр.

Если исходное число дробное, т.е. имеет целую и дробную часть, то в двоичном числе запятая ставится между триадами, представляющими соответствующие цифры исходного числа.

Пример.

Преобразуем восьмеричное число 371,62.

Для этого запишем для каждой цифры соответствующую триаду:

 3 --> 011

 7 --> 111

 1 --> 001

 6 --> 110

 2 --> 010

Теперь можно записать число в двоичной форме (для наглядности между триадами поместим пробелы):

#####  371,62 -> 011 111 001 , 110 010

И, наконец, запишем полученное двоичное число так, как это принято в математике, без незначащих нулей, а также отбросив правые нули в дробной части числа:

 371,62 -> 11111001,11001

### П2. Правило перевода “2с/с -> 8c/c”

При переводе многоразрядного двоичного числа в восьмеричную форму поступают следующим образом: Исходное число разбивают на триады. При этом для целой части числа разбиение проводят от местонахождения запятой влево, а для дробной части - от этого же места вправо. Затем самая левая группа при необходимости дополняется незначащими нулями до образования триады, а самая правая группа только в дробной части дополняется нулями справа также до образования полной триады. После этого каждая триада заменяется соответствующей восьмеричной цифрой. Местоположение запятой сохраняется по тем же правилам, что и в правиле П1.

Пример.

Представить двоичное число 1101100,01111101 в форме восьмеричного.

Разобьем исходное число на группы по три цифры, приняв в качестве точки отсчета местоположение запятой (для наглядности между триадами поместим пробелы):

#####  1 101 100 , 011 111 01

Теперь дополним до трех цифр нулями самую левую группу слева и самую правую группу справа:

#####  001 101 100 , 011 111 010

И, наконец, заменим каждую триаду соответствующей восьмеричной цифрой:

 001 101 100 , 011 111 100 --> 154,372

### П3. Правило перевода “16с/с -> 2c/c”

При переводе многоразрядного шестнадцатиричного числа в двоичную форму каждую цифру исходного числа заменяют группой точно из четырех двоичных цифр (заменяют ***тетрадой*** двоичных цифр). Местоположение запятой сохраняется по тем же правилам, что и в правиле П1. В окончательной записи можно отбросить самые левые (незначащие) нули и самые правые нули дробной части.

Пример. Преобразовать шестнадцатиричное число “6C,7D” в двоичную форму.

Для этого запишем для каждой цифры соответствующую тетраду:

 6 --> 0110

 C --> 1100

 7 --> 0111

 D --> 1101

Теперь можно записать число в двоичной форме (для наглядности между тетрадами поместим пробелы):

#####  6C,7D -> 0110 1100 , 0111 1101

И, наконец, запишем полученное двоичное число так, как это принято в математике, без незначащих нулей:

#####  6C,7D -> 1101100,01111101

### П4. Правило перевода “2с/с -> 16c/c”

При переводе многоразрядного двоичного числа в шестнадцатиричную форму поступают следующим образом. Исходное число разбивают на тетрады. При этом для целой части числа разбиение проводят от местонахождения запятой влево, а для дробной части от этого же места вправо. Затем самая левая группа при необходимости дополняется незначащими нулями до образования тетрады, а самая правая группа только в дробной части дополняется нулями справа также до образования полной тетрады. После этого каждая тетрада заменяется соответствующей шестнадцатиричной цифрой. Местоположение запятой сохраняется по тем же правилам, что и в правиле П1.

Пример. Представить двоичное число 1101100,01111101 в форме шест-надцатиричного.

Разобьем исходное число на группы по четыре цифры, приняв в качестве точки отсчета местоположение запятой (для наглядности между тетрадами поместим пробелы):

#####  110 1100 , 0111 1101

Теперь дополним до четырех цифр нулями слева самую левую группу:

#####  0110 1100 , 0111 1101

И, наконец, заменим каждую тетраду соответствующей шестнадцатиричной цифрой:

 0110 1100 , 0111 1101 -> 6С,7D.

Шестнадцатиричная и восьмеричная системы счисления используются для более компактной и удобной записи двоичных чисел.

Так, известность шестнадцатиричной системе принесло то, что с ее использованием удобно представлять программы в кодах большинства современных ЭВМ.

## 1.2. Перевод чисел из одной системы счисления

## в другую

Поскольку в практической деятельности люди привыкли оперировать десятичной системой счисления, а в ЭВМ числа представляются в двоичной, необходимо научиться преобразовывать числа из одной системы счисления в другую. Рассмотренные выше правила перевода из двоичной системы счисления в восьмеричную и шестнадцатиричную и наоборот носят частный характер и не могут быть распространены на другие системы. Здесь же мы рассмотрим общие правила перевода, справедливые для любой пары систем счисления, хотя и более громоздкие и трудоемкие по сравнению с рассмотренными выше.

Правила перевода целых и дробных чисел не совпадают, поэтому приведем три правила перевода чисел из системы счисления с основанием R в систему счисления с основанием Q.

### Правило 1. Перевод целых чисел

 Для перевода целого числа N, представленного в системе счисления (с/с) с основанием R, в с/с с основанием Q необходимо данное число делить на основание Q по правилам с/с с основанием R до получения целого остатка, меньшего Q. Полученное частное снова необходимо делить на основание Q до получения нового целого остатка, меньшего Q, и т.д., до тех пор, пока последнее частное будет меньше Q. Число N в с/с с основанием Q представится в виде не упорядоченной последовательности остатков деления в порядке, обратном их получению (иными словами, старшую цифру числа N дает последнее частное).

