Sistema binário (matemática)

Origem: Wikipédia, a enciclopédia livre.

Ir para: navegação, pesquisa

Nota: *Se procura outros significados para este termo, consulte Sistema binário.*

O **sistema binário** é um sistema de numeração posicional em que todas as quantidades se representam com base em dois numeros, com o que se dispõe das cifras: **zero e um** (0 e 1).

Os computadores digitais trabalham internamente com dois níveis de tensão, pelo que o seu sistema de numeração natural é o sistema binário (aceso, apagado). Com efeito, num sistema simples como este é possível simplificar o cálculo, com o auxílio da lógica booleana. Em computação, chama-se um dígito binário (0 ou 1) de *bit*, que vem do inglês *Binary Digit*. Um agrupamento de 8 bits corresponde a um byte (Binary Term). Um agrupamento de 4 bits é chamado de nibble.

O sistema binário é base para a Álgebra booleana (de George Boole - matemático inglês), que permite fazer operações lógicas e aritméticas usando-se apenas dois dígitos ou dois estados (sim e não, falso e verdadeiro, tudo ou nada, 1 ou 0, ligado e desligado). Toda eletrônica digital e computação está baseada nesse sistema binário e na lógica de Boole, que permite representar por circuitos eletrônicos digitais (portas lógicas) os números, caracteres, realizar operações lógicas e aritméticas. Os programas de computadores são codificados sob forma binária e armazenados nas mídias (memórias, discos, etc) sob esse formato.

Índice

[esconder]

[editar] História

Página do artigo "Explication de l'Arithmétique Binaire", 1703/1705, de Leibniz.
O matemático italiano Pingala apresentou a primeira descrição conhecida de um sistema numérico binário no século III aC.
Um conjunto de 8 trigramas e 64 hexagramas, análogos a números binários com precisão de 3 e 6 bits, foram utilizados pelos antigos chineses no texto clássico I Ching. Conjuntos similares de combinações binárias foram utilizados em sistemas africanos de adivinhação tais como o Ifá, bem como na Geomancia do medievo ocidental.
Uma sistematização binária dos hexagramas do I Ching, representando a sequência decimal de 0 a 63, e um método para gerar tais sequências, foi desenvolvida pelo filósofo e estudioso Shao Yong no século XI. Entretanto, não há evidências que Shao Wong chegou à aritmética binária.
O sistema numérico binário moderno foi documentado de forma abrangente por Gottfried Leibniz no século XVIII em seu artigo "Explication de l'Arithmétique Binaire". O sistema de Leibniz utilizou 0 e 1, tal como o sistema numérico binário corrente nos dias de hoje.
Em 1854, o matemático britânico George Boole publicou um artigo fundamental detalhando um sistema lógico que se tornaria conhecido como Álgebra Booleana. Seu sistema lógico tornou-se essencial para o desenvolvimento do sistema binário, particularmente sua aplicação a circuitos eletrônicos.
Em 1937, Claude Shannon produziu sua tese no MIT que implementava Álgebra Booleana e aritmética binária utilizando circuitos elétricos pela primeira vez na história. Intitulado "A Symbolic Analysis of Relay and Switching Circuits", a tese de Shannon praticamente fundou o projeto de circuitos digitais.

[editar] Operações com binários

[editar] Binários a decimais

Dado um número N, binário, para expressá-lo em decimal, deve-se escrever cada número que o compõe (bit), multiplicado pela base do sistema (base = 2), elevado à posição que ocupa. Uma posição à esquerda da vírgula representa uma potência positiva e à direita uma potência negativa. A soma de cada multiplicação de cada dígito binário pelo valor das potências resulta no número real representado. Exemplo:
1011(binário)
1 × **23** + 0 × **22** + 1 × **21** + 1 × **20** = 11
Portanto, 1011 é 11 em decimal

[editar] Decimais em binários

[editar] Decimais inteiros em binários

Dado um número decimal inteiro, para convertê-lo em binário, basta dividi-lo sucessivamente por 2, anotando o resto da divisão inteira:
12(dec) -> bin

12 / 2 = 6 Resta 0

06 / 2 = 3 Resta 0
03 / 2 = 1 Resta 1
01 / 2 = 0 Resta 1

12(dec) = 1100(bin)
Observe que os números devem ser lidos **de baixo para cima: 1100** é 12 em decimal.
Existe um método muito simples para converter binário em decimal, e vice-versa.

