**Позиционные** **системы** **счисления**

***Системой*** ***счисления*** называют способ изображения произ-вольного числа ограниченным набором символов, называемых цифрами. Номер позиции, определяющий вес, с которым данная цифра складывается в числе, называют ***разрядом***, а системы счис-ления, обладающие отмеченным свойством, — ***позиционными****.*

В общем случае *n-*разрядное положительное число *N* в произвольной системе счисления с основанием *р* представляется суммой вида

*N* *n*−1*a* *pk*, (3.1) *k* 0

∑

*k*

где *ak* — отдельные цифры в записи числа, значения которых равны членам натурального ряда в диапазоне от 0 до (*р* – 1).

При выполнении вычислений цифровыми электронными устройствами используются элементы с двумя устойчивыми со-стояниями. По этой причине в цифровой технике широкое рас-пространение получила позиционная ***двоичная*** система счисле-ния (с основанием 2). В каждом двоичном разряде, получившем название ***бит***, может стоять 1 или 0. Сама же запись числа (дво-ичный код) представляет собой последовательность из единиц и нулей. Чтобы отличить двоичное число от десятичного, будем дополнять его справа суффиксом *В* (*Binary*), как это принято в специальных машинно-ориентированных языках программиро-вания, называемых ассемблерами.

Веса соседних разрядов двоичного кода числа отличаются в два раза, а самый правый разряд (младший) имеет вес 1. Поэтому, например

101101*В* = 1.25 + 0.24 + 1.23 +1.22 + 0.21 + 1.20 = 45. Четыре соседних бита называют ***тетрадой***, группу из 8 бит

называют ***байтом***, а из 16 бит — ***машинным*** ***словом***. Совокуп-ность из 1024 (210) байтов называют килобайтом, из 1024 килобай-тов — мегабайтом, из 1024 мегабайтов — гигабайтом.

1 Гб = 210 Мб = 220 Кб = 230 байт*.*

24

Современные персональные ЭВМ могут хранить в своей памяти на жестких магнитных дисках цифровую информацию объемом в десятки гигабайтов.

Арифметические операции в двоичной системе счисления исключительно просты и легко реализуются аппаратно. Однако при вводе и выводе информации в цифровое устройство она должна быть представлена в более привычной для человека деся-тичной системе счисления. Стремление упростить процедуру пе-ресчета двоичных чисел к десятичному эквиваленту привело к использованию ***двоично-десятичной*** ***системы*** ***счисления*** (BD — Binary Decimals). Она используется в ЭВМ не только в качестве вспомогательной системы счисления при вводе и выводе данных, но и в качестве основной при решении задач, когда в ЭВМ вво-дится и выводится большое количество чисел, а вычислений над ними производится мало. Десятичные числа в двоично-десятич-ной системе счисления кодируются в прямом нормально-взвешенном коде 8-4-2-1, т. е. каждую цифру десятичного числа необходимо заменить соответствующей тетрадой двоичных чи-сел. Например, десятичное число 9531 в двоично-десятичном ко-де представляется машинным словом из четырех тетрад

9531 = 1001 0101 0011 0001.

Записывать двоичные числа большой разрядности утоми-тельно. Поэтому, как правило, они представляются более ком-пактными записями с использованием ***шестнадцатеричной*** сис-темы счисления. В этой системе используют первые десять чле-нов натурального ряда от 0 до 9, а в качестве остальных цифр — первые шесть латинских букв A = 10, B = 11, C = 12, D = 13, E = 14, F = 15. Справа шестнадцатеричное число будем дополнять суф-фиксом Н (*Hexadecimal*)*.*

Перевод двоичного числа в число системы с основанием 16 и наоборот не вызывает затруднений. Для этого исходное двоич-ное число справа налево разбивается на тетрады, а затем содер-жимое каждой из них рассматривается как двоичный код соот-ветствующей цифры шестнадцатеричной системы. Для обратного перехода каждую цифру шестнадцатеричного числа заменяют тетрадой двоичного кода, например:

*N* = 8B5FH = 1000 1011 0101 1111 B*.*

25

Таблица 3.1 — **Соответствие** **чисел** **различных** **систем** **счисления**

|  |  |  |
| --- | --- | --- |
| Десятичное число (*D*) | Шестнадцатеричное число (*H*) | Двоичное число (*B*) |
| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | 0 1 2 3 4 5 6 7 8 9 A B C D E F | 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 |

Для перевода целого числа *Np*, представленного в системе счисления с основанием *р*, в систему счисления с основанием *q* необходимо данное число делить на основание *q* (по правилам системы с основанием *р*) до получения целого остатка, меньшего *q*. Полученное частное снова необходимо разделить на основание *q* и т. д., пока последнее частное не станет меньше *q*. Число *Nq* в новой системе счисления представится в виде упорядоченной по-следовательности остатков в порядке, обратном их получению. Причем цифру старшего разряда дает последнее частное.