Пример. Преобразовать десятичное число 67 в двоичную форму.

Основание исходной системы счисления R=107. Основание новой системы счисления Q=2.

Согласно приведенному правилу надо исходное число 67 делить на основание новой системы (на 2) по правилам десятичной системы счисления (исходная с/с).

Поскольку процесс деления на 2 очень прост, воспользуемся следующим приемом: в левом столбце будем писать текущие частные, а в правом - текущие остатки от их деления на 2 (это может быть либо 0, либо 1):

 67 1 При делении 67 на 2 получается частное 33 и остаток 1;

 33 1 при делении 33 - частное 16 и остаток 1 и т.д.

 16 0

 8 0

 4 0

 2 0

 1 1 <- Старшая цифра числа.

 0

Теперь можно записать число 67 в новой системе счисления. Оно равно 1000011.

### Правило 2. Перевод правильной дроби

Перевод правильной дроби, представленной в с/с с основанием R, в с/с с основанием Q заключается в последовательном умножении этой дроби на основание Q по правилам системы счисления с основанием R, причем перемножают только дробные части. Дробь N в с/с с основанием Q представляется в виде упорядоченной последовательности целых частей произведений в порядке их получения. (Иными словами, старший разряд является первой цифрой произведения). Количество последовательных произведений определяет количество цифр в полученном числе.

Для многих чисел указанный процесс умножения потенциально никогда не кончается. Поэтому он продолжается до тех пор, пока не будет получено необходимое число цифр дробной части. При переводе числа с целью представления ее в “машинной” форме можно точно указать требуемое количество цифр. (Это будет рассматриваться позже, в разделе 1.5).

Пример. Перевести в двоичную систему счисления десятичную дробь 0,7243.

Основание исходной системы счисления R=10. Основание новой системы счисления Q=2.

Согласно приведенного правила исходное число 0,7243 надо умножать на основание новой системы (на 2) по правилам десятичной системы счисления (исходная с/с). Выполним серию умножений до получения, например, шести цифр в двоичном числе:

Искомые цифры дроби:

 0,7243 \* 2 = **1**,4486 1 -> старшая цифра

 0,4486 \* 2 = **0**,8972 0

 0,8942 \* 2 = **1**,7944 1

 0,7944 \* 2 = **1**,5888 1

 0,5888 \* 2 = **1**,1776 1

 0,1776 \* 2 = **0**,3552 0

 0,3552 \* 2 = **0**,7104 0

 Искомое представление число 0,7243 в двоичной системе счисления -> 0,101110.

Обратите внимание, что ***для получения шести цифр дроби выполнено семь умножений***

Это связано с необходимостью выполнить округление, чтобы представить дробь заданной длины более точно.

Из последнего примера, конечная дробь в одной системе счисления может стать бесконечной в другой. Это утверждение справедливо для всех случаев, когда одна система счисления не может быть получена возведением в целую степень основания другой.

Примеры.

* Десятичная дробь  **0,2** представляется бесконечной дробью  **0,33333...** в шестнадцатиричной системе счисления (основания с/с 10 и 16).
* Шестнадцатиричная дробь  **0,В1** представляется конечной дробью  **0,10110001** в двоичной системе счисления (основания с/с 16 и 2).

### Правило 3. Перевод неправильной дроби

Перевод неправильной дроби из одной системы счисления в другую осуществляется отдельно для целой и дробной части по правилам, изложенным выше.

## 1.3. Двоичные коды для десятичных цифр

В ряде случаев в вычислительной технике применяется не только двоичная, но и десятичная система счисления. Однако и в этом случае для представления десятичных цифр используется оборудование, разработанное для представления двоичных цифр. В этом случае говорят о двоично-десятичных кодах десятичных цифр.

Согласно формулы Хартли для представления 10 различных цифр требуется четыре бита информации:

 3 бита < I = log(10) < 4 бита.

Таким образом, при необходимости представить десять разных десятичных цифр комбинациями двоичных цифр, каждую из них можно представить минимум тетрадой двоичных чисел. Большинство кодов десятичных цифр использует тетрады, хотя есть и коды, в которых для кодирования используется большее число битов.

Наиболее распространены двоично-десятичные коды, в которых для представления десятичных цифр используются позиционные методы кодирования. Так, если рассматривать четыре двоичных разряда тетрады как четырехразрядное двоичное число, то веса ее отдельных разрядов слева направо будут равны соответственно 8, 4, 2 и 1.

Поэтому первый двоично-десятичный код, который мы рассмотрим, обозначается как код “8421”. Его можно назвать кодом с естественными весами.

В этом коде каждая десятичная цифра представляется ее двоичным эквивалентом :

 цифра 0 как 0000,

 цифра 1 как 0001,

 цифра 2 как 0010,

 цифра 5 как 0101,

 цифра 8 как 1000,

 цифра 9 как 1001.

В то же время, имея четыре двоичных цифры, можно представить не 10, а 16 различных комбинаций. Таким образом, при использовании кода “8421” шесть комбинаций : 1010, 1011, ..., 1111 останутся неиспользованными, т.е. не будут изображать ни одной из десятичных цифр. Эти комбинации считаются запрещенными.

