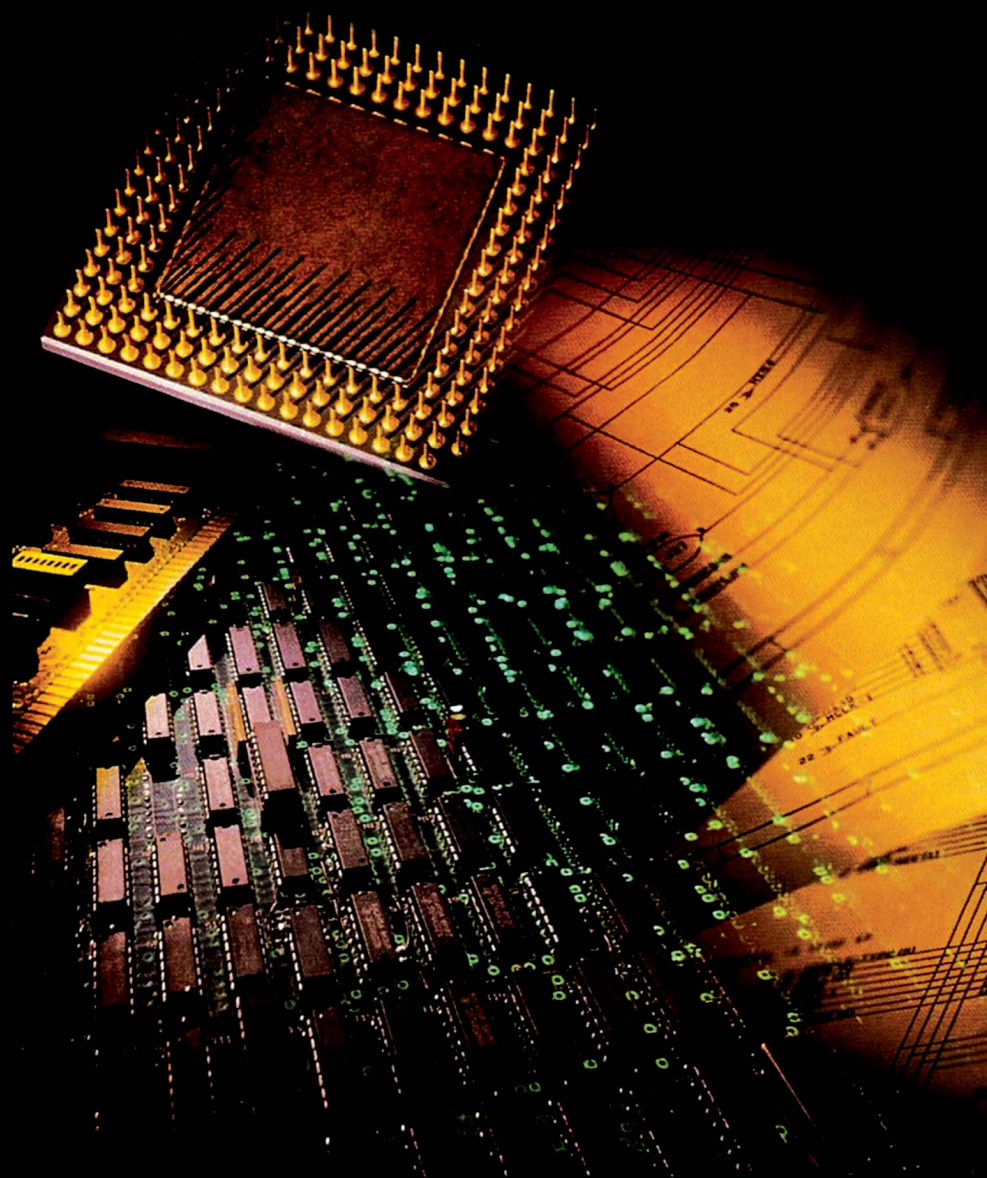


CD com arquivos  
MultiSIM (inglês)

# SISTEMAS DIGITAIS

## FUNDAMENTOS E APLICAÇÕES

9ª edição



# FLOYD



# 2

## SISTEMAS DE NUMERAÇÃO, OPERAÇÕES E CÓDIGOS

### TÓPICOS DO CAPÍTULO

- |     |  |      |   |
|-----|--|------|---|
| 2-1 | Números Decimais                             | 2-7  | Operações Aritméticas com Números Sinalizados |
| 2-2 | Números Binários                             | 2-8  | Números Hexadecimais                          |
| 2-3 | Conversão de Decimal para Binário            | 2-9  | Números Octais                                |
| 2-4 | Aritmética Binária                           | 2-10 | Decimal Codificado em Binário (BCD)           |
| 2-5 | Complementos de 1 e de 2 de Números Binários | 2-11 | Códigos Digitais                              |
| 2-6 | Números Sinalizados                          | 2-12 | Códigos de Detecção e Correção de Erro        |



## OBJETIVOS DO CAPÍTULO

- Revisar o sistema de numeração decimal
- Contar no sistema de numeração binário
- Converter de decimal para binário e vice-versa
- Aplicar operações aritméticas em números binários
- Determinar os complementos de 1 e de 2 de um número binário
- Expressar números binários sinalizados nos formatos sinal-magnitude, complemento de 1, complemento de 2 e ponto flutuante.
- Realizar operações aritméticas com carry de saída sobre números binários sinalizados
- Realizar conversões entre os sistemas de numeração binário e hexadecimal
- Somar números na forma hexadecimal
- Realizar conversões entre os sistemas de numeração binário e octal
- Expressar números decimais na forma de decimal codificado em binário (BCD)
- Somar números BCD
- Realizar conversões entre o sistema binário e o código gray
- Interpretar o código padrão americano para troca de informações (ASCII)
- Explicar como detectar e corrigir erros de código

## TERMOS IMPORTANTES

Termos importantes na ordem em que aparecem no capítulo.

- |                             |                  |
|-----------------------------|------------------|
| ■ LSB                       | ■ Octal          |
| ■ MSB                       | ■ BCD            |
| ■ Byte                      | ■ Alfanumérico   |
| ■ Número de ponto flutuante | ■ ASCII          |
| ■ Hexadecimal               | ■ Paridade       |
|                             | ■ Código Hamming |

## INTRODUÇÃO

O sistema de numeração binário e os códigos digitais são fundamentais para os computadores e para a eletrônica digital em geral. Nesse capítulo, estudaremos o sistema de numeração binário e as suas relações com outros sistemas de numeração como decimal, hexadecimal e octal. Abordaremos as operações aritméticas com números binários com o intuito de fornecer uma base de conhecimento para o entendimento de como os computadores e muitos outros sistemas digitais funcionam. Além disso, abordaremos também o código BCD (decimal codificado em binário), o código Gray e o código ASCII. O método da paridade para detecção de erros em códigos é introduzido e um método de correção de erro é descrito. Os tutoriais sobre o uso de calculadoras em certas operações se baseiam na calculadora gráfica TI-86 e na calculadora TI-36X. Os procedimentos mostrados podem diferir de outros tipos de calculadoras.

WWW. ACESSE O SITE

**Recursos que o ajudarão no estudo deste capítulo estão disponíveis em**

<http://www.prenhall.com/floyd>

## 2-1 NÚMEROS DECIMAIS

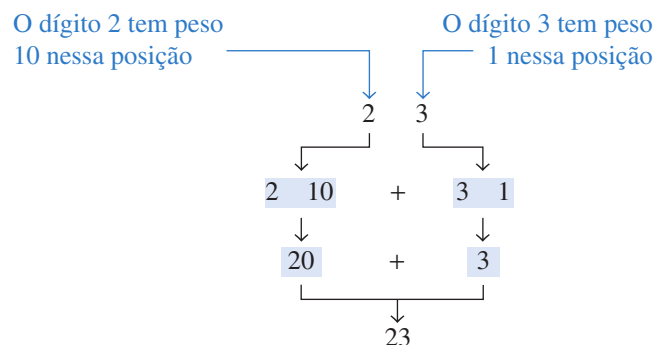
O leitor já tem familiaridade com o sistema de numeração decimal porque deve usar números decimais todos os dias. Embora os números decimais sejam comuns, a sua estrutura de pesos não é freqüentemente compreendida. Nesta seção, a estrutura dos números decimais é revisada. Essa revisão lhe ajudará a compreender mais facilmente a estrutura do sistema de numeração binário, o qual é importante no estudo de computadores e eletrônica digital.

Ao final do estudo desta seção você deverá ser capaz de:

- Explicar por que o sistema de numeração decimal é um sistema em que os dígitos apresentam pesos
- Explicar como potências de dez são usadas no sistema decimal
- Determinar o peso de cada dígito em um número decimal.

O sistema de numeração decimal tem dez dígitos.

No sistema de numeração **decimal**, cada um dos dígitos, de 0 a 9, representa uma certa quantidade. Como sabemos, os dez símbolos (**dígitos**) não nos limita a expressar apenas dez quantidades diferentes porque usamos vários dígitos posicionados adequadamente formando um número para indicar a magnitude (módulo) da quantidade. Podemos expressar quantidades até nove antes de usar mais dígitos; se queremos expressar uma quantidade maior que nove, usamos dois ou mais dígitos, sendo que a posição de cada dígito dentro do número nos diz a magnitude que ele representa. Se, por exemplo, queremos expressar a quantidade vinte e três, usamos (pela suas respectivas posições no número) o dígito 2 para representar a quantidade vinte e o dígito 3 para representar a quantidade três, conforme ilustrado a seguir.