128	64	32	16	8	4	2	1	
0	0	0	0	1	0	1	0	= 10 (2+8=10)
0	0	0	1	1	0	0	0	= 24 (8+16=24)
1	1	0	0	0	0	0	0	= 192 (64+128=192)
1	0	1	1	1	0	1	0	= 186 (2+8+16+32+128=186)

[editar] Decimais fracionários em binários

Exemplo I
0.562510
Parte inteira = 0 10 = 02
Parte fracionária = 0.562510
Multiplica-se a parte fracionária por 2 sucessivamente, até que ela seja igual a zero ou cheguemos na precisão desejada.

fração x 2 = vai-um + fração seguinte
0.5625 x 2 = 1 + 0.1250
0.1250 x 2 = 0 + 0.2500
0.2500 x 2 = 0 + 0.5000
0.5000 x 2 = 1 + 0.0000 <-- nesta linha a fração zerou, finalizamos a conversão

Anotando a seqüência de **vai-um (carry)** na ordem de cima para baixo, temos: **1001**
Portanto, 0.562510 = 0.10012
No entanto, é mais comum nunca zerarmos a fração seguinte da multiplicação.
Neste caso, devemos parar as multiplicações quando atingirmos uma certa precisão desejada.

Exemplo II
67.57510
Parte inteira = 6710 = 10000112
Parte fracionária = 0.57510

fração x 2 = vai-um + fração seguinte
0.5750 x 2 = 1 + 0.1500
0.1500 x 2 = 0 + 0.3000
0.3000 x 2 = 0 + 0.6000 <--- esta fração e suas subseqüentes serão repetidas em breve.
0.6000 x 2 = 1 + 0.2000
0.2000 x 2 = 0 + 0.4000
0.4000 x 2 = 0 + 0.8000
0.8000 x 2 = 1 + 0.6000 <--- a partir daqui repetimos a fração 0.6000 e suas subseqüentes
0.6000 x 2 = 1 + 0.2000

Ou seja, entramos em um ciclo sem fim. Escolhemos uma precisão e finalizamos o processo quando esta precisão for atingida, então na ordem de cima para baixo, temos: **100100112**

[editar] Soma de Binários

0+0=0
0+1=1
1+0=1
1+1=10, ou seja 0 e vai 1* (para somar ao digito imediatamente à esquerda)

Para somar dois números binários, o procedimento é o seguinte:
Exemplo 1:

```
    *
   1100
 +   111
   -----
 = 10011
```

Explicando: Os números binários são base 2, ou seja, há apenas dois algarismos: 0 (zero) ou 1 (um). Na soma de 0 com 1 o total é 1. Quando se soma 1 com 1, o resultado é 2, mas como 2 em binário é 10, o resultado é 0 (zero) e passa-se o outro 1 para a "frente", ou seja, para ser somado com o próximo elemento, conforme assinalado pelo asterisco,como no exemplo acima.
Exemplo 2:

```
   **
   1100
 +  1111
   -----
 = 11011
```

Explicando: Nesse caso acima (exemplo 2), na quarta coluna da direita para a esquerda, nos deparamos com uma soma de 1 com 1 mais a soma do 1 (*) que veio da soma anterior. Quando temos esse caso (1 + 1 + 1), o resultado é 1 e passa-se o outro 1 para frente

[editar] Subtração de Binários

0-0=0
0-1=1 e vai 1* para ser subtraido no digito seguinte
1-0=1
1-1=0
Para subtrair dois números binários, o procedimento é o seguinte:

```
     * ***
   1101110
 -   10111
   -------
 =  1010111
```

Explicando: Quando temos 0 menos 1, precisamos "pedir emprestado" do elemento vizinho. Esse empréstimo vem valendo 2 (dois), pelo fato de ser um número binário. Então, no caso da coluna 0 - 1 = 1, porque na verdade a operação feita foi 2 - 1 = 1. Esse processo se repete e o elemento que cedeu o "empréstimo" e valia 1 passa a valer 0. Os asteriscos marcam os elementos que "emprestaram" para seus vizinhos. Perceba, que, logicamente, quando o valor for zero, ele não pode "emprestar" para ninguém, então o "pedido" passa para o próximo elemento e esse zero recebe o valor de 1.

