ПРЕДСТАВЛЕНИЕ ДАННЫХ

Чтобы сделать компьютеры более надежными и простыми, в них применяют схемы, которые могут находиться только в двух состояниях; одно из них обозначается 0, а другое - 1. С помощью комбинаций из нескольких 0 и 1 можно представить любое число различных объектов. Комбинация, состоящая из одного 0 или одной 1, называется *битом*. В общем случае *n* бит могут представлять 2" различных объектов и добавление еще одного бита удваивает число возможных комбинаций.

В компьютерах цепочки бит представляют собой числа, буквы, знаки пунктуации и любую другую информацию. Числа ассоциируются с двоичными комбинациями в соответствии с *числовыми форматами*. Имеются три основных формата:

- двоичный (или целый);
- плавающая точка (или вещественный);
- двоично-кодированный десятичный (BCD или десятичный).

Форматы целого и плавающей точки соответствуют типам целых и вещественных чисел, которые применяют в языках высокого уровня.

Символьный код устанавливает соответствие букв и других символов двоичным комбинациям.

ДВОИЧНЫЙ ФОРМАТ

Компьютер работает с двоичной информацией. Человеку удобнее интерпретировать двоичную информацию посредством шестнадцатеричной системы счисления, а производить вычисления, используя десятичную систему счисления.

Системы счисления

Системой счисления называется совокупность правил записи чисел. Системы счисления подразделяются на *позиционные* и *непозиционные*. Как позиционные, так и непозиционные системы счисления используют определенный набор символов — *цифр*, последовательное сочетание которых образует число. Непозиционные системы счисления появились раньше позиционных. Они характеризуются тем, что в них символы, обозначающие то или иное число, не меняют своего значения в зависимости от местоположения в записи этого числа. Классическим примером такой системы счисления является *римская*. В ней для записи чисел используются буквы латинского алфавита. При этом буква I означает единицу, V — пять, X — десять, L — пятьдесят, C — сто, D — пятьсот, M — тысячу. Для получения количественного эквивалента числа в римской системе счисления необходимо просто просуммировать количественные эквиваленты входящих в него цифр. Исключение из этого правила составляет случай, когда младшая цифра идет перед старшей, — в этом случае нужно не складывать, а вычитать число вхождений этой младшей цифры. К примеру, количественный эквивалент числа 577 в римской системе счисления - это DLXXVII = 500 + 50 + 10 + 10 + 5 + 1 + 1 = 577. Другой пример:

CDXXIX = 500 - 100 + 10 + 10 - 1 + 10 = 429.

В позиционной системе счисления количество символов в наборе равно *основанию* системы счисления. Место каждой цифры в числе называется *позицией*. Номер позиции символа (за вычетом единицы) в числе называется *разрядом*. Разряд 0 называется *младшим* разрядом. Каждой цифре соответствует определенный количественный эквивалент (позиционный вес). Введем обозначение — запись $A_{(p)}$ будет означать количественный эквивалент числа A, состоящего из *n* цифр a_k (где $k = 0, ..., n-1$) в системе счисления с основанием p. Это число можно представить в виде последовательности цифр:

$A_{(p)} = a_{n-1}a_{n-2}...a_1a_0$,при этом всегда выполняется неравенство $a_k < p$.

В общем случае, количественный эквивалент некоторого положительного числа A в позиционной системе счисления можно представить выражением

$$A_{(p)} = a_{n-1} * p^{n-1} + a_{n-2} * p^{n-2} + ... + a_1 * p^1 + a_0 * p^0$$

где: p — основание системы счисления (некоторое целое положительное число);
a — цифра данной системы счисления; n — номер старшего разряда числа.

Для получения количественного эквивалента числа в некоторой позиционной системе счисления необходимо сложить произведения количественных значений цифр на степени основания, показатели которых равны номерам разрядов (обратите внимание, что нумерация разрядов начинается с нуля).

Двоичная система счисления

Набор цифр для двоичной системы счисления:

{0, 1}, основание степени p = 2.

Например, рассмотрим двоичное число 10100111. Количественный эквивалент этого двоичного числа равен сумме:
$1*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0=167$.