O sistema de numeração decimal tem uma base 10.

A posição de cada dígito em um número decimal indica a magnitude da quantidade representada e pode ser associada a um **peso**. Os pesos para os números inteiros são potências de dez positivas que aumentam da direita para a esquerda, começando com  $10^0 = 1$ .

$$\dots 10^5 \ 10^4 \ 10^3 \ 10^2 \ 10^1 \ 10^0$$

Para números fracionários, os pesos são potências de dez negativas que diminuem da esquerda para a direita começando com  $10^{-1}$ .

O valor de um dígito é determinado por sua posição no número.

$$10^2 \ 10^1 \ 10^0, 10^{-1} \ 10^{-2} \ 10^{-3} \dots$$

↑ vírgula decimal

O valor de um número decimal é a soma dos dígitos após cada um ser multiplicado pelo seu peso, conforme ilustra os Exemplos 2-1 e 2-2.

**EXEMPLO 2-1**

Expresse o número decimal 47 como uma soma dos valores de cada dígito.

**Solução** O dígito 4 tem um peso de 10, que é  $10^1$ , conforme indicado pela sua posição. O dígito 7 tem um peso de 1, que é  $10^0$ , conforme indicado pela sua posição.

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

**Problema relacionado\*** Determine o valor de cada dígito em 939.

\* As respostas estão no final do capítulo.

**EXEMPLO 2-2**

Expresse o número decimal 568,23 como uma soma dos valores de cada dígito.

**Solução** O dígito 5, na parte inteira do número, tem um peso de 100, que é  $10^2$ , o dígito 6 tem um peso de 10, que é  $10^1$ , o dígito 8 tem um peso de 1, que é  $10^0$ , o dígito fracionário 2 tem um peso de 0,1, que é  $10^{-1}$ , e o dígito fracionário 3 tem um peso de 0,01, que é  $10^{-2}$ .

$$\begin{aligned} 568,23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0,1) + (3 \times 0,01) \\ &= \mathbf{500 + 60 + 8 + 0,2 + 0,03} \end{aligned}$$

**Problema relacionado** Determine o valor de cada dígito em 67,924.

**TUTORIAL:  
CALCULADORA****Potência de Dez**

**Exemplo** Determine o valor de  $10^3$ .

			$10^x$	
TI-86	Passo 1	<b>2</b>	<b>^</b>	
	Passo 2	<b>3</b>		$10 \wedge 30$
	Passo 3	<b>ENTER</b>		1000
TI-36X	Passo 1	<b>1</b>	<b>0</b>	<b>y<sup>x</sup></b>
	Passo 2	<b>3</b>	<b>=</b>	1000

**SEÇÃO 2-1  
REVISÃO**

As respostas estão no final do capítulo.

- Qual é o peso que o dígito 7 tem em cada um dos seguintes números?  
(a) 1370    (b) 6725    (c) 7051    (d) 58,72
- Expresse cada um dos seguintes números decimais como uma soma dos produtos obtidos pela multiplicação de cada dígito pelo peso apropriado:  
(a) 51    (b) 137    (c) 1492    (d) 106,58



## 2-2 NÚMEROS BINÁRIOS

O sistema de numeração binário é uma outra forma de representar quantidades. Ele é menos complicado que o sistema decimal porque usa apenas dois dígitos. O sistema decimal com os seus dez dígitos é um sistema de base dez; o sistema binário com seus dois dígitos é um sistema de base dois. Os dois dígitos binários (bits) são 1 e 0. A posição de um 1 ou um 0 em um número binário indica o seu peso, ou valor dentro do número, assim como a posição de um dígito decimal determina o valor daquele dígito. Os pesos em um número binário são baseados em potência de dois.

Ao final do estudo desta seção você deverá ser capaz de:

- Contar em binário
- Determinar o maior número decimal que pode ser representado por um determinado número de bits
- Converter um número binário em um número decimal

### Contagem em Binário

O sistema de numeração binário faz uso de dois dígitos (bits).

Para aprender a contar no sistema binário, primeiro analise como contamos no sistema decimal. Começamos pelo zero e contamos de forma crescente até nove antes de acabar os dígitos. Então começamos com o dígito de outra posição (à esquerda) e continuamos a contagem de 10 até 99. Neste momento, esgotamos todas as combinações de dois dígitos, sendo necessário um terceiro dígito para contar de 100 a 999.

Uma situação semelhante ocorre quando contamos em binário, exceto que temos apenas dois dígitos, denominados *bits*. Começando a contagem: 0, 1. Nesse momento, usamos os dois dígitos, assim incluímos uma nova posição de dígito e continuamos: 10, 11. Esgotamos todas as combinações de dois dígitos, de forma que é necessário uma terceira posição. Com posições para três dígitos podemos continuar a contagem: 100, 101, 110 e 111. Agora precisamos de uma quarta posição de dígito para continuar, e assim por diante. A Tabela 2-1 mostra uma contagem binária de zero a quinze. Observe o padrão de alternância de 1s e 0s em cada coluna.

O sistema de numeração binário tem base 2.

► TABELA 2-1

NÚMERO DECIMAL	NÚMERO BINÁRIO			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Como vemos na Tabela 2-1, são necessários 4 bits para contar de zero a 15. Em geral, com  $n$  bits podemos contar até um número igual a  $2^n - 1$ .

Maior número decimal =  $2^n - 1$

Por exemplo, com cinco bits ( $n = 5$ ) podemos contar de zero a trinta e um.


$$2^5 - 1 = 32 - 1 = 31$$

Com seis bits ( $n = 6$ ) podemos contar de zero a sessenta e três.

$$2^6 - 1 = 64 - 1 = 63$$

Uma tabela de potências de dois é dada no Apêndice A.

O valor de um bit é determinado pela sua posição no número.



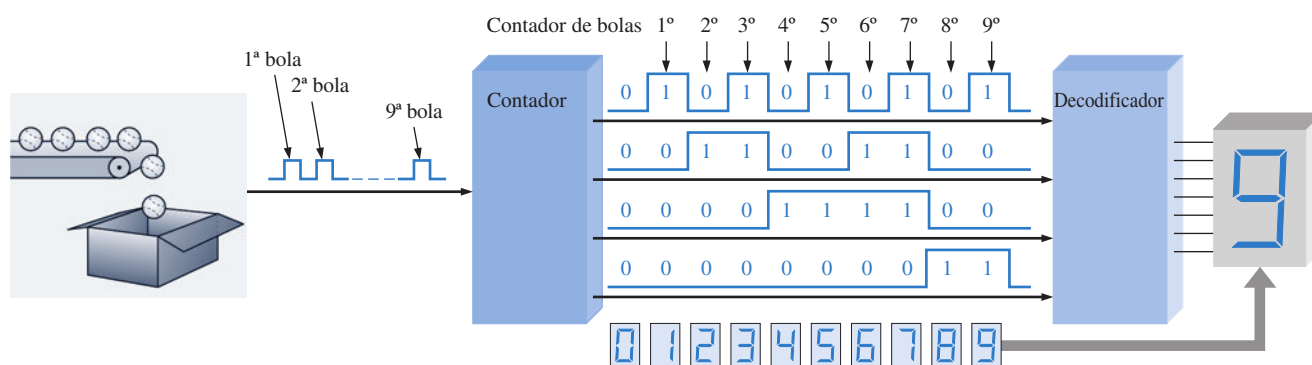
TUTORIAL:  
CALCULADORA

Potência de Dois			
<i>Exemplo</i> Determine o valor de $2^5$ .			
<b>TI-86</b>	<b>Passo 1</b>	2    ^	2 ^ 5 32
	<b>Passo 2</b>	5    ENTER	
<b>TI-36X</b>	<b>Passo 1</b>	2    y <sup>x</sup>	32
	<b>Passo 2</b>	5    =	

## Uma Aplicação

Ao aprender a contar em binário, entenderemos basicamente como os circuitos digitais podem ser usados para contar eventos. Isso pode ser qualquer coisa, desde a contagem de itens em uma linha de montagem até a contagem de operações em um computador. Vamos considerar um exemplo simples da contagem de bola de tênis colocadas em uma caixa a partir de uma correia transportadora. Considere que são colocadas nove bolas em cada caixa.

O contador ilustrado na Figura 2-1 conta os pulsos de um sensor que detecta a passagem de uma bola e gera uma seqüência de níveis lógicos (formas de onda digitais) em cada uma das suas quatro saídas paralelas. Cada conjunto de níveis lógicos representa um número binário de 4 bits (nível ALTO = 1 e nível BAIXO = 0), conforme indicado. À medida que o decodificador recebe essas formas de onda, ele decodifica cada conjunto de quatro bits e converte no número decimal correspondente e mostra num display de 7 segmentos. Quando o contador chega no estado binário 1001, é porque ele contou 9 bolas de tênis, o display mostra o decimal 9 e uma nova caixa é posicionada sob o transportador. Então o contador retorna ao estado zero (0000) e o processo começa novamente. (O número 9 foi usado apenas com o intuito de se ter a simplicidade de um único dígito.)



▲ FIGURA 2-1

Ilustração de uma simples aplicação de contagem binária.

O peso ou o valor de um bit aumenta da direita para a esquerda em um número binário.