 а) **Коды с избытком**

Кроме рассмотренной системы кодирования достаточно широко используются также так называемые коды с избытком. Рассмотрим группу кодов “8421” с избытком”.

Код “8421” с избытком W” строится по следующим правилам:

При кодировании десятичной цифры, к ней вначале прибавляют W, и затем полученное число представляют как двоичное в коде “8421”.

Значение W может быть равным 1, 2, 3, 4, 5 или 6. При любом значении избытка W шесть из шестнадцати комбинаций останутся неиспользованными. Только для разных избытков эти значения будут разными.

Пример. Рассмотрим код “8421” с избытком 3”.

а)Представим цифру 8 в данном коде.

Вначале увеличим 8 на 3. Получится 11.

Затем запишем 11 в коде “8421”. Получится 1011.

Число 1011 и есть представление цифры 8 в данном коде.

б)Восстановим цифру, которая изображается комбинацией 0101.

Вначале представим десятичное число, рассматривая комбинацию 0101, как его изображение в коде “8421”. Получится число 5.

Затем вычтем из него (из 5) избыток 3. Получится 2.

Это и есть искомый ответ: Комбинация 0101 изображает десятичную цифру 2 в коде “8421” с избытком 3”.

в)Восстановим цифру, которая изображается комбинацией 1110.

Восстановим десятичное число. Получится 14.

Вычтем из него избыток 3. Получится 11.

Поскольку 11 не является десятичной цифрой (это двухразрядное десятичное число), делаем вывод, что комбинация 1110 не изображает никакой десятичной цифры и является запрещенной.

 б)  **Код “2421”**

Кроме кодов с естественными весами разрядов применяются и другие. Одним из широко известных кодов является позиционный код, построенный с использованием тетрады двоичных цифр, веса которых слева направо равны соответственно : 2, 4, 2 и 1.

Представим коды цифр в таблице:

|  |  |  |  |
| --- | --- | --- | --- |
| Цифра | Код “2421” | Цифра | Код “2421” |
| 0 |  0000 | 5 |  0101 или 1011 |
| 1 |  0001 | 6 |  0110 или 1100 |
| 2 |  0010 или 1000 | 7 |  0111 или 1101 |
| 3 |  0011 или 1001 | 8 |  1110 |
| 4 |  0100 или 1010 | 9 |  1111 |

Как видно из таблицы, ряд десятичных цифр могут быть представлены двумя не совпадающими двоичными комбинациями.

Например, комбинации 0100 и 0010 изображают цифру 2, комбинации 1010 и 0100 изображают цифру 4 и т.д. Отличительной особенностью данного кода является то, что в нем нет неиспользованных (запрещенных) комбинаций.

в) **Код “2 из 5”**

Данный код принадлежит к непозиционным кодам. Как и все непозиционные коды он определяется табличным способом. Его название отражает принцип построения кода: любая десятичная цифра представляется комбинацией из 5 двоичных цифр, в которой точно две цифры 1 и, следовательно, три цифры 0.

Представим таблицу одного из возможных вариантов для данного кода:

|  |  |  |  |
| --- | --- | --- | --- |
| Цифра | Код | Цифра | Код |
| 0 | 11000 | 5 | 01010 |
| 1 | 00011 | 6 | 01100 |
| 2 | 00101 | 7 | 10001 |
| 3 | 00110 | 8 | 10010 |
| 4 | 01001 | 9 | 10100 |

Все остальные возможные комбинации, в которых число единиц не равно двум, являются запрещенными.

Также как и все коды на основе тетрады “8421”, последний код принадлежит к группе кодов, обладающих так называемыми диагностическими возможностями: Если известно, что некоторая комбинация должна изображать десятичную цифру, но попадает в область запрещенных, значит произошло искажение информации. Это свойство кодов активно используется в аппаратуре ЭВМ.

Для закрепления материала по переводу чисел из одной системы счисления в другую выполним несколько примеров.

Пример 1. Представить десятичное число 581 в двоичной, восьмеричной и шестнадцатиричной системах счисления.

Задачу можно решить с минимальными затратами усилий, выполнив, например, перевод в двоичную с/с по общему правилу (т. е. делением на основание 2 по правилам десятичной системы счисления), а затем из двоичной в восьми- и шестнадцатиричную системы счисления, используя упрощенные правила (кодированием соответственно тетрад и триад).

Пример 2. Представить десятичное число 993,761 в двоичной и шестнадцатиричной системах счисления.

Задачу можно решить таким же образом, как и предыдущую.

Пример 3. Представить шестнадцатиричное число 8363 в десятичной системе счисления.

Воспользуемся общим правилом. Для этого надо исходное число делить на 10 по правилам шестнадцатиричной системы счисления.

Что же это за правила? Это такие же правила сложения, вычитания, умножения и деления, что и в десятичной с/с, но над числами в позиционной шестнадцатиричной с/с.

Выполним перевод:

 **8 3 6 3** **A**

 \_(131)

 (130) **D 2 3** **A**

 ------- A

 1 6 ---- **1 5 0** **A**

 \_ (22) 3 2 \_(21)

 (20) \_(50) (20) **2 1** **A**

 ----- (50) ----- \_(33)

 2 3 ----- 1 0 (30) **3**

 \_(35) 0 **3** \_ (16) -----

 (30) (10) **3**

 ----- -----

 **5** **6**

Искомое число в десятичной системе равно 33635.