Сложение и вычитание двоичных чисел (рис. 6.1) выполняется так же, как и для других позиционных систем счисления, например десятичной. Точно так же выполняются заем и перенос единицы из (в) старший разряд. К примеру:

```
 11   11111    перенос        1  1   1       заем
  110011011               11010010011
+ 110010101             − 00111011011
-----------             ---------------
 1100110000               10010111000
```

Степени двойки

k	2^k	k	2^k
1	2	9	512
2	4	10	1024
3	8	11	2048
4	16	12	4096
5	32	13	8192
6	64	14	16384
7	128	15	32768
8	256	16	65536

Шестнадцатеричная система счисления

Данная система счисления имеет следующий набор цифр:
{0, 1, 2, ..., 9, A, B, C, D, E, F}, основание системы p = 16.
Например, рассмотрим шестнадцатеричное число ed23c. Количественный эквивалент этого числа равен
$14*16^4+13*16^3+2*16^2+3*16^1+12*16^0= 971324$.

Соответствие десятичных, двоичных и шестнадцатеричных чисел

Десятичное число	Двоичная тетрада	Шестнадцатеричное число
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4

5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A, a
11	1011	B,b
12	1100	C, c
13	1101	D, d
14	1110	E, e
15	1111	F, f
16	10000	10

Сложение и вычитание шестнадцатеричных чисел.

```
 1 1          перенос          1 1          заем
   E F 1 5    1 слагаемое    - B C D 8      уменьшаемое
+  C 1 E 8    2 слагаемое      5 E F 4      вычитаемое
 -------------                 ----------
 1 B 0 F D    результат        5 D E 4      результат
```

Перевод чисел из одной системы счисления в другую

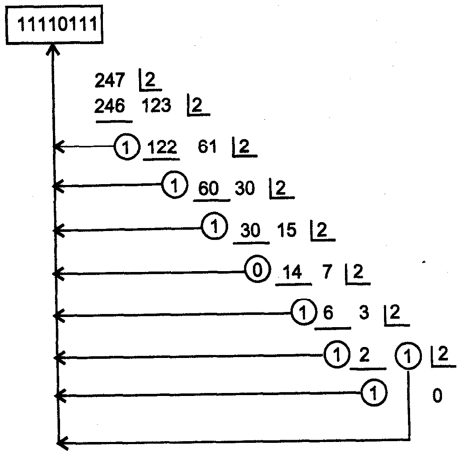
Перевод в десятичную систему счисления

Этот тип перевода наиболее прост. Обычно его производят с помощью так называемого *алгоритма замещения*, суть которого заключается в следующем: сначала в десятичную систему счисления переводится основание степени p, а затем — цифры исходного числа. Результаты подставляются в формулу $A_{(p)} = a_{n-1} * p^{n-1} + a_{n-2} * p^{n-2} + ... + a_1 * p^1 + a_0 * p^0$. Полученная сумма и будет искомым результатом.

Алгоритм перевода из десятичной системы счисления в двоичную

1. Разделить десятичное число A на 2. Запомнить частное q и остаток a.
2. Если в результате шага 1 частное $q \neq 0$, то принять его за новое делимое и отметить остаток a, который будет очередной значащей цифрой числа, вернуться к шагу 1, на котором в качестве делимого (десятичного числа) участвует полученное на шаге 2 частное.
3. Если в результате шага 1 частное $q = 0$, алгоритм прекращается. Выписать остатки в порядке обратном их получению. Получится двоичный эквивалент исходного числа.

К примеру, требуется перевести число 247_{10} в двоичную систему счисления :



Перевод из шестнадцатеричной системы счисления в двоичную

Необходимо заменить шестнадцатеричные цифр соответствующими двоичными тетрадами.