## A Estrutura de Pesos dos Números Binários

Um número binário é um número em que os dígitos apresentam pesos. O bit mais à direita é o bit menos significativo (**LSB** – *least significant bit*) em um número inteiro binário e tem um peso de  $2^0 = 1$ . Os pesos aumentam da direita para a esquerda em potências de dois para cada bit. O bit mais à esquerda é o mais significativo (**MSB** – *most significant bit*); seu peso depende do tamanho do número binário.

Números fracionários também podem ser representados em binário colocando os bits à direita da vírgula binária, assim como os dígitos decimais fracionários são colocados à direita da vírgula decimal. O bit mais à esquerda é o MSB em um número binário fracionário e tem um peso de  $2^{-1} = 0,5$ . Os pesos da parte fracionária diminuem da esquerda para a direita por uma potência negativa de dois para cada bit.

A estrutura de pesos de um número binário é a seguinte:

$$2^{n-1} \dots 2^3 2^2 2^1 2^0, 2^{-1} 2^{-2} \dots 2^{-n}$$

↑  
vírgula binária

onde  $n$  é o número de bits a partir da vírgula binária. Portanto, todos os bits à esquerda da vírgula binária têm pesos que são potências positivas de dois, conforme discutido anteriormente para números inteiros. Todos os bits à direita da vírgula binária têm pesos que são potências negativas de dois, ou pesos fracionários.

As potências de dois e os seus pesos decimais equivalentes para um número inteiro binário de 8 bits e um número fracionário binário de 6 bits são mostradas na Tabela 2–2. Observe que o peso dobra para cada potência de dois positiva e que o peso é reduzido à metade a cada potência de dois negativa. Podemos estender facilmente a tabela dobrando o peso da potência de dois positiva mais significativa e reduzindo pela metade o peso da potência de dois negativa menos significativa, por exemplo,  $2^9 = 512$  e  $2^{-7} = 0,00787125$ .

▼ TABELA 2–1

Pesos binários

POTÊNCIAS DE DOIS POSITIVAS NÚMEROS INTEIROS									POTÊNCIAS DE DOIS NEGATIVAS NÚMEROS FRACIONÁRIOS					
$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	$2^{-5}$	$2^{-6}$
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0,5	0,25	0,125	0,0625	0,03125	0,015625

A soma dos pesos de todos os 1s de um número binário resulta no valor decimal.

## Conversão de Binário para Decimal

O valor decimal de um número binário pode ser determinado somando-se os pesos de todos os bits que são 1 e descartando todos os pesos dos bits que são 0.

### EXEMPLO 2–3

Converta o número binário inteiro 1101101 para decimal.

**Solução** Determine o peso de cada bit que for 1 e em seguida calcule o somatório desses pesos para obter o número decimal.

$$\begin{aligned} \text{Peso: } & 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0 \\ \text{Número binário: } & 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \\ 1101101 = & 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ = & 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

**Problema relacionado** Converta o número binário 10010001 para decimal.



### NOTA: COMPUTAÇÃO

Os computadores usam números binários para selecionar posições memória. Cada posição de memória está associada a um único número denominado endereço. Alguns microprocessadores Pentium, por exemplo, têm 32 linhas de endereço, as quais possibilitam selecionar (endereçar)  $2^{32}$  (4.294.967.296) posições únicas de memória.



**EXEMPLO 2-4**

Converta o número binário fracionário 0,1011 para decimal.

**Solução** Determine o peso de cada bit que vale 1 e então some os pesos para obter o número decimal fracionário.

$$\begin{aligned} \text{Peso: } & 2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \\ \text{Número binário: } & 0, 1 \quad 0 \quad 1 \quad 1 \\ 0,1011 = & 2^{-1} + 2^{-3} + 2^{-4} \\ = & 0,5 + 0,125 + 0,0625 = \mathbf{0,6875} \end{aligned}$$

**Problema relacionado** Converta o número binário 10,111 para decimal.

**SEÇÃO 2-2  
REVISÃO**

1. Qual é o maior número decimal que pode ser representado em binário por 8 bits?
2. Determine o peso do bit 1 no número binário 10000.
3. Converta o número binário 10111101,011 para decimal.

**2-3 CONVERSÃO DE DECIMAL PARA BINÁRIO**

Na Seção 2-2, aprendemos como converter um número binário para o seu equivalente decimal. Agora vamos estudar duas formas de converter um número decimal em binário.

Ao final do estudo desta seção você deverá ser capaz de:

- Converter um número decimal para binário usando o método da soma dos pesos
- Converter um número inteiro decimal para binário usando o método da divisão sucessiva por dois
- Converter um número fracionário decimal para binário usando o método da multiplicação sucessiva por dois

**Método da Soma dos Pesos**

Uma forma de determinar o número binário que é equivalente a um dado número decimal é determinar o conjunto dos pesos binários cuja soma é igual ao número decimal. Um jeito fácil de lembrar dos pesos binários é saber que o menor dos pesos é 1, que corresponde a  $2^0$ , e que dobrando esse peso obtemos o próximo peso de maior ordem; assim, uma lista de sete pesos em binário consta os pesos 64, 32, 16, 8, 4, 2, 1 conforme aprendido na seção anterior. O número decimal 9, por exemplo, pode ser expresso como a soma dos pesos binários mostrados a seguir:

$$9 = 8 + 1 \quad \text{ou} \quad 9 = 2^3 + 2^0$$

Colocando 1s nas posições apropriadas,  $2^3$  e  $2^0$ , e 0s nas posições  $2^2$  e  $2^1$ , determina-se o número binário para o decimal 9.

$$\begin{array}{cccc} 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 0 & 0 & 1 \end{array} \quad \text{Número binário para o decimal 9}$$

Para obter um número binário a partir de um número decimal dado, determine os pesos que somados resultam no número decimal.

**EXEMPLO 2-5**

Converta os seguintes números decimais para binário:

- (a) 12      (b) 25      (c) 58      (d) 82

**Solução**

(a)  $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow 1100$

(b)  $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow 11001$

(c)  $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow 111010$

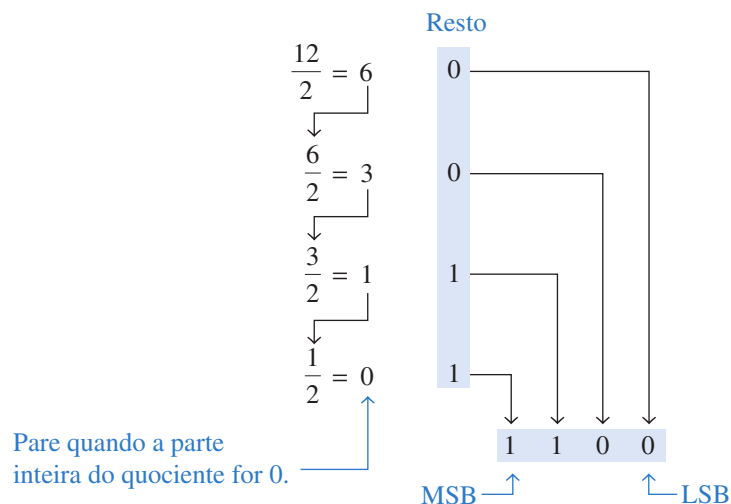
(d)  $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow 1010010$

**Problema relacionado** Converta o número decimal 125 para binário.

**Método da Divisão Sucessiva por 2**

Para obter o número binário que corresponde a um dado número decimal, divida o número decimal por 2 até que o quociente seja 0. Os restos formam o número binário.

Um método sistemático de conversão de números decimais inteiros para o formato binário é o processo de *divisões sucessivas por 2*. Por exemplo, para converter o número decimal 12 para binário, comece dividindo 12 por 2. Em seguida divida cada quociente resultante por 2 até que a parte inteira do quociente seja 0. Os **restos** gerados em cada divisão formam o número binário. O primeiro resto gerado é o LSB (bit menos significativo) no número binário e o último resto gerado é o MSB (bit mais significativo). Esse procedimento é mostrado nos passos a seguir para converter o número decimal 12 em binário.