[editar] Multiplicação de Binários

A multiplicação entre binários é similar à realizada com números decimais. A única diferença está no momento de somar os termos resultantes da operação:

```
       1 0 1 1
      x 1 0 1 0
      ---------
       0 0 0 0
 +    1 0 1 1
 +   0 0 0 0
 + 1 0 1 1
 ---------------
```

```
 = 1 1 0 1 1 1 0
     *
```

Perceba que na soma de 0 e 1 o resultado será 1, mas na soma de 1 com 1, ao invés do resultado ser 2, ele será 0 (zero) e passa-se o 1 para a próxima coluna, conforme assinalado pelo asterisco. Nota que se a soma passar de 2 dígitos, deve-se somar o número em binário correspondente (ex. 4 = 100, 3 =11).

```
        1 1 1
     x  1 1 1
     ---------
        1 1 1
 +     1 1 1
 +    1 1 1
   ---------------
 =  1 1 0 0 0 1
```

No caso, a terceira coluna a soma dá 4 (com mais um da anterior), que adiciona um "1" duas colunas depois (100).

[editar] Divisão de Binários

Essa operação também é similar àquela realizada entre números decimais:

```
  110 |__10__
- 10   11
  --
  010
-  10
   --
   00
```

Deve-se observar somente a regra para subtração entre binários. Nesse exemplo a divisão de 110 por 10 teve como resultado 11.

[editar] Códigos Binários

A conversão de um número decimal no seu equivalente binário é chamada codificação. Um número decimal é expresso como um código binário ou número binário. O sistema numérico binário, como apresentado, é conhecido como código binário puro. Este nome o diferencia de outros tipos de códigos binários.

[editar] Decimal Codificado em Binário

O sistema numérico decimal é fácil de se usar devido à familiaridade. O sistema numérico binário é menos conveniente de se usar pois, nos é menos familiar. É difícil olhar em número binário e rapidamente reconhecer o seu equivalente decimal.
Por exemplo, o número binário 1010011 representa o número decimal 83. É difícil dizer imediatamente, por inspeção do número, qual seu valor decimal. Entretanto, em alguns minutos, usando os procedimentos descritos anteriormente, pode-se prontamente calcular seu valor decimal. A quantidade de tempo que leva para converter ou reconhecer um número binário é uma desvantagem no trabalho com este código, a despeito das numerosas vantagens de "hardware".
Os engenheiros reconheceram este problema cedo, e desenvolveram uma forma especial de código binário que era mais compatível com o sistema decimal. Como uma grande quantidade de dispositivos digitais, instrumentos e equipamentos usam entradas e saídas decimais, este código especial tornou-se muito difundido e utilizado. Esse código especial

é chamado decimal codificado em binário (BCD - binary coded decimal). O código BCD combina algumas das características dos sistemas numéricos binário e decimais.

[editar] Código BCD 8421

O código BCD é um sistema de representação dos dígitos decimais desde 0 até 9 com um código binário de 4 bits. Esse código BCD usa o sistema de pesos posicionais 8421 do código binário puro. Exatamente como binário puro, pode-se converter os números BCD em seus equivalentes decimais simplesmente somando os pesos das posições de bits onde aparece 1.

Decimal	Binário Puro	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

Decimal, Binário Puro e BCD

Observe, entretanto, que existem apenas dez códigos válidos. Os números binários de 4 bits representando os números decimais desde 10 até 15 são inválidos no sistema BCD. Para representar um número decimal em notação BCD substitue-se cada dígito decimal pelo código de 4 bits apropriados.