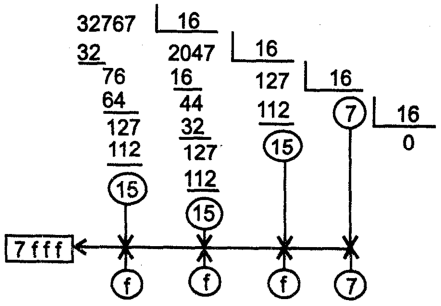
Например $e4d5_{16}$ ->• 1110 0100 1101 0l0l$_2$

Перевод из десятичной системы счисления в шестнадцатеричную

Общая идея такого перевода аналогична рассмотренной выше в алгоритме перевода в двоичную систему счисления из десятичной.

1. Разделить десятичное число А на 16. Запомнить частное q и остаток а.
2. Если в результате шага 1 частное $q \neq 0$, то принять его за новое делимое, записать остаток и вернуться к шагу 1.
3. Если частное $q = 0$, прекратить работу алгоритма. Выписать остатки в порядке обратном их получению. Получится шестнадцатеричный эквивалент исходного десятичного числа.

Например, преобразовать $32\ 767_{10}$ в шестнадцатеричную систему счисления.



Перевод из двоичной системы счисления в шестнадцатеричную

Двоичное число разбивается на тетрады, начиная с младшего разряда. Далее каждая тетрада приводится к соответствующему шестнадцатеричному числу

Например, преобразовать $1110010110101111010110001101100011110101010110l_2$ в шестнадцатеричную систему счисления.

Разобьем его на тетрады:
0111 0010 1101 0111 1010 1100 0110 1100 0111 1010 1010 1101.

По тетрадам приводим последовательности нулей и единиц к шестнадцатеричному представлению: 72d7ac6c7aad

Числа со знаком

Для представления отрицательных целых чисел можно ввести дополнительный бит, обозначающий знак. В этом случае 0 обычно соответствует знаку плюс, 1 — знаку минус, а получающийся формат называется *прямым кодом*. Однако этот формат на практике

используется редко. В компьютерах почти всегда применяется *дополнительный код*, в котором отрицательные целые числа представляются в виде

$-b=2^n - b$

где b — абсолютное значение целого числа, а n — число бит, используемых для его представления. Для знака выделяется старший бит поля, представляющего число. Естественно, что физически этот бит ничем не отличается от других — все зависит от команды, работающей с данным полем. Если в ее алгоритме заложена работа с целыми числами со знаком, то она будет по-особому трактовать старший бит поля.

Например, если $n = 16$ и $b = 1234_{10} = 10011010010_2 = 04D2_{16}$, то $-b$ в дополнительном коде равно $2^{16} - 1234 = 1111101100101110_2 = = FB2E_{16}$. При использовании n бит допускается однозначное представление целых чисел в диапазоне $—2^{n-1} . . . 2^{n-1} — 1$. Когда $n = 16$, целые числа из диапазона —32768 . . . 32767 записываются в следующем виде:

$1000000000000000_2 = 8000_{16}$
$1000000000000001_2 = 8001_{16}$
……
$1111111111111111_2 = FFFF_{16}$
$0000000000000000_2 = 0000_{16}$
$0000000000000001_2 = 0001_{16}$
……
$0111111111111111_2 = 7FFF_{16}$

Для упрощения вычислений дополнительный код числа b можно образовать, заменив нули на единицы и единицы на нули, что дает $(2^n — 1) — b$ *(обратный код b)*, и прибавив 1.

Пусть $n = 8$ и $b = 46$:

$2^8 – 1$ =11111111
b =00101110
$(2^8 - 1) - b$ =11010001 обратный код
\+ 1
$2^8 - b$ = 11010010 дополнительный код

Можно показать, что при обычном сложении знаковых целых чисел в дополнительном коде результат имеет правильное представление, если сохранять только младшие n бит. Вычитание выполняется путем прибавления дополнительного кода вычитаемого. Например, для $n = 8$:

```
 72 =  01001000      01001000
+
-35 = -00100011      11011101
---                  --------
 37                1 00100101 = 37
                     ↑ Пропадает

-29 =  11100011      11100011
+
-(-90) = -10100110   01011010
---                  --------
 61                1 00111101 = 61
                     ↑ Пропадает
```

При работе с числами со знаком потребуется умение выполнять обратное действие — имея двоичное дополнение числа, определить значение его модуля. Для этого необходимо выполнить два действия:

1. Выполнить инвертирование битов двоичного дополнения.
2. К полученному двоичному числу прибавить двоичную единицу.