Примечание. В круглых скобках записаны десятичные эквиваленты соответствующих шестнадцатиричных чисел. При этом в каждой паре чисел, расположенных друг под другом, первое число - частичное делимое, а второе - произведение делителя на частичное частное. Так запись (131) - эквивалент шестнадцатиричного делимого 83, а (130) - результат умножения делителя А (=10) на D (13).

# 1.4. Арифметика цифровых вычислительных машин

Как уже говорилось выше, практически все современные цифровые ЭВМ в качестве основной используют двоичную систему счисления. А все арифметические операции над двоичными числами можно свести к двум элементарным - сложению и сдвигу двоичных кодов, изображающих числа. Это позволит технически реализовать четыре действия арифметики в одном устройстве, называемом арифметико-логическом (АЛУ), используя одни и те же электрические схемы.

### 1.4.1. Представление чисел со знаками

При выполнении арифметических операций в ЭВМ применяют прямой, обратный и дополнительный коды.

Как уже говорилось выше, кодом называют такую запись числа, которая отличается от естественной и общепринятой. Так вот, в математике естественной формой записи числа является запись, при которой непосредственно перед старшей значащей цифрой числа помещается знак плюс(+) или минус(-), а длина записи определяется величиной числа (иначе, количество символов, использованных для записи разных чисел, как правило, не совпадает). В ЭВМ это не так. Одной из важнейших характеристик любой ЭВМ является длина слова в ней. Длина слова определяется количеством двоичных разрядов слова.

Поэтому в ЭВМ,  ***вне зависимости от величины числа, его код всегда имеет фиксированное количество двоичных цифр.***

Кроме этого, в двоичном алфавите нет никаких символов, кроме цифр 0 и 1, и необходимы новые правила для указания знака числа. Суть этих правил сводится к тому, что ***знак плюс изображается цифрой 0, знак минус - цифрой 1, а цифра, изображающая знак всегда записывается самой первой в записи числа.***

Обратите внимание, что  ***код числа всегда содержит изображение его знака***, в отличие от математической записи, которая позволяет опускать знак плюс при изображении положительного числа.

Так, код 011101, согласно этим правилам, изображает положительное (самая левая цифра - 0) двоичное число 11101.

Для того, чтобы более просто, и, следовательно, более экономично реализовать устройство АЛУ применяют несколько разных кодов чисел. Это связано с тем, что разные операции в ЭВМ более просто реализуются в разных кодах.

При выполнении арифметических операций в ЭВМ применяют ***прямой, обратный*** и ***дополнительный*** коды чисел.

***Прямой код*** двоичного числа - это само двоичное число, в котором все цифры, изображающие его значение, записываются как в математической записи, а знак числа записывается двоичной цифрой.

При этом никакого символа, отделяющего эту цифру от старшей цифры, используемой при изображении его величины, не допускается. В таких случаях говорят о том, что назначение цифры в коде определяется его позицией.

Примеры.

 Изображаемое число Код

* +1101 (+13) 0000 1101 ( В примерах коды )
* +1011101 (+93) 0101 1101 ( изображаются )
* 1101 (-13) 1000 1101 ( восемью цифрами )

Итак, прямой код почти не отличается от принятого в математике: для выявления абсолютной величины (модуля) числа, надо отбросить цифру, обозначающую его знак.

Однако применительно к операциям сложения и вычитания такой код неудобен: правила счета для положительных и отрицательных чисел различаются. Чтобы прояснить это обстоятельство, представим что длина кода (слова) равна 5 двоичным разрядам и запишем несколько чисел в нем:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Число | -2 | -1 | 0 | +1 | +2 |
| Код | 10010 | 10001 | 00000 | 00001 | 00010 |

Как видно из примера, при использовании прямого кода при переходе значения число через ноль, происходит скачкообразное изменение кода**!**  Поэтому построение устройства, в котором должны выполняться такие действия арифметики, как сложение чисел с разными знаками и вычитание, становится сложной задачей.

Прямой код используется при хранении чисел в памяти ЭВМ, а также при выполнении операций умножения и деления.

Чтобы построить более простые схемы АЛУ предложены и активно применяются обратный и дополнительный коды.

***Обратный код*** положительного числа совпадает с прямым, а при записи отрицательного числа все его цифры, кроме цифры, изображающей знак числа, заменяются на противоположные ( 0 заменяется на 1, а 1 - на 0).

Примеры записи.

 Изображаемое число Код

* +1101 (+13) 0000 1101 ( В примерах коды )
* +1011101 (+93) 0101 1101 ( изображаются )
* 1101 (-13) 1111 0010 ( восемью цифрами )

Сопоставление этой записи с прямым кодом показывает, что непосредственно восстановить абсолютную величину (модуль) отрицательного числа непросто. Однако, в этом коде как к положительным, так и к отрицательным числам можно применять одни и те же правила, а операцию А-В можно заменить операцией сложения чисел А и “минус В”.

Посмотрим, как представляется последовательные числа при переходе через ноль:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Число | -2 | -1 | 0 | +1 | +2 |
| Код | 11101 | 11110 | 00000 | 00001 | 00010 |

Из примера видно, что переход через ноль также не выглядит естественным. Отмеченная особенность влечет за собой и следующее - в обратном коде ноль изображают две различающиеся комбинации: 00000 (+0) и 11111 (-0), что усложняет аппаратную реализацию операций.