Por exemplo, o inteiro decimal 834 em BCD é 1000 0011 0100. Cada dígito decimal é representado pelo seu código BCD 8421 equivalente. Um espaço é deixado entre cada grupo de 4 bits para evitar confusão do formato BCD com o código binário puro. Este método de representação também se aplica as frações decimais.

Por exemplo, a fração decimal 0,764 é “0.0111 0110 0100” em BCD. Novamente, cada dígito decimal é representado pelo seu código equivalente 8421, com um espaço entre cada grupo.

Uma vantagem do código BCD é que as dez combinações do código BCD são fáceis de lembrar. Conforme se começa a trabalhar com números binários regularmente, os números BCD tornam-se tão fáceis e automáticos como números decimais. Por esta razão, por simples inspeção da representação BCD de um número decimal pode-se efetuar a conversão quase tão rápido como se já estivesse na forma decimal.

Como exemplo, converter o número BCD no seu equivalente decimal. 0110 0010 1000.1001 0101 0100 = 628,954

O código BCD simplifica a interface Homem-máquina, mas é menos eficiente que o código binário puro. Usam-se mais bits para representar um dado número decimal em BCD que em notação binária pura.

Por exemplo, o número decimal 83 é escrito como 1000 0011. Em código binário puro, usam-se apenas 7 bits para representar o número 83. Em BCD, usam-se 8 bits. O código BCD é ineficiente, pois, para cada bit numa palavra de dado, há usualmente alguma circuitaria digital associada. A circuitaria extra associada com o código BCD custa mais, aumenta a complexidade do equipamento e consome mais energia. Operações aritméticas com números BCD também consomem mais tempo e são mais complexas que aquelas com números binários puros. Com quatro bits de informação binária, você pode representar um total de 24 = 16 estados diferentes ou os números decimais equivalentes desde o 0 até o 15. No sistema BCD, seis destes estados (10-15) são desperdiçados. Quando o sistema numérico BCD é usado, alguma eficiência é perdida, mas aumenta-se o entendimento entre o equipamento digital e o operador humano.

[editar] Conversão Binário para BCD

A conversão de decimal para BCD é simples e direta. Entretanto, a conversão de binário para BCD não é direta. Uma conversão intermediária deve ser realizada primeiro. Por exemplo, o número 1011.01 é convertido no seu equivalente BCD.
Primeiro o número binário é convertido para decimal. 1011.012 =
(1x23)+(0x22)+(1x21)+(1x20)+(0x2-1)+(1x2-2) =8+0+2+1+0+0,25 = 11,2510
Então o resultado decimal é convertido para BCD. 11,2510 = 0001 0001.0010 01012
Para converter de BCD para binário, as operações anteriores são invertidas. Por exemplo, o número BCD 1001 0110.0110 0010 0101 é convertido no seu equivalente binário.

1. O número BCD é convertido para decimal. 1001 0110.0110 0010 0101 = 96,625
2. O resultado decimal é convertido para binário

Inteiro Resto Posição Fração Inteiro Posição
96 ÷ 2 = 48 0 -> LSB 0,625 x 2 = 1,25 = 0,25 1 <- MSB
48 ÷ 2 = 24 0   0,250 x 2 = 0,50 = 0,50 0
24 ÷ 2 = 12 0   0,500 x 2 = 1,00 = 0 0 <- LSB
12 ÷ 2 = 06 0
06 ÷ 2 = 03 0
03 ÷ 2 = 01 1
01 ÷ 2 = 00 1 <- MSB
9610 = 11000002 0,62510 = 0.1012
96,62510 = 9610 + 0,62510= 11000002 + 0.1012 = 1100000.1012
Como o número decimal intermediário contém uma parte inteira e uma parte decimal, cada parte é convertida como visto anteriormente. A soma binária (inteiro mais fração) 1100000.101 é equivalente ao número BCD 1001 0110.0110 0010 0101.
Vários códigos binários são chamados códigos alfanuméricos pois eles são usados para representar caracteres assim como números.