**EXEMPLO 2-6**

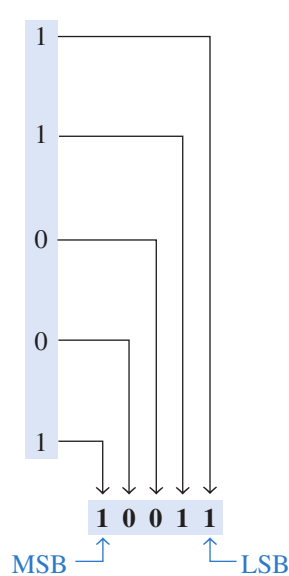
Converta os seguintes números decimais em binário:

- (a) 19      (b) 45

**Solução****(a)**

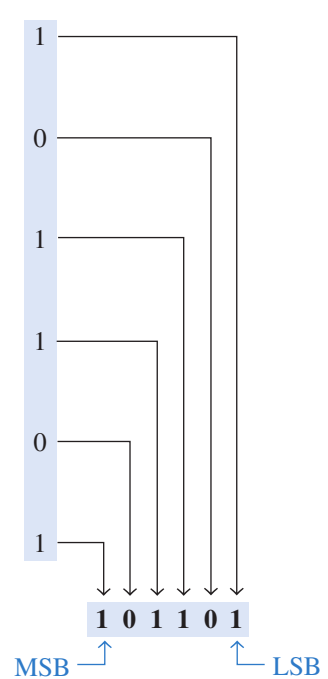
$$\begin{array}{r} 19 \\ 2 \overline{) 9} \\ \underline{9} \phantom{0} \\ 0 \phantom{0} \\ 2 \overline{) 4} \\ \underline{4} \phantom{0} \\ 0 \phantom{0} \\ 2 \overline{) 2} \\ \underline{2} \phantom{0} \\ 0 \phantom{0} \\ 2 \overline{) 1} \\ \underline{1} \phantom{0} \\ 0 \end{array}$$

Resto

**(b)**

$$\begin{array}{r} 45 \\ 2 \overline{) 22} \\ \underline{22} \phantom{0} \\ 0 \phantom{0} \\ 2 \overline{) 11} \\ \underline{11} \phantom{0} \\ 0 \phantom{0} \\ 2 \overline{) 5} \\ \underline{4} \phantom{0} \\ 1 \phantom{0} \\ 2 \overline{) 2} \\ \underline{2} \phantom{0} \\ 0 \phantom{0} \\ 2 \overline{) 1} \\ \underline{1} \phantom{0} \\ 0 \end{array}$$

Resto

**Problema relacionado** Converta o número decimal 39 em binário.**TUTORIAL:  
CALCULADORA****Conversão de um número decimal em número binário****Exemplo** Converta o decimal 57 em binário.

**TI-86**

**Passo 1** BASE 2 1 F3

**Passo 2** 5 7

**Passo 3** F1

**Passo 4** ENTER

**TI-36X**

**Passo 1** BASE 3rd EE

**Passo 2** 5 7

**Passo 3** BIN 3rd X

57 ► Bin

111001b

A-F	TYPE	CONV	BOOL	BIT
► Bin	► Hex	► Oct	► Dec	

**Conversão de Decimal Fracionário em Binário**

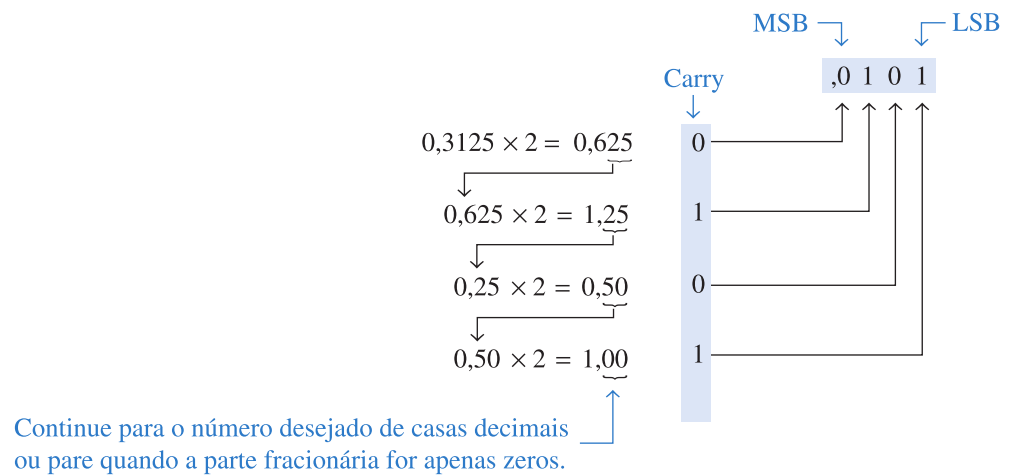
Os Exemplos 2-5 e 2-6 demonstraram conversões de números inteiros. Agora vamos analisar conversões entre números fracionários. Uma forma fácil de lembrar dos pesos da parte fracionária de um número binário é lembrar que o peso do bit mais significativo é  $0,5$ , que equivale a  $2^{-1}$ , e que dividindo qualquer peso por dois obtemos o próximo peso menos significativo; portanto, uma lista de quatro pesos binários fracionários seria  $0,5$ ;  $0,25$ ;  $0,125$ ;  $0,0625$ .

**Soma dos Pesos** O método da soma dos pesos pode ser aplicado a números decimais fracionários, conforme mostra o exemplo a seguir:

$$0,625 = 0,5 + 0,125 = 2^{-1} + 2^{-3} = 0,101$$

Existe um 1 na posição  $2^{-1}$ , um 0 na posição  $2^{-2}$  e um 1 na posição  $2^{-3}$ .

**Multiplicações Sucessivas por 2** Como vimos, números decimais inteiros podem ser convertidos para binário por meio de divisões sucessivas por 2. Decimais fracionários podem ser convertidos para binário por meio de multiplicações sucessivas por 2. Por exemplo, para converter o fracionário decimal 0,3125 para binário, comece multiplicando 0,3125 por 2 e então multiplicar por 2 cada parte fracionária resultante do produto até que o produto seja 0 ou até que o número desejado de casas decimais seja alcançado. Os dígitos de carry, ou **carries**, gerados pela multiplicação formam o número binário. O primeiro carry gerado é o MSB e o último é o LSB. Esse procedimento é ilustrado a seguir:



### SEÇÃO 2-3 REVISÃO

1. Converta cada número decimal a seguir em binário usando o método da soma dos pesos.  
(a) 23    (b) 57    (c) 45,5
2. Converta cada número decimal a seguir em binário usando o método das divisões sucessivas por 2 (multiplicações sucessivas por 2 no caso da parte fracionária):  
(a) 14    (b) 21    (c) 0,375

## 2-4 ARITMÉTICA BINÁRIA

A aritmética binária é essencial em todos os computadores digitais e em muitos outros tipos de sistemas digitais. Para entender os sistemas digitais, temos que saber os fundamentos das operações de soma, subtração, multiplicação e divisão em binário. Esta seção apresenta uma introdução ao assunto, que será estendido em seções posteriores.

Ao final do estudo desta seção você deverá ser capaz de:

- Somar números em binário
- Subtrair números em binário
- Multiplicar números em binário
- Dividir números em binário

## Adição Binária

As quatro regras básicas para a adição de dígitos binários (bits) são:

$0 + 0 = 0$	Resultado: 0; carry: 0
$0 + 1 = 1$	Resultado: 1; carry: 0
$1 + 0 = 1$	Resultado: 1; carry: 0
$1 + 1 = 10$	Resultado: 0; carry: 1

Lembre-se de que, em binário,  $1 + 1 = 10$ , e não 2.

Observe que nas primeiras três regras a soma resulta em um único bit e na quarta regra a soma de dois 1s resulta no número binário dois (10). Quando números binários são somados, o último caso acima gera um resultado 0 em uma dada coluna e um carry de 1 para a próxima coluna à esquerda, conforme ilustrado na adição a seguir ( $11 + 1$ ):

$$\begin{array}{r}
 \text{Carry} \quad \text{Carry} \\
 \begin{array}{r}
 1 \leftarrow 1 \leftarrow 1 \\
 0 \quad 1 \quad 1 \\
 + 0 \quad 0 \quad 1 \\
 \hline
 1 \quad 0 \quad 0
 \end{array}
 \end{array}$$

Na coluna da direita,  $1 + 1 = 0$  com um carry de 1 para a próxima coluna à esquerda. Na coluna do meio,  $1 + 1 + 0 = 0$  com um carry de 1 para a próxima coluna à esquerda. Na última coluna,  $1 + 0 + 0 = 1$ .

Quando existe um carry de 1, temos uma situação na qual três bits estão sendo somados (um bit de cada um dos dois números e um bit de carry). Essa situação é ilustrada a seguir:

bits de carry →

$1 + 0 + 0 = 01$	Resultado: 1; carry: 0
$1 + 1 + 0 = 10$	Resultado: 0; carry: 1
$1 + 0 + 1 = 10$	Resultado: 0; carry: 1
$1 + 1 + 1 = 11$	Resultado: 1; carry: 1

### EXEMPLO 2-7

Efetue as seguintes adições de números binários:

- (a)  $11 + 11$     (b)  $100 + 10$     (c)  $111 + 11$     (d)  $110 + 100$

**Solução** A soma decimal equivalente também é mostrada para referência.

(a)	11	3	(b)	100	4	(c)	111	7	(d)	110	6
	$+11$	$+3$		$+10$	$+2$		$+11$	$+3$		$+100$	$+4$
	<b>110</b>	6		<b>110</b>	6		<b>1010</b>	10		<b>1010</b>	10

**Problema relacionado** Some 1111 com 1100.

## Subtração Binária

As quatro regras básicas para a subtração de bits são:

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$10 - 1 = 1$ sendo o empréstimo igual a 1

Lembre-se de que, em binário,  $10 - 1 = 1$ , e não 9.

Quando subtraímos números, às vezes temos que fazer um empréstimo (borrow) da próxima coluna à esquerda. Em binário um borrow é necessário apenas quando tentamos subtrair 1 de 0. Nesse caso, quando um 1 é obtido como empréstimo da próxima coluna à esquerda, um 10 é criado na coluna que está ocorrendo a subtração e a última das quatro regras básicas apresentadas acima tem que ser aplicada. Os Exemplos 2-8 e 2-9 ilustram a subtração binária: as subtrações decimais equivalentes também são mostradas.