Для восстановления прямого кода отрицательного числа из обратного кода надо все цифры, кроме цифры, изображающей знак числа, заменить на противоположные.

 ***Дополнительный код*** положительного числа совпадает с прямым, а код отрицательного числа образуется как результат увеличения на 1 его обратного кода.

Иными словами, процесс построения дополнительного кода отрицательного числа можно разбить на два этапа - построить обратный код, а затем из него построить дополнительный.

Проиллюстрируем это на примере.

 Число -> - 101101

 Прямой код -> 1101101

 Обратный код -> 1010010

 +1

Дополнительный -> 1010011

Примеры записи.

 Изображаемое число Код

* +1101 (+13) 0000 1101 ( В примерах коды )
* +1011101 (+93) 0101 1101 ( изображаются )
* 1101 (-13) 1111 0011 ( восемью цифрами )

В дополнительном коде, в отличие от обратного, ноль изображается только одной комбинацией, и кроме этого, достаточно естественно получается переход через ноль, если иметь в виду, что любое число, большее другого на 1, получается при прибавлении к этому другому 1 по правилам сложения. Применительно к дополнительному коду это именно так, если принять к сведению, что разрядность слова фиксирована, и единица переноса из старшего разряда теряется, поскольку ее некуда записать:

 2 -> 11101 + 1 = 11110

 1 -> 11110 + 1 = 11111

 0 -> 11111 + 1 = (1)00000 (перенос отбрасывается)

 +1 -> 00000 + 1 = 00001

 +2 -> 00001 + 1 = 00010

Для восстановления прямого кода числа из дополнительного нужно полностью повторить (и именно в том же порядке!) действия, которые использовались при переводе из прямого в дополнительный код: сначала все цифры, кроме цифры, изображающей знак, заменить на противоположные, а затем прибавить 1.

Основным достоинством дополнительного кода является то, что в нем единообразно реализуются операции сложения чисел разных знаков (алгебраическое сложение), а операцию вычитания можно свести к операции сложения заменой знака вычитаемого на обратный. Вспомнив, что в памяти ЭВМ числа хранятся в прямом коде, станет ясно, что замена знака вычитаемого может быть выполнена чрезвычайно просто (заменой знака числа в прямом коде на обратный). Именно по указанной причине дополнительный код применяется чаще обратного.

### 1.4.2. Сложение и вычитание чисел

Сложение и вычитание чисел в обратном и дополнительном кодах выполняется с использованием обычного правила арифметического сложения многоразрядных чисел. Общей для этих кодов особенностью (и очень удобной особенностью) является лишь то, что при поразрядном сложении чисел разряды, изображающие знаки чисел рассматриваются как равноправные разряды двоичного числа, которые складываются друг с другом и с единицей переноса из предыдущего разряда числа по обычным правилам арифметики. Различия же обратного и дополнительного кодов связаны с тем, что делается с единицей переноса из старшего разряда (изображающего, как неоднократно говорилось, знак числа).

При сложении чисел в дополнительном коде единица переноса из старшего разряда игнорируется (теряется), а в обратном коде эту единицу надо прибавить к младшему разряду результата.

Пример 1. Сложить числа +12 и -5.

а) В обратном коде

 Десятичная форма -> +12 -5

 Двоичная форма -> +1100 -101

 Прямой код -> 00001100 10000101

 Обратный код -> 00001100 11111010

Выполним сложение в столбик:

 0 0 0 0 1 1 0 0

 1 1 1 1 1 0 1 0

 ===============

(1) 0 0 0 0 0 1 1 0

 + 1 (Добавление 1 переноса)

 ==============

 0 0 0 0 0 1 1 1

Итак, результат в обратном коде = 00000111.

Поскольку знаковый разряд равен 0, результат положительный, и, следовательно, запись кода числа совпадает с записью прямого кода. Теперь можно восстановить алгебраическую запись результата. Он равен +111 (незначащие нули отброшены), или в десятичной форме +7.

Проверка (+12-5=+7) показывает, что результат верный.

а) В дополнительном коде

 Десятичная форма -> +12 -5

 Двоичная форма -> +1100 -101

 Прямой код -> 00001100 10000101

 Обратный код -> 00001100 11111010

 +1

 Дополнительный код -> 00001100 11111011

Выполним сложение в столбик:

 0 0 0 0 1 1 0 0

 1 1 1 1 1 0 1 1

 ============

(1) 0 0 0 0 0 1 1 1

 (Перенос игнорируется)

Итак, результат в дополнительном коде = 00000111.

Поскольку знаковый разряд равен 0, результат положительный, и, следовательно, запись кода числа совпадает с записью прямого кода. Теперь можно восстановить алгебраическую запись результата. Он равен +111 (незначащие нули отброшены), или в десятичной форме +7.

Проверка (+12-5=+7) показывает, что результат верный.

Умножение и деление двоичных чисел производится в ЭВМ в прямом коде, а знаки их используются лишь для определения знака результата. Также как и в математике, умножение сводится к операциям сложения и сдвига. Деление выполняется за счет комбинирования сдвигов, вычитаний (в этот момент могут использоваться обратный или дополнительный коды) и сложений.