К примеру, определим модуль двоичного представления числа

$-185_{10} = 1111111101000111_2$:

инвертируем биты -> 0000000010111000_2.

Добавляем двоичную единицу: + 0000000000000001_2

$0000000010111001_2 = |-185|$

ДВОИЧНО-КОДИРОВАННЫЙ ДЕСЯТИЧНЫЙ ФОРМАТ

В *двоично-кодированном десятичном формате* (или BCD-формате) десятичные цифры хранятся в виде 4-битных двоичных эквивалентов. Имеются две основные разновидности этого формата: упакованный и неупакованный. В упакованном BCD-формате цепочка десятичных цифр хранится в виде последовательности 4-битных групп, например число 9502 — в виде 1001 0101 0000 0010. В неупакованном BCD-формате каждая цифра находится в младшей тетраде 8-битной группы, а содержимое старшей тетрады несущественно. Число 9502 будет храниться в виде

xxxx1001 xxxx0101 xxxx0000 xxxx0010

В отличие от двоичного дополнительного кода, который применяется для представления отрицательных целых чисел в двоичном формате, при отсутствии в системе специальных схем соглашение о знаке для BCD-формата устанавливается программистом (и после введения такого соглашения его необходимо неукоснительно соблюдать). Отрицательные целые числа допускают представления в десятичном обратном или в десятичном дополнительном коде. При выборе десятичного обратного кода для знаков плюс и минус берутся две из неиспользуемых 4-битных комбинаций, например 1100 обозначает плюс, а 1101 — минус. Знак допускается размещать до или после цепочки цифр.

Десятичный n-разрядный дополнительный код произвольного целого числа *d* определяется как 10^n — *d*. При заданном *n* диапазон целых чисел, которые однозначно представляются в десятичном дополнительном коде, составляет от-5х 10^{n-1} до5х 10^{n-1} -1.В упакованном BCD-формате при *n* = = 8 для хранения целых чисел из диапазона —50000000 . . . 49999999 потребуются 32 бита. Десятичный дополнительный код обладает такими же свойствами, что и двоичный дополнительный, и сложение дает правильный результат в десятичном дополнительном коде, например для *n* = 4 имеем:

$$\begin{array}{rcl} 252 & = & 0252 \\ + & & \\ (-485) & = & 9515 \\ \hline -233 & & 9767 \end{array}$$

Вычитание сводится к прибавлению к уменьшаемому отрицательного вычитаемого.

БУКВЕННО-ЦИФРОВЫЕ КОДЫ

Для представления символьной информации используется код ASCII. При нажатии клавиши на терминале производится формирование и передача в компьютер соответствующего кода ASCII. Первые 32 символа кода являются служебными. К ним относятся код 10 перевод строки, 13 – возврат каретки, 9 – табуляция и другие. Код ASCII передается в двоичном виде.

Например, символьная цепочка

JOHN соответствует цепочке

4a 4f 48 4e

Ни цифра 0, ни пробел не соответствуют нулевой комбинации. Двоичная комбинация, состоящая из нулей, называется *пустым символом* и не вызывает никаких действий.

Число бит, которое необходимо в коде для представления символа, называется длиной. Код длиной *n* допускает идентификацию 2^n символов.

Числа, представляющие собой цифры, следуют в возрастающем порядке, поэтому для сравнения значений применимы арифметические действия непосредственно над кодами чисел.

Числа передаются в (из) компьютер (а) в виде последовательности кодов. Например, число 7902 передается как 37 39 30 32 . Такое число соответствует неупакованному BCD-формату. Неупакованные BCD-числа не требуется преобразовывать для операций ввода-вывода, но они занимают в памяти больше места. Беззнаковое целое число 7902 требует 32 бита памяти в неупакованном BCD-формате, 16 бит в упакованном и всего 13 бит в двоичном. Арифметические операции с числами в двоичном формате выполняются быстрее, чем в BCD-формате.