***Пример*** ***3.1.*** Перевести десятичное число 15710 в двоичный код, результат проверить.

|  |  |
| --- | --- |
| число | делитель остаток |
| 157 78 39 19 9  4 2 1 0 | 2\_\_\_\_\_\_\_\_\_\_\_\_\_1 (младший разряд) 2\_\_\_\_\_\_\_\_\_\_\_\_\_0  2\_\_\_\_\_\_\_\_\_\_\_\_\_1 2\_\_\_\_\_\_\_\_\_\_\_\_\_1 2\_\_\_\_\_\_\_\_\_\_\_\_\_1  2\_\_\_\_\_\_\_\_\_\_\_\_\_0 15710 = 100111012 2\_\_\_\_\_\_\_\_\_\_\_\_\_0  2\_\_\_\_\_\_\_\_\_\_\_\_\_1 (старший разряд) |

26

Проверка:

100111012 = 1⋅27 + 0⋅26 + 0⋅25 +1⋅24 + 1⋅23 + 1⋅22 + 0⋅21 + 1⋅20 = = 128 + 16 + 8 + 4 +1 =15710.

Для облегчения работы с двоичными кодами желательно знать наизусть десятичные значения чисел 2*n* от n = 0 до n = 14 (табл. 3.2).

Таблица 3.2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *n* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 2 *n* | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |

***Пример*** ***3.2*.** Перевести десятичное число 15710 в восьме-ричный код, результат проверить.

|  |  |
| --- | --- |
| число | делитель остаток |
| 157 19 2  0 | 8\_\_\_\_\_\_\_\_\_\_\_\_\_5 (младший разряд)  8\_\_\_\_\_\_\_\_\_\_\_\_\_3 15710 = 2358 8\_\_\_\_\_\_\_\_\_\_\_\_\_2 (старший разряд) |

Проверка: 2358 = 2⋅82 + 3⋅81 + 5⋅80 = 128 + 24 + 5 = 15710.

***Пример*** ***3.3*.** Перевести десятичное число 15710 в шестна-дцатеричный код, результат проверить.

|  |  |
| --- | --- |
| число | делитель остаток |
| 157 9 0 | 16\_\_\_\_\_\_\_\_\_\_\_\_\_13 (младший разряд) 16\_\_\_\_\_\_\_\_\_\_\_\_\_9 (старший разряд) 15710=9D16 |

Проверка: 9D16 = 9⋅161 + 13⋅160 = 144 + 13 = 15710.

С помощью байта данных можно представить различную информацию:

– целое число без знака (от 0 до 255);

– число от 0 до 99 в двоично-десятичном коде; – машинный код команд микропроцессора;

27

– состояние восьми датчиков;

– двоичное число со знаком в прямом, обратном или ***допол-нительном*** коде *Х*, где *Х* — модуль числа (от 0 до 127), для отображения которого используется семь младших разрядов. Старший разряд — знаковый (0 — для положительных чисел, 1 — для отрицательных).

Пример: прямой код обратный код дополнительный код

+16

0*,* *Х* 00010000 0*,* *Х* 00010000 0*,* *Х* 00010000

–16

1*,* *Х* 10010000 1*,* *Х* 11101111 1*,* *Х* 1 11110000

Прямой, обратный и дополнительный коды положительных чисел совпадают. Для получения дополнительного кода отрица-тельного числа можно проинвертировать код положительного числа и прибавить единицу. Дополнительный код однобайтового числа минус *Х* равен дополнению до 256, т. е. двоичному коду числа 256−*X* . Преобразование дополнительного кода числа в прямой осуществляется по тому же правилу, что прямого в до-полнительный.

***Пример*** ***3.4*.** Записать дополнительный код однобайтового числа минус 100. Для отображения знака используется старший разряд числа.

*Решение*. Запишем двоичный код числа плюс 100: 01100100 Проинвертируем его: 10011011 Прибавим единицу: 10011100

Проверка: 10011100=128+16+8+4=156=256–100.

Ответ: дополнительный код числа минус 100 равен 10011100В.

**3.2** **Таблица** **истинности**

На рис. 3.1, *а* приведено функциональное обозначение циф-рового устройства с тремя входами и одним выходом. Каждый из входных сигналов *А*, *В* и *С* может принимать лишь два значе-ния: 1 и 0*.* Выходной сигнал *F*, который можно рассматривать как

28

логическую функцию входных переменных *А,* *В,* *С,* на каждом их наборе может быть равен 1 или 0.