# 1.5. Кодирование чисел в ЭВМ

В ЭВМ применяется чаще всего одна из двух форм представления чисел:

* с фиксированной запятой;
* с плавающей запятой.

Числа представляются в машинном слове, имеющем для конкретной ЭВМ всегда фиксированное число разрядов (битов). Это число является одной из важнейших характеристик любой ЭВМ и называется разрядностью машины. Разные разряды слова при кодировании команд и данных имеют несовпадающие функциональные назначения. При рассмотрении их функций используют также термин “разрядная сетка машины”.

### 1.5.1. Числа с фиксированной запятой

В числах с фиксированной запятой положение запятой в разрядной сетке машины заранее обусловлено для всех чисел раз и навсегда. Поэтому в коде числа запятая никак не обозначается. В большинстве машин место запятой подразумевается после последней цифры (справа от нее). А такие числа - целые. При необходимости представлять дробные числа с использованием формы с фиксированной запятой программист должен алгоритмическими средствами обеспечить использование множителя, выполняющего функцию масштабирования (масштабного множителя).

Определим диапазон представимых чисел.

Вначале рассмотрим пример, в котором положим, что мы имеем дело с ***десятичной*** (а не двоичной) системой счисления, и что для записи абсолютной величины числа (без учета его знака) в нашем распоряжении имеется шесть разрядов.

Тогда максимальное (по абсолютной величине) целое будет равно 999999 или иначе 10\*\*6-1. А поскольку в разрядной сетке машины для записи знака числа всегда предусматривается один разряд, то для нашего случая диапазон представимых чисел составит все целые числа, начиная от

-999999 до +999999, а количество различных целых - 2\*10\*\*6-1.

В двоичных ЭВМ их разрядность определяется числом разрядов в слове. Так, если разрядность некоторой ЭВМ равна 16, то один разряд отводится для кодирования знака числа, а остальные 15 - для записи его величины. При этом максимальное по модулю целое значение в машинном слове будет равно 2\*\*15-1, что составит 32767. (Посмотрите диапазон целых (***integer***) чисел в языке программирования Паскаль для ПЭВМ типа IMB PC).

В общем случае, если разрядность машины составляет N битов. Тогда максимальное по абсолютной величине целое число, которое можно в ней записать, будет равно 2\*\*(N-1)-1.

###  Особенности арифметических операций над числами

Поскольку (если положение запятой фиксировано после последней цифры числа) числа с фиксированной запятой - целые, они представляются в машине точно. А потому операции сложения, вычитания и умножения корректны всегда: как операнды, так и результат - целые числа.

Единственной особенностью, о которой необходимо упомянуть, является ситуация, которая носит название “переполнение разрядной сетки” (FixedOverflow - переполнение с фиксированной запятой) и которая возникает, когда результат умножения превышает максимально возможное для данной разрядности значение. Эта ситуация считается в ЭВМ исключительной. При ее возникновении записать получившееся значение невозможно. В этом случае устанавливается в “1” специальный флаг переполнения, старший бит результата (бит переноса из старшего разряда слова) теряется, а в качестве результата выдается искаженное число. Описываемая ситуация не считается критической, и после окончания данной операции вычисления продолжаются. Таким образом, программист сам должен позаботиться о корректной реакции на возникновение переполнения, используя для обнаружения указанной ситуации содержимое флага переполнения.

Иначе обстоит дело с операцией деления. При делении целого числа на другое целое результат совсем не обязательно должен быть целым. А поскольку и результат должен быть представлен целым числом, возникает коллизия, которую проиллюстрируем примером:

 5 / 2 = 2

 5 / 3 = 1

 5 / 4 = 1

 5 / 5 = 1

 5 / 6 = 0

И в отличие от умножения, с позиций ЭВМ никаких ошибок при этом нет, и никакие флаги не устанавливаются, а указанные особенности деления целых должны учитываться программистом самостоятельно. В ряде языков программирования эти особенности отражаются набором допустимых арифметических операций. Так, например, в языке Паскаль для целых (integer) определены две операции:

 ***div*** - целочисленное деление, при котором в качестве результата представляется целая часть частного,

 ***mod*** - остаток от деления целых (деление по модулю), при котором в качестве результата представляется целый остаток от деления, по абсолютной величине меньший делителя.

Примеры:

 5 ***div*** 3 = 1

 5 ***mod*** 3 = 2

### 1.5.2. Числа с плавающей запятой

В форме с плавающей запятой число представляется двумя компонентами : мантиссой и порядком. Мантисса используется для записи цифр числа, а порядок - для указания положения запятой.

Разрядная сетка машины в этом случае делится на несколько частей:

***один разряд*** - для кодирования знака числа (это всегда самый старший, левый, разряд слова);

***M разрядов*** - для записи мантиссы;

***Р разрядов*** - для записи порядка (с учетом его знака).

Местоположение запятой при этом тоже строго фиксируется: считается, что мантисса всегда представляется как число, меньшее единицы, но такое, в котором первая цифра после запятой для всех абсолютно чисел отлична от нуля (единственное исключение составляет число 0). Такая форма представления мантиссы называется нормализованной. Иначе говорят, что мантисса нормализована (приведена к виду: 1 < M <= 0,1).