**EXEMPLO 2-8**

Efetue as seguintes subtrações binárias:

(a)  $11 + 01$       (b)  $11 - 10$

**Solução**

(a)	$\begin{array}{r} 11 \\ - 01 \\ \hline 10 \end{array}$	$\begin{array}{r} 3 \\ - 1 \\ \hline 2 \end{array}$
(b)	$\begin{array}{r} 11 \\ - 10 \\ \hline 01 \end{array}$	$\begin{array}{r} 3 \\ - 2 \\ \hline 1 \end{array}$

Nenhum empréstimo foi solicitado neste exemplo. O número binário 01 é o mesmo que 1.

**Problema relacionado** Efetue a subtração:  $100 - 111$ .

**EXEMPLO 2-9**

Efetue a subtração de 011 a partir de 101.

**Solução**

$\begin{array}{r} 101 \\ - 011 \\ \hline 010 \end{array}$	$\begin{array}{r} 5 \\ - 3 \\ \hline 2 \end{array}$
---	---

Vamos analisar exatamente o que foi feito para subtrair os dois números binários visto que um empréstimo foi solicitado. Comece pela coluna à direita.

Coluna da esquerda:

Quando um 1 é emprestado, um 0 é deixado, assim  $0 - 0 = 0$ .

Coluna do meio:

Pega emprestado 1 da próxima coluna, sendo que essa coluna passa a ser 10, então  $10 - 1 = 1$ .

Coluna da direita

$\begin{array}{r} 01 \\ - 011 \\ \hline 010 \end{array}$	$\begin{array}{r} 1 - 1 = 0 \end{array}$
--	--

**Problema relacionado** Efetue a subtração:  $101 - 110$ .

**Multiplicação Binária**

A multiplicação binária de dois bits é realizada da mesma forma que na multiplicação dos dígitos decimais 0 e 1.

As quatro regras básicas para a multiplicação de bits são:

0	0 = 0
0	1 = 0
1	0 = 0
1	1 = 1



A multiplicação é realizada com números binários da mesma maneira que com números decimais. Ela envolve a formação de produtos parciais, deslocamento de cada produto parcial sucessivo uma posição à esquerda, para então somar todos os produtos parciais. O Exemplo 2–10 ilustra o procedimento; as multiplicações decimais equivalentes são mostradas para referência.

**EXEMPLO 2–10**

Realize as seguintes multiplicações binárias:

(a) 11 11      (b) 101 111

**Solução**

(a)	$\begin{array}{r} 11 \\ \times 11 \\ \hline 11 \\ +11 \\ \hline 1001 \end{array}$	$\begin{array}{r} 3 \\ \times 3 \\ \hline 9 \end{array}$	(b)	$\begin{array}{r} 111 \\ \times 101 \\ \hline 111 \\ 000 \\ +111 \\ \hline 100011 \end{array}$	$\begin{array}{r} 7 \\ \times 5 \\ \hline 35 \end{array}$
	Produtos parciais		Produtos parciais		

**Problema relacionado** Efetue a multiplicação: 1011 1010.

**Divisão Binária**

A divisão binária segue os mesmos procedimentos que a divisão decimal, como ilustra o Exemplo 2–11. As divisões decimais equivalentes também são mostradas.

Uma calculadora pode ser usada para realizar operações aritméticas com números binários enquanto a capacidade da calculadora não for excedida.

**EXEMPLO 2–4**

Realize as seguintes divisões binárias:

(a) 110 11      (b) 110 10

**Solução**

(a)	$\begin{array}{r} 10 \\ 11 \overline{)110} \\ \underline{11} \\ 000 \end{array}$	$\begin{array}{r} 2 \\ 3 \overline{)6} \\ \underline{6} \\ 0 \end{array}$	(b)	$\begin{array}{r} 11 \\ 10 \overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array}$	$\begin{array}{r} 3 \\ 2 \overline{)6} \\ \underline{6} \\ 0 \end{array}$
-----	--	---	-----	---	---

**Problema relacionado** Efetue a divisão: 1100 ÷ 100.

**SEÇÃO 2-4  
REVISÃO**

- Realize as seguintes adições binárias:  
(a) 1101 + 1010      (b) 10111 + 01101
- Realize as seguintes subtrações binárias:  
(a) 1101 – 0100      (b) 1001 – 0111
- Realize as operações binárias indicadas:  
(a) 110 111      (b) 1100 011

## 2-5 COMPLEMENTOS DE 1 E DE 2 DE NÚMEROS BINÁRIOS

O complemento de 1 e o complemento de 2 de um número binário são importantes porque eles permitem a representação de números negativos. O método da aritmética do complemento de 2 é geralmente usado em computadores na operação com números negativos.

Ao final do estudo desta seção você deverá ser capaz de:

- Converter um número binário no seu complemento de 1
- Converter um número binário no seu complemento de 2 usando os dois métodos

### Determinação do Complemento de 1

Troque cada bit no número pelo seu complemento de 1.

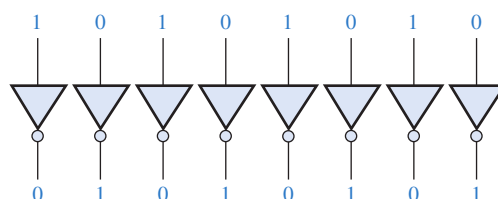
O **complemento de 1** de um número binário é determinado trocando-se todos os 1s por 0s e todos os 0s por 1s, conforme ilustrado a seguir:

1	0	1	1	0	0	1	0	Número binário
↓	↓	↓	↓	↓	↓	↓	↓	
0	1	0	0	1	1	0	1	Complemento de 1

A forma mais simples de obter o complemento de 1 de um número binário com um circuito digital é usar inversores em paralelo (circuitos NOT), conforme mostra a Figura 2-2 para um número binário de 8 bits.

► FIGURA 2-2

Exemplo do uso de inversores para obter o complemento de 1 de um número binário.



### Determinação do Complemento de 2

Some 1 ao complemento de 1 para obter o complemento de 2.

O complemento de 2 de um número binário é determinado somando 1 ao LSB do complemento de 1.

$$\text{complemento de 2} = (\text{complemento de 1}) + 1$$

#### EXEMPLO 2-12

Determine o complemento de 2 de 10110010.

**Solução**

10110010	Número binário
01001101	Complemento de 1
+ 1	Soma-se 1
<b>01001110</b>	Complemento de 2

**Problema relacionado** Determine o complemento de 2 de 11001011.

Um método alternativo para determinar o complemento de 2 de um número binário é:

1. Comece à direita com o LSB e escreva os bits como eles aparecem até o primeiro 1 (inclusive).
2. Tome o complemento de 1 dos bits restantes.

Troque todos os bits à esquerda do bit 1 menos significativo para obter o complemento de 2.

### EXEMPLO 2-13

Determine o complemento de 2 de 10111000 usando o método alternativo.

**Solução**

Complemento de 1 dos bits originais

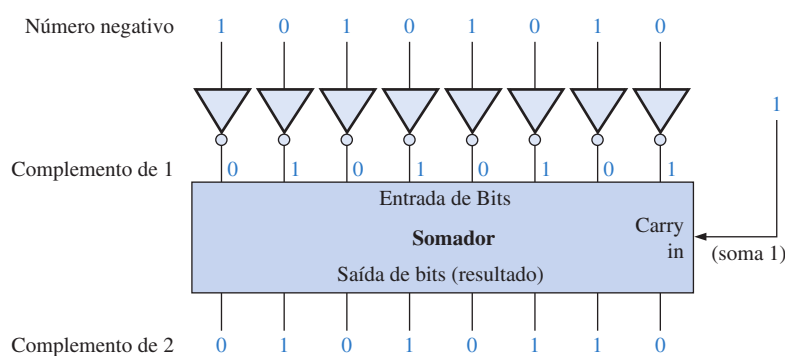
10111000  
01001000

Número binário  
Complemento de 2

Estes bits permanecem os mesmos

**Problema relacionado** Determine o complemento de 2 de 11000000.

O complemento de 2 de um número binário negativo pode ser obtido usando inversores e um somador, conforme indicado na Figura 2-3. Essa figura ilustra como um número de 8 bits pode ser convertido no seu complemento de 2 invertendo primeiro cada bit (tomando o complemento de 1) e em seguida somando 1 ao complemento de 1 com um circuito somador.



◀ FIGURA 2-3

Exemplo da obtenção do complemento de 2 de número binário negativo.

Para converter a partir do complemento de 1 ou de 2 de volta para a forma binária verdadeira (não complementada), usamos os mesmos dois procedimentos descritos anteriormente. Para passar do complemento de 1 de volta para o binário verdadeiro, inverta todos os bits. Para passar do complemento de 2 de volta para a forma binária verdadeira, tome o complemento de 1 do número na forma do complemento de 2 e some 1 ao bit menos significativo.

### SEÇÃO 2-5 REVISÃO

1. Determine o complemento de 1 e cada número binário a seguir:
  - (a) 00011010
  - (b) 11110111
  - (c) 10001101
2. Determine o complemento de 2 de cada número binário a seguir:
  - (a) 00010110
  - (b) 11111100
  - (c) 10010001

## 2-6 NÚMEROS SINALIZADOS

Os sistemas digitais, como o computador, têm que ser capazes de operar com números positivos e negativos. Um número binário sinalizado é constituído de duas informações: sinal e magnitude. O sinal indica se um número é positivo ou negativo e a magnitude é o valor do número. Existem três formas por meio das quais os números inteiros podem ser representados em binário: sinal-magnitude, complemento de 1 e complemento de 2. Dentre esses, a forma do complemento de 2 é a mais importante e a forma sinal-magnitude é a menos usada. Os números fracionários (não-inteiros) e muito grandes ou muito pequenos podem ser expressos na forma de ponto flutuante.