*А* *В* *С*

≥2 *F*

*а*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *N* | *A* | *B* | *C* | *F* |
| 0 1 2 3 4 5 6 7 | 0 0 0 0 1 1 1 1 | 0 0 1 1 0 0 1 1 | 0 1 0 1 0 1 0 1 | 0 0 0 1 0 1 1 1 |

*А* *В*

|  |  |
| --- | --- |
|  | =1 |
|  |
|  |

*б*

*С*

&

|  |  |
| --- | --- |
| & |  |
|  |

*в*

|  |  |  |
| --- | --- | --- |
|  | 1 |  |
|  |
|  |

*F*

Рис. 3.1 — Функциональное обозначение, таблица истинности и пример построения цифрового устройства

В простейшем случае функция *F*(*A,B,C*) может быть задана словесным описанием. Например, функция *F* равна 1, если все три ее переменные или любая пара из них равны 1, в противном случае *F* *=* 0.

Любая логическая функция может быть задана в виде таб-лицы истинности. На рис. 3.1, *б* представлена таблица истинно-сти для функции трех переменных, описанной выше словесно. Она определена на восьми наборах, которые располагаются в по-рядке нарастания десятичного эквивалента *N* их двоичного кода. В правом столбце указаны значения логической функции *F* на каждом наборе. Задание логической функции таблицей истинно-сти не всегда удобно, так как при большом числе переменных она становится слишком громоздкой. В этом смысле наиболее при-влекателен аналитический способ задания функций в виде так на-зываемых структурных формул, показывающих, какие логиче-

29

ские операции необходимо выполнить над входящими в них пе-ременными, чтобы получить значения данной функции.

**3.3** **Совершенная** **дизъюнктивная** **нормальная** **форма**

По таблице истинности можно составить выражение для логической функции в **СДНФ** (совершенной дизъюнктивной нормальной форме), т. е. в виде суммы логических произведений, соответствующих единичным наборам функции:

*F* *ABC* *ABC* *ABC* *ABC*. (3.2) Выражение (3.2) записано с использованием операций логи-

ческого сложения (дизъюнкции), логического умножения (конъ-юнкции) и логического отрицания (инверсии), которые выполня-ют простейшие логические элементы ИЛИ, И и НЕ соответствен-но. Для каждого единичного набора составляется логическое произведение входных переменных, в которое переменная входит с инверсией при нулевом ее значении на данном наборе. Эти ло-гические произведения объединяются затем знаком логического сложения (+ или ∨).

На рис. 3.2 представлены таблицы истинности и условные графические обозначения двухвходовых логических элементов. Кроме указанных выше, на практике широко используются эле-менты И-НЕ, ИЛИ-НЕ, Исключающее ИЛИ. Логическая функция последнего (функция «неравнозначность» или сумма по модулю два) в СДНФ записывается в виде *A*⊕*B* *AB**AB*.

Логические функции, представляющие собой *дизъюнкции* отдельных членов, каждый из которых есть некоторая функция, содержащая только конъюнкции, называют логическими функ-циями ***дизъюнктивной*** ***нормальной*** ***формы*** (ДНФ), например: *F* *XY* *XZ* . Если же каждый член дизъюнкции нормальной формы от *n* аргументов содержит все эти аргументы, часть ко-торых входит в него с инверсией, а часть — без нее, то такая форма представления функции называется ***совершенной*** ***дизъ-***

***юнктивной*** ***нормальной*** ***формой*** (СДНФ), например: *F* *ABC* *ABC* *ABC* *ABC*.

30

|  |  |
| --- | --- |
| & |  |
|  |

|  |  |
| --- | --- |
|  | & |
|  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер набора | *A* | *B* | *AB* | *AB* | *A+B* | *A+B* | *A* ⊕*B* |
| 0 1 2 3 | 0 0 1 1 | 0 1 0 1 | 0 0 0 1 | 1 1 1 0 | 0 1 1 1 | 1 0 0 0 | 0 1 1 0 |
| Элемент  *А*  *B* *F* | | | И  ЛИ | И-НЕ  ЛА | ИЛИ  1  ЛЛ | ИЛИ-НЕ  1  ЛЕ | *Исключающее* ИЛИ  =1  ЛП |

Рис. 3.2 — Таблицы истинности и условные графические обозначения двухвходовых логических элементов

Каждая конъюнкция этой дизъюнкции включает каждую переменную только один раз в прямом или инверсном виде, об-ращаясь в единицу при определенном наборе значений перемен-ных, и носит название *минтерм*.

Правило перехода от табличного задания логической функ-ции к ее записи в СДНФ (правило записи логической функции по единицам) заключается в следующем:

1. Составить минтермы для строк таблицы истинности, на которых функция *F* равна 1. Если значение переменной в этой строке равно 0, то в минтерме записывается отрицание этой пе-ременной.

2. Записать дизъюнкцию составленных минтермов, которая будет представлять переключательную функцию в СДНФ.