Ну, а если известно, что мантисса имеет вид “0,цццц..”, то ее код в машинном слове может не содержать символов “0,”, а местоположение запятой предполагается перед старшей значащей цифрой мантиссы.

Порядок Р всегда представляется целым числом со знаком + или -. А для кодирования абсолютной величины порядка остается (Р-1) цифр.

Теперь можно рассмотреть диапазон представимых чисел.

Вначале рассмотрим пример применительно к двоичной системе счисления.

Пусть ***m*** - количество разрядов мантиссы,

 ***р*** - количество разрядов порядка, включая знаковый.

Тогда максимальное по абсолютной величине число будет равно

 0,1111..1 \* 2\*\*(+111..1) = (1-2\*\*(-***м***))\*2\*\*(2\*\*(***р***-1)-1),

 ***m*** цифр (***p***-1) цифр

или приблизительно 2\*\*(2\*\*(***р***-1)-1),

а минимальное по абсолютной величине число

 0,1000..0 \* 2\*\*(-111..1) = 2\*\*(-2\*\*(***р***-1)).

 ***m*** цифр (***p***-1) цифр

Итак, число в форме с плавающей запятой представляется последовательностью битов без каких либо явно указанных разделителей, но функционально разбитой на три группы {(знак числа, мантисса числа, порядок числа) или (знак числа, порядок числа, мантисса числа)}.

Рассмотренная форма кодирования числа приводит к следующим последствиям:

* Диапазон чисел, представимых в форме с плавающей запятой, определяется главным образом разрядностью порядка (Р).
* Разрядность мантиссы (М) определяет точное количество значащих цифр в изображении числа.

Следовательно, большинство чисел в форме с плавающей запятой представляется приближенно и причиной этого является ограниченное число разрядов мантиссы. Величина же абсолютной погрешности при приближенном представлении числа зависит как от абсолютной величины числа, так и от разрядности мантиссы и порядка.

Рассмотрим примеры. При этом для простоты положим, что числа представляются в десятичной системе счисления, количество цифр мантиссы равно 4, количество цифр порядка - 2, знак порядка записывается как в математике, а знак числа мы не изображаем, полагая все числа положительными.

Пример 1. Пусть имеется число 12,42=0,1242\*10\*\*(+2).

В заданном формате оно представляется цепочкой символов

 **1 2 4 2 + 0 2**

При этом

* цепочка “1 2 4 2” представляет мантиссу, т.е. в математическом смысле число 0,1242 ,
* а цепочка “+ 0 2” - порядок - целое положительное число 2.

Тогда ближайшее большее этого число может быть задано цепочкой

 **1 2 4 3 + 0 2**

и оно равно 0,1243\*10\*\*(+2)= 12,43.

Таким образом, ближайшие числа на числовой оси, которые различимы при кодировании их в форме с плавающей запятой для данного примера различаются на 0,01 (абсолютная погрешность представления всех чисел между 12,42 и 12,43 имеет верхнюю оценку 0,01).

Пример 2. Пусть имеется число 0,001242=0,1242\*10\*\*(-2).

В заданном формате оно представляется цепочкой символов

 **1 2 4 2 - 0 2**,

а ближайшее большее этого число представляется цепочкой

 **1 2 4 3 - 0 2**

и равно 0,1243\*10\*\*(-2)= 0,001243.

Таким образом, абсолютная погрешность представления всех чисел между 0,001242 и 0,001243 имеет верхнюю оценку 0,000001.

Пример 3. Пусть имеется число 0,1242\*10\*\*(+12).

В естественной форме записи это число 124 200 000 000, а в заданном формате оно представляется цепочкой символов

 **1 2 4 2 + 1 2**,

а ближайшее большее этого число представляется цепочкой

 **1 2 4 3 + 1 2**

и равно 0,1243\*10\*\*(+12)= 124 300 000 000.

Таким образом, абсолютная погрешность представления всех чисел между 124 200 000 000 и 124 300 000 000 имеет верхнюю оценку 100 000 000 = 10\*\*8.

Обратите внимание, что в последнем примере невозможно записать ни одного числа в интервале размером 10\*\*8.

Важный вывод, который следует из анализа формы кодирования чисел с плавающей запятой и иллюстрируется в рассмотренных примерах: числа в форме с плавающей запятой, несмотря на то что, эта форма предложена для представления в ЭВМ непрерывных величин, представляются дискретным множеством на числовой оси и располагаются на ней неравномерно.

Если изобразить на (бесконечной) числовой оси области существования чисел, то можно выделить следующие области (см. рис.):

 **1 2 3 4 5 6**

  **R**

 ***МаксВещ -МинВещ 0 +МинВещ +МаксВещ***

* область 1: **Х<-*МаксВещ*** - ни одного значения из области нельзя представить в машинном слове (***МаксВещ*** - максимальное по абсолютной величине число, которое можно закодировать);
* область 2:  **-*МаксВещ*<=X<=*-МинВещ*** - в данном интервале может быть представлено столько различных чисел, сколько их можно записать по заданной разрядности мантиссы и порядка;
* область 3: ***-МинВещ*<X<*0*** - ни одного значения из этой области представить в машинном слове нельзя;
* область 4:  ***0*<X<*+МинВещ*** - ни одного значения из этой области представить в машинном слове нельзя;
* область 5: ***+МинВещ*>=X>=*+МаксВещ*** - в данном интервале может быть представлено столько различных чисел, сколько их можно записать по заданной разрядности мантиссы и порядка;
* область 6: **X>*+МаксВещ*** - ни одного значения из области нельзя представить в машинном слове (МаксВещ - максимальное по абсолютной величине число, которое можно закодировать).