Ao final do estudo desta seção você deverá ser capaz de:

- Expressar números positivos e negativos na forma sinal-magnitude
- Expressar números positivos e negativos na forma do complemento de 1
- Expressar números positivos e negativos na forma do complemento de 2
- Determinar o valor decimal de números binários sinalizados
- Expressar um número binário na forma de ponto flutuante

### ○ Bit de Sinal

O bit mais à esquerda em um número binário sinalizado é o **bit de sinal**, o qual nos diz se o número é positivo ou negativo.

**Um bit de sinal 0 indica um número positivo e um bit de sinal 1 indica um número negativo.**

### Forma Sinal-Magnitude

Quando um número binário sinalizado é representado na forma sinal-magnitude, o bit mais à esquerda é o bit de sinal e os bits restantes são os bits de magnitude. Os bits de magnitude estão na forma de binário verdadeiro (não-complementado) tanto para números positivos quanto para negativos. Por exemplo, o número decimal +25 é expresso como um número binário sinalizado de 8 bits usando a forma sinal-magnitude como a seguir:

00011001  
 Bit de sinal —↑      ↑— Bits de magnitude

O número decimal -25 é expresso como

10011001

Observe que a diferença entre +25 e -25 é apenas o bit de sinal porque os bits de magnitude estão na forma de binário verdadeiro tanto para números positivos quanto negativos.

**Na forma sinal-magnitude, um número negativo tem os mesmos bits de magnitude como o número positivo correspondente mas o bit de sinal é 1 em vez de zero.**

### Forma do Complemento de 1

Números positivos na forma do complemento de 1 são representados da mesma forma que números positivos expressos como sinal-magnitude. Entretanto, os números negativos estão na forma do complemento de 1 do número positivo correspondente. Por exemplo, usando oito bits, o número decimal -25 é expresso como o complemento de 1 de +25 (00011001) como a seguir:

11100110

**Na forma do complemento de 1, um número negativo é o complemento de 1 do número positivo correspondente.**

#### NOTA: COMPUTAÇÃO



Os computadores usam a representação em complemento de 2 para os números inteiros negativos em todas as operações aritméticas. A razão para isso é que a subtração de um número é o mesmo que a adição com o complemento de 2 do número. Os computadores geram o complemento de 2 invertendo os bits e somando 1, usando instruções especiais que geram o mesmo resultado que o somador visto na Figura 2-3.

## Forma do Complemento de 2

Os números positivos na forma do complemento de 2 são expressos da mesma forma que as representações sinal-magnitude e complemento de 1. Os números negativos são expressos em complemento de 2 dos números positivos correspondentes. Exemplificando novamente, usando 8 bits, vamos tomar o número decimal  $-25$  e expressá-lo como complemento de 2 de  $+25$  (00011001).

11100111

**Na forma do complemento de 2, um número negativo é o complemento de 2 do correspondente número positivo.**

### EXEMPLO 2-14

Expresse o número decimal  $-39$  como um número de 8 bits nas formas sinal-magnitude, complemento de 1 e complemento de 2.

**Solução** Primeiro escreva o número de 8 bits para  $+39$ .

00100111

Na *forma sinal-magnitude*,  $-39$  é gerado alterando o bit de sinal para 1 e deixando os bits de magnitude como estavam. O número é

**10100111**

Na *forma do complemento de 1*,  $-39$  é gerado tomando o complemento de 1 de  $+39$  (00100111).

**11011000**

Na *forma do complemento de 2*,  $-39$  é gerado tomando o complemento de 2 de  $+39$  (00100111) como a seguir:

$$\begin{array}{rcl} 11011000 & \text{Complemento de 1} \\ + & 1 & \\ \hline 11011001 & \text{Complemento de 2} \end{array}$$

**Problema relacionado** Expresse  $+19$  e  $-19$  nas formas sinal-magnitude, complemento de 1 e complemento de 2.

## O Valor Decimal de Números Sinalizados

**Sinal-magnitude** Os valores decimais de números positivos e negativos na forma sinal-magnitude são determinados somando os pesos de todos os bits de magnitude que são 1s e ignorando aqueles que são zeros. O sinal é determinado pela análise do bit de sinal.

### EXEMPLO 2-15

Determine o valor decimal do número binário que vem a seguir expresso na forma sinal-magnitude: 10010101.

**Solução** Os sete bits de magnitude e os pesos em potências de dois são:

$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
0	0	1	0	1	0	1

Somando os pesos dos bits que são 1s temos:

$$16 + 4 + 1 = 21$$

O bit de sinal é 1; portanto, o número decimal é  $-21$ .

**Problema relacionado** Determine o valor decimal do número 01110111 dado na forma sinal-magnitude.

**Complemento de 1** Valores decimais de números positivos na forma do complemento de 1 são determinados somando os pesos de todos os bits 1s e ignorando os pesos relativos aos zeros. Os valores decimais de números negativos são determinados atribuindo um valor negativo ao peso do bit de sinal, somando os pesos relativos aos bits 1s e somando 1 ao resultado.

### EXEMPLO 2-16

Determine os valores decimais dos números binários sinalizados expressos em complemento de 1:

(a) 00010111      (b) 11101000

**Solução** (a) Os bits e os respectivos pesos em potências de dois são:

$-2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
0	0	0	1	0	1	1	1

Somando os pesos correspondentes aos bits 1, temos:

$$16 + 4 + 2 + 1 = +23$$

(b) Os bits e os respectivos pesos em potências de dois para o número negativo são mostrados a seguir. Observe que o bit de sinal negativo tem um peso de  $-2^7$  ou  $-128$ .

$-2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1	1	1	0	1	0	0	0

Somando os pesos em que os bits são 1s, temos:

$$-128 + 64 + 32 + 8 = -24$$

Somando 1 ao resultado, o número decimal final é

$$-24 + 1 = -23$$

**Problema relacionado** Determine o valor decimal do número 11101011 expresso na forma do complemento de 1.

**Complemento de 2** Valores decimais de números positivos e negativos na forma do complemento de 2 são determinados somando os pesos das posições de todos os bits 1s e ignorando as posições em que os bits são zeros. O peso do bit de sinal em números negativos é dado com um valor negativo.

### EXEMPLO 2-17

Determine os valores decimais dos números binários sinalizados a seguir expressos na forma do complemento de 2:

(a) 01010110      (b) 10101010



**Solução** (a) Os bits e seus respectivos pesos em potências de dois para números positivos são:

$-2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
0	1	0	1	0	1	1	0

Somando-se os pesos relativos aos bits 1s, temos:

$$64 + 16 + 4 + 2 = +86$$

(b) Os bits e seus respectivos pesos em potências de dois para números positivos são os seguintes. Observe que o bit de sinal negativo tem um peso de  $-2^7$  ou  $-128$ .

$-2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1	0	1	0	1	0	1	0

Somando-se os pesos relativos aos bits 1s, temos:

$$-128 + 32 + 8 + 2 = -86$$

**Problema relacionado** Determine o valor decimal do número 11010111 expresso na forma do complemento de 2.

A partir desses exemplos, podemos ver por que a forma do complemento de 2 é a preferida para representar números inteiros sinalizados: para converter para decimal, é necessário simplesmente somar os pesos independente se o número é positivo ou negativo. O sistema do complemento de 1 requer somar 1 ao resultado da soma dos pesos para números negativos, porém não para números positivos. Além disso, a forma do complemento de 1 não é muito usada porque existem duas representações possíveis para o zero (00000000 ou 11111111).

## Faixa de Números Inteiros que Pode ser Representada

Temos usado números de 8 bits para ilustração porque os grupos de 8 bits são comuns na maioria dos computadores, conhecidos como **byte**. Com um byte ou oito bits, podemos representar 256 números diferentes. Com dois bytes ou dezesseis bits, podemos representar 65.536 números diferentes. Com quatro bytes ou 32 bits, podemos representar  $4,295 \times 10^9$  números diferentes. A fórmula para encontrar o número de combinações diferentes de  $n$  bits é

$$\text{Total de combinações} = 2^n$$

Para números sinalizados na forma do complemento de 2, a faixa de valores para números de  $n$  bits é

$$\text{Faixa} = -(2^{n-1})a + (2^{n-1} - 1)$$

onde existe em cada caso um bit de sinal e  $n - 1$  bits de magnitude. Por exemplo, com quatro bits podemos representar números em complemento de 2 desde  $-(2^3) = -8$  até  $2^3 - 1 = +7$ . De forma similar, com oito bits podemos representar desde  $-128$  até  $+127$ , e com dezesseis bits podemos representar desde  $-32.768$  até  $+32.767$ , e assim por diante.

A faixa da magnitude de um número binário depende do número de bits( $n$ ).

## Números em Ponto Flutuante

Para representar números **inteiros** muito grandes, são necessários muitos bits. Existe também um problema quando números que têm parte inteira e fracionária, como 23,5618, precisam ser representados. O sistema de numeração de ponto flutuante\*, baseado em notação científica, é capaz de representar números muito grandes e muito pequenos sem o aumento do número de bits e também representa números que têm parte inteira e fracionária.