[editar] Código ASCII

O "American Standart Code for Information Interchange" comumente referido como ASCII, é uma forma especial de código binário que é largamente utilizado em microprocessadores e equipamentos de comunicação de dados.
Um novo nome para este código que está se tornando popular é "American National Standart Code for Information" (ANSCII). Entretanto, utilizaremos o termo consagrado, ASCII. É um código binário que usado em transferência de dados entre microprocessadores e seus dispositivos periféricos, e em comunicação de dados por rádio e telefone. Com 7 bits pode-se representar um total de 27 = 128 caracteres diferentes. Estes caracteres compreendem números decimais de 0 até 9, letras maiúsculas e

minúsculas do alfabeto, mais alguns outros caracteres especiais usados para pontuação e controle de dados.
Também chamado ASCII completo, ou ASCII estendido. O código ASCII é mostrado nas tabelas abaixo.

NULL - Null
DLE - Data Link Escape
SOH - Start of Heading
DC1 - Device Control 1
DC2 - Device Control 2
DC3 - Device Control 3
DC4 - Device Control 4
STX - Start of Text
ETX - End of Text
EOT - End of Transmission
ENQ - Enquiry
NAK - Negative Acknowledge
ACK - Acknowledge
SYN - Synchronous Idle
BEL - Bell (audible signal)
ETB - End Transmission Block
BS - Backspace
CAN - Cancel
HT - Horizontal Tabulação (punched card skip)
EM - End of Medium
SUB - Substitute
LF - Line Feed
ESC - Escape
VT - Vertical Tabulation
FS - File Separator
FF - Form Feed
GS - Group Separato
CR - Carriage Return
RS - Record Separator
SO - Shift Out
US - Unit Separator
SI - Shift In
DEL - Delete
SP - Space

bres entiers au-dessous du double du plus haut degré. Car icy, c'est comme si on disoit, par exemple, que 111 ou 7 est la somme de quatre, de deux & d'un. Et que 1101 ou 13 est la somme de huit, quatre & un. Cette proprieté sert aux Essayeurs pour peser toutes sortes de masses avec peu de poids, & pourroit servir dans les monnoyes pour donner plusieurs valeurs avec peu de pieces.

100	4
10	2
1	1
111	7

1000	8
100	
1	1
1101	13

Cette expression des Nombres étant établie, sert à faire tres-facilement toutes sortes d'operations.

Pour *l'Addition* par exemple. ☽

110	6	101	5	1110	14
111	7	1011	11	10001	17
1101	13	10000	16	11111	31

Pour la *Soustraction.*

1101	13	10000	16	11111	31
111	7	1011	11	10001	17
110	6	101	5	1110	14

Pour la *Multiplication.* ⊙

11	3	101	5	101	5
11	3	11	3	101	5
11		101		101	
11		101		1010	
1001	9	1111	15	11001	25

Pour la *Division.*

15	1111	101	5
3	111		
	11		

Et toutes ces operations sont si aisées, qu'on n'a jamais besoin de rien essayer ni deviner, comme il faut faire dans la division ordinaire. On n'a point besoin non-plus de rien apprendre par cœur icy, comme il faut faire dans le calcul ordinaire, où il faut sçavoir, par exemple, que 6 & 7 pris ensemble font 13; & que 5 multiplié par 3 donne 15, suivant la Table *d'une fois un est un*, qu'on appelle Pythagorique. Mais icy tout cela se trouve & se prouve de source, comme l'on voit dans les exemples précedens sous les signes ☽ & ⊙.

TABLE DES NOMBRES.

000000	0
000001	1
000010	2
000011	3
000100	4
000101	5
000110	6
000111	7
001000	8
001001	9
001010	10
001011	11
001100	12
001101	13
001110	14
001111	15
010000	16
010001	17
010010	18
010011	19
010100	20
010101	21
010110	22
010111	23
011000	24
011001	25
011010	26
011011	27
011100	28
011101	29
011110	30
011111	31
100000	32
&c.	&c.