Особое место занимает величина ***0***. Она также кодируется в форме с плавающей запятой, причем как ее порядок, так и мантисса(!) полагаются равными нулю.

###  Особенности арифметических операций над числами

При выполнении арифметических операций все четыре действия арифметики корректны. Следует однако иметь в виду, что дискретный характер представления чисел в форме с плавающей запятой и разбиение числовой оси на области, в ряде из которых невозможно представить ни одного числа, приводит:

* Во-первых, к тому, что при выполнении арифметической операции теоретически возможно формирование результата, который попадает в области 2 или 5, но который нельзя закодировать в форме с плавающей запятой точно. В этом случае, результат заменяется ближайшим из множества допустимых значений с учетом правила округления (ошибка метода представления чисел, вызванная ограниченной разрядностью мантиссы).
* Во-вторых, к тому, что при выполнении арифметической операции теоретически возможно формирование результата, который попадает в область 1 или в область 6. Этот случай является критическим, поскольку результат представить нельзя принципиально. Рассматриваемая ситуация называется “Переполнение с плавающей запятой” (Overflow), а при ее возникновении происходит аппаратное прерывание работы ЭВМ и выполнение программы аварийно прекращается. Причиной этого является ограниченная разрядность порядка.
* В-третьих, к тому, что при выполнении арифметической операции теоретически возможно формирование результата, который попадает в область 3 или в область 4. Рассматриваемая ситуация называется “Потеря значимости”, а при ее возникновении результат заменяется ближайшим допустимым, как правило нулем. Выполнение программы после этого продолжается. В некоторых ЭВМ при этой ситуации вырабатывается предупредительное (информационное) сообщение. Причиной этой ситуации также является ограниченная разрядность порядка.

В заключении отметим, что при выполнении арифметических операций мантиссы чисел и их порядки обрабатываются по разным алгоритмам. При этом в операциях сложения и вычитания чисел порядки выравниваются за счет сдвига мантиссы меньшего операнда на число разрядов, равное разнице порядков операндов, а в операциях умножения и деления порядки чисел соответственно складывают или вычитают. Поскольку, как мы уже видели раньше, вычитание алгебраических чисел (т.е. с учетом их знаков) в прямом коде реализовать не просто, а порядки представляются как числа целые со знаком в прямом коде, в ряде ЭВМ при представлении числа с плавающей запятой порядок числа заменяется его характеристикой.

Характеристика числа получается из его порядка, если осуществить преобразование координат: Значение 0 на оси, изображающей характеристику, совпадает с значением **-*МаксПорядок***:

Порядок:

 ***МаксПорядок 0 +МаксПорядок***

Характеристика:

 ***0 МаксХаракт*.**

При этом характеристика числа рассматривается только как положительное число, а следовательно, в нем не надо и кодировать знак. Признаком же того, какой знак имеет порядок некоторого числа, является содержимое старшего разряда характеристики: Если он равен 0 - порядок отрицательный, в противном случае - порядок положительный. В случае записи характеристики цепочкой цифр 1000..0 принимается, что порядок равен нулю.

Рассмотрим еще одну ситуацию, типичную для операции над числами в форме с плавающей запятой.

Пусть необходимо вычислить разницу чисел

 **X**=13,45 и **Y**=13,45\*10\*\*(-5) ,

при условии, что они представлены в форме с плавающей запятой при разрядности мантиссы, равной 4, и порядка, равной 2. Для простоты операцию проиллюстрируем на примере десятичной системы счисления.

Запишем числа Х и Y в форме с плавающей запятой:

 **X**: 1 3 4 5 + 0 2

 **Y:** 1 3 4 5 - 0 3

Как видно из этой записи, оба числа представлены в форме с плавающей запятой без искажения. Не воспроизводя логику вычитания, принятую в ЭВМ, выполним вычитание в столбик. Для этого представим оба операнда в естественной форме и так, чтобы соответствующие разряды операндов находились друг под другом :

 **X**: 1 3 , 4 5

 **Y**: 0 , 0 0 1 3 4 5

 =============

######  **X-Y**: 1 3 , 4 4 8 6 5 5

Округлим результат, учитывая, что в нашем распоряжении для записи цифр числа имеется всего 4 разряда, и запишем его вновь в форме с плавающей запятой, в заданной разрядной сетке:

######   **X-Y:** **1 3 4 5 + 0 0**

Сравнив результат с исходными операндами увидим, что хотя оба операнда были отличны от нуля, результат и уменьшаемое полностью совпадают!

**Вывод.** При вычитании двух чисел большое значение имеют соотношение их величин и разрядность мантисс, используемая для их кодирования. Так что программисты могут столкнуться с нежелательными последствиями выполнения указанных действий в некоторых критических местах алгоритма. Например, если подобное вычитание выполняется в условии прекращения цикла, имеющем вид “(X-Y)>0.01”, то данное условие может никогда не выполниться, т.е. произойдет так называемое зацикливание.

Дополнительная литература по материалу раздела.

 1. Савельев А.Я. Прикладная теория цифровых автоматов. Учебник для втузов. М.: 1989.