\* N. de T.: A denominação ponto flutuante é mantida por ser de uso comum. O ponto em inglês equivale à vírgula em português. Então, o que vai se mover (flutuar) na representação numérica é a vírgula decimal.

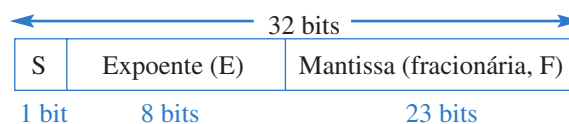
Um **número em ponto flutuante** (também conhecido como *número real*) consiste em duas partes mais um sinal. A **mantissa** é a parte do número em ponto flutuante que representa a magnitude do número. O **expoente** é a parte do número em ponto flutuante que representa o número de casas decimais que a vírgula decimal (ou vírgula binária) é movida.

Um exemplo com número decimal será útil para a compreensão dos conceitos básicos de números em ponto flutuante. Vamos considerar um número decimal que, na forma de inteiro, é 241.506.800. A mantissa dele é 0,2415068 e o expoente é 9. Quando um inteiro é expresso como número em ponto flutuante, ele é normalizado movendo-se a vírgula decimal para a esquerda de todos os dígitos de forma que a mantissa seja um número fracionário e o expoente seja uma potência de dez. O número em ponto flutuante é escrito como a seguir:

$$0,2415068 \cdot 10^9$$

Para números binários em ponto flutuante, o formato é definido pelo padrão 754-1985 da ANSI/IEEE de três formas: *precisão simples*, *precisão dupla* e *precisão estendida*. Todos têm o mesmo formato exceto pelo número de bits. Números em ponto flutuante de precisão simples têm 32 bits, números de precisão dupla têm 64 bits e números de precisão estendida têm 80 bits. Restringiremos nossa discussão ao formato de ponto flutuante de precisão simples.

**Números Binários de Ponto Flutuante de Precisão Simples** No formato padrão para um número binário de precisão simples, o bit de sinal (S) é o bit mais à esquerda, o expoente (E) corresponde aos próximos 8 bits e a mantissa ou parte fracionária (F) inclui os 23 bits restantes, como mostrado a seguir.



Na mantissa ou parte fracionária, entende-se que a vírgula binária está à esquerda dos 23 bits. Efetivamente, existem 24 bits na mantissa porque em qualquer número binário o bit mais à esquerda (o mais significativo) é sempre um 1. Portanto, esse 1 está lá, embora não ocupe uma posição real de um bit.

Os oito bits do expoente representam um *expoente polarizado*, o qual é obtido acrescentando 127 ao expoente real. A finalidade da polarização é permitir números muito grandes ou muito pequenos sem a necessidade de um bit de sinal separado para os expoentes. O expoente polarizado permite uma faixa de valores de expoente de -126 até +128.

Para ilustrar como um número binário é expresso no formato de ponto flutuante, vamos usar o número 1011010010001 como exemplo. Primeiro, ele tem que ser expresso como 1 mais um número binário fracionário movendo a vírgula binária 12 posições para a esquerda e então multiplicar pela potência de dois apropriada.

$$1011010010001 = 1,011010010001 \cdot 2^{12}$$

Admitindo que esse número seja positivo, o bit de sinal (S) é 0. O expoente, 12, é expresso como um expoente polarizado somando-se 127 ( $12 + 127 = 139$ ). O expoente polarizado (E) é expresso como o número binário 10001011. A mantissa é a parte fracionária (F) do número binário 0,011010010001. Como existe um 1 à esquerda da vírgula binária na expressão da potência de dois, não é incluído na mantissa. O número completo representado em ponto flutuante é:

S	E	F
0	10001011	011010010001000000000000

Em seguida, vamos ver como avaliar um número binário que já está no formato de ponto flutuante. A abordagem geral para determinar o valor de um número em ponto flutuante é expressa pela fórmula:

$$\text{Número} = (-1)^S (1 + F)(2^{E-127})$$

#### NOTA: COMPUTAÇÃO



Além da unidade central de processamento (CPU – *central processing unit*), os computadores usam co-processadores para realizar operações matemáticas complexas usando números em ponto flutuante. A finalidade é aumentar a performance liberando a CPU para outras tarefas. O co-processador matemático também é conhecido como unidade de ponto flutuante (FPU – *floating-point unit*).

Para ilustrar, vamos considerar o seguinte número binário em ponto flutuante:

S	E	F
1	10010001	100011100010000000000000

O bit de sinal é 1. O expoente polarizado é 10010001 = 145. Aplicando a fórmula, obtemos:

$$\begin{aligned}\text{Número} &= (-1)^1 (1,10001110001) (2^{145-127}) \\ &= (-1) (1,10001110001) (2^{18}) = -1100011100010000000\end{aligned}$$

Esse número binário em ponto flutuante é equivalente a -407.688 em decimal. Como o expoente pode ser qualquer número entre -126 e +128, pode-se expressar números extremamente grandes e pequenos. Um número de ponto flutuante de 32 bits pode substituir um número binário inteiro de 129 bits. Como o expoente determina a posição da vírgula binária, números que contêm partes inteira e fracionária podem ser representados.

Existem duas exceções para números no formato de ponto flutuante: O número 0,0 é representado por 0s em todas as posições e infinito é representado 1s em todas as posições do expoente e por 0s em todas as posições da mantissa.

### EXEMPLO 2-18

Converta o número decimal  $3,248 \cdot 10^4$  para um número binário no formato de ponto flutuante de precisão simples.

**Solução** Converta o número decimal em binário.

$$3,248 \cdot 10^4 = 32480 = 111111011100000_2 = 1,11111011100000 \cdot 2^{14}$$

O MSB não ocupa a posição de um bit porque ele é sempre um 1. Portanto, a mantissa é o número binário fracionário de 23 bits 11111011100000000000000 e o expoente polarizado é

$$14 + 127 = 141 = 10001101_2$$

O número completo em ponto flutuante é

0	10001101	11111011100000000000000
---	----------	-------------------------

**Problema relacionado** Determine o valor binário do seguinte número em ponto flutuante:

0 10011000 10000100010100110000000

### SEÇÃO 2-6 REVISÃO

1. Expresse o número decimal +9 como um número binário de 8 bits no sistema sinal-magnitude.
2. Expresse o número decimal -33 como um número binário de 8 bits no sistema de complemento de 1.
3. Expresse o número decimal -46 como um número binário de 8 bits no sistema de complemento de 2.
4. Faça uma lista especificando as três partes de um número sinalizado no formato de ponto flutuante.

## 2-7 OPERAÇÕES ARITMÉTICAS COM NÚMEROS SINALIZADOS

Na seção anterior, aprendemos como os números sinalizados são representados de três formas diferentes. Nesta seção, estudaremos como os números sinalizados são somados, subtraídos, multiplicados e divididos. Devido à forma do complemento de 2 para representação de números sinalizados ser a mais usada em computadores e sistemas microprocessados, a abordagem nesta seção se limita a aritmética do complemento de 2. Os processos abordados podem ser estendidos a outros formatos, se necessário.

Ao final do estudo desta seção você deverá ser capaz de:

- Somar números binários sinalizados
- Explicar como os computadores somam seqüências numéricas
- Definir *overflow*
- Subtrair números binários sinalizados
- Multiplicar números binários sinalizados usando o método da adição direta
- Multiplicar números binários sinalizados usando o método dos produtos parciais
- Dividir números binários sinalizados

### Adição

Os dois números de uma adição são **1ª parcela** e **2ª parcela**. O resultado é a **soma**. Existem quatro casos que podem ocorrer quando dois números binários sinalizados são somados.

1. Os dois números são positivos
2. O número positivo com magnitude maior que o número negativo
3. O número negativo com magnitude maior que o número positivo
4. Os dois números são negativos

Vamos analisar um caso de cada vez usando números sinalizados de 8 bits como exemplos. Os números decimais equivalentes são mostrados para referência.

A adição de dois números positivos resulta em um número positivo.

Ambos os números são positivos	00000111	7
	<u>+ 00000100</u>	<u>+ 4</u>
	00001011	11

A soma é positiva estando portanto em binário verdadeiro (não complementado).

**Número positivo com magnitude maior que a do número negativo:**

A adição de um número positivo com um número negativo menor resulta em um número positivo.

	00001111	15
	<u>+ 11110101</u>	<u>+ -6</u>
Carry descartado →	1 00001001	9

O bit de carry final é descartado. A soma é positiva e portanto é um binário verdadeiro (não complementado).

A adição de um número positivo com um número negativo maior, ou a adição de dois números negativos, resulta em um número negativo em complemento de 2.

**Número negativo com magnitude maior que a do número positivo:**

00010000	16
<u>+ 11101000</u>	<u>+ -24</u>
11111000	-8

A soma é negativa e portanto na forma do complemento de 2.

**Ambos os números são negativos:**

	11111011	-5
	<u>+ 11110111</u>	<u>+ -9</u>
Carry descartado →	1 11110010	-14

O bit de carry final é descartado. A soma é negativa e portanto na forma do complemento de 2.

Em um computador, os números negativos são armazenados na forma do complemento de 2. Assim, como podemos ver, o processo de adição é muito simples: *somar os dois números e descartar o bit de carry final*.

**Condição de Overflow** Quando dois números são somados e o número de bits necessário para representar a soma excede o número de bits nos dois números, resulta em um **overflow** (transbordamento de capacidade) conforme indicado por um bit de sinal incorreto. Um overflow pode ocorrer apenas quando os dois números são positivos ou ambos negativos. O exemplo a seguir com números de 8 bits ilustra essa condição.

$$\begin{array}{r}
 01111101 \\
 + 00111010 \\
 \hline
 10110111
 \end{array}
 \qquad
 \begin{array}{r}
 125 \\
 + 58 \\
 \hline
 183
 \end{array}$$

Sinal incorreto —————  
 Magnitude incorreta —————

Nesse exemplo a soma de 183 requer oito bits de magnitude. Como existem sete bits de magnitude nos números (um bit é o sinal), existe um carry no lugar do bit de sinal que produz a indicação de overflow.

**Os Números são Somados Dois de Cada Vez** Agora vamos analisar a adição de uma sequência de números, somados dois de cada vez. Essa operação pode ser realizada somando-se os dois primeiros números, somando em seguida o resultado ao terceiro número, somando outra vez o resultado ao quarto número e assim por diante. É assim que os computadores somam uma sequência de números. A adição de números tomados dois de cada vez é ilustrada no Exemplo 2-19.

### EXEMPLO 2-19

Some os seguintes números sinalizados: 01000100, 00011011, 00001110 e 00010010.

**Solução** As adições decimais equivalentes são dadas como referência.

68	01000100	
+ 27	+ 00011011	Soma dos dois primeiros números
95	01011111	1º subtotal
+ 14	+ 00001110	Soma do 3º número
109	01101101	2º subtotal
+ 18	+ 00010010	Soma do 4º número
127	01111111	Resultado final

**Problema relacionado** Some 00110011, 10111111 e 01100011. Esses números são sinalizados.

## Subtração

A subtração é um caso especial da adição. Por exemplo, a subtração de +6 (o **subtraendo**) de +9 (o **minuendo**) é equivalente à soma de -6 com +9. Basicamente, *a operação de subtração troca o sinal do subtraendo e o soma ao minuendo*. O resultado da subtração é denominado de **diferença**.

A subtração é uma soma com o sinal do subtraendo trocado.

**O sinal de um número binário positivo ou negativo é trocado tomando-se o complemento de 2 dele.**

Por exemplo, quando se toma o complemento de 2 do número positivo 00000100 (+4), obtemos 11111100, que é -4, como mostra a análise da soma dos pesos a seguir:

$$-128 + 64 + 32 + 16 + 8 + 4 = -4$$

Em outro exemplo, quando tomamos o complemento de 2 do número negativo 11101101 (−19), obtemos 00010011, que é +19, conforme a análise da soma dos pesos a seguir:

$$16 + 2 + 1 = 19$$

Como a subtração é simples, uma adição com o subtraendo de sinal trocado, o processo é descrito da seguinte forma:

**Para subtrair dois números sinalizados, tome o complemento de 2 do subtraendo e faça uma soma. Descarte qualquer bit de carry final.**

O Exemplo 2-20 ilustra o processo de subtração.

### EXEMPLO 2-20

Realize cada uma das seguintes subtrações de números sinalizados:

(a)  $00001000 - 00000011$

(b)  $00001100 - 11110111$

(c)  $11100111 - 00010011$

(d)  $10001000 - 11100010$

**Solução** Assim como em outros exemplos, as subtrações decimais equivalentes são dadas para referência.

(a) Neste caso,  $8 - 3 = 8 + (-3) = 5$ .

	00001000	Minuendo (+8)
	+ 1111101	Complemento de 2 do subtraendo (−3)
Carry descartado →	<b>1 0000101</b>	Diferença (+5)

(b) Neste caso,  $12 - (-9) = 12 + 9 = 21$

00001100	Minuendo (+12)
+ 00001001	Complemento de 2 do subtraendo (+9)
00010101	Diferença (+21)

(c) Neste caso,  $-25 - (+19) = -25 + (-19) = -44$

	11100111	Minuendo (−25)
	+ 11101101	Complemento de 2 do subtraendo (−19)
Carry descartado →	<b>1 11010100</b>	Diferença (−44)

(d) Neste caso,  $-120 - (-30) = -120 + 30 = -90$

10001000	Minuendo (−120)
+ 00011110	Complemento de 2 do subtraendo (+30)
<b>10100110</b>	Diferença (−90)

**Problema relacionado** Subtraia 01000111 de 01011000.

### Multiplicação

Os termos de uma multiplicação são o **multiplicando**, o **multiplicador** e o **produto**. Eles são ilustrados na seguinte multiplicação decimal:

8	Multiplicando
× 3	Multiplicador
24	Produto

A operação de multiplicação na maioria dos computadores é realizada usando adição. Como já vimos, a subtração é feita com um somador; agora, veremos como a multiplicação é feita.

A multiplicação é equivalente à adição de um número com ele mesmo um número de vezes igual ao multiplicador.



*Adição direta e produtos parciais* são dois métodos básicos para realizarmos multiplicação usando adição. No método da adição direta, somamos o multiplicando um número de vezes igual ao multiplicador. No exemplo decimal anterior ( $3 \times 8$ ), três multiplicandos são somados:  $8 + 8 + 8 = 24$ . A desvantagem dessa abordagem é que ela se torna muito lenta se o multiplicador for um número grande. Por exemplo, para multiplicar  $75 \times 350$ , temos que somar 350 com ele mesmo 75 vezes. Aliás, esse é o motivo do termo vezes ser usado para significar multiplicação.

Quando dois números binários são multiplicados, os dois números têm que estar na forma verdadeira (não complementados). O método da adição direta é ilustrado no Exemplo 2-21 onde são somados dois números binários de cada vez.

### EXEMPLO 2-21

Multiplique os seguintes números binários sinalizados: 01001101 (multiplicando) e 00000100 (multiplicador) usando o método da adição direta.

**Solução** Como os dois números são positivos, eles estão na forma verdadeira, sendo que o produto será positivo. O valor decimal do multiplicador é 4, de forma que o multiplicando deve ser somado com ele mesmo quatro vezes como mostrado a seguir:

01001101	1ª vez
<u>+ 01001101</u>	2ª vez
10011010	Resultado parcial
<u>+ 01001101</u>	3ª vez
11100111	Resultado parcial
<u>+ 01001101</u>	4ª vez
<b>100110100</b>	Produto

Como o bit de sinal do multiplicando é 0, ele não tem efeito no resultado. Todos os bits do produto são bits de magnitude.

**Problema relacionado** Multiplique 01100001 por 00000110 usando o método da adição direta.

O método do produto parcial talvez seja o mais comumente usado porque ele reflete a forma com que multiplicamos manualmente. O multiplicando é multiplicado por cada dígito do multiplicador começando pelo dígito menos significativo. O resultado da multiplicação do multiplicando por um dígito do multiplicador é denominado de *produto parcial*. Cada produto parcial sucessivo é movido (deslocado) uma posição para a esquerda e quando todos os produtos parciais são gerados, eles são somados para se obter o produto final. Eis um exemplo em decimal.

239	Multiplicando
<u>123</u>	Multiplicador
717	1º produto parcial ( $3 \times 239$ )
478	2º produto parcial ( $2 \times 239$ )
<u>+ 239</u>	3º produto parcial ( $1 \times 239$ )
29.397	Produto final

O sinal do produto de uma multiplicação depende dos sinais do multiplicando e multiplicador de acordo com as seguintes regras:

- Se os sinais são iguais, o produto é positivo.
- Se os sinais são diferentes, o produto é negativo.

Os passos básicos no método dos produtos parciais da multiplicação binária são:

**Passo 1** Determine se os sinais do multiplicando e multiplicador são iguais ou diferentes. Isso determina qual sinal o produto terá.

- Passo 2** Troque qualquer número negativo para a forma verdadeira (não complementado). Como a maioria dos computadores armazena números negativos na forma do complemento de 2, uma operação de complemento de 2 é necessária para mudar um número negativo para a forma verdadeira.
- Passo 3** Começando com o bit menos significativo do multiplicador, gere o produto parcial. Quando o bit do multiplicador for 1, o produto parcial é o mesmo que o multiplicando. Quando o bit do multiplicador for 0, o produto parcial é zero. Desloque cada produto parcial sucessivo um bit para a esquerda.
- Passo 4** Some cada produto parcial sucessivo ao resultado da soma do produto parcial anterior até obter o produto final.
- Passo 5** Se o bit de sinal que foi determinado no passo 1 for negativo, tome o complemento de 2 do produto. Caso seja positivo, deixe o produto na forma verdadeira. Acrescente o bit de sinal ao produto.

**EXEMPLO 2-22**

Multiplique os seguintes números binários sinalizados: 01010011 (multiplicando) e 11000101 (multiplicador).

**Solução** **Passo 1** O bit de sinal do multiplicando é 0 e o bit de sinal do multiplicador é 1. O bit de sinal do produto será 1 (negativo).

**Passo 2** Obtenha o complemento de 2 para colocá-lo na forma verdadeira.

11000101 → 00111011

**Passo 3 e 4** Os procedimentos da multiplicação são registrados a seguir. Observe que apenas os bits de magnitude são usados nesses passos.

1010011	Multiplicando
<u>0111011</u>	Multiplicador
1010011	1º produto parcial
<u>+ 1010011</u>	2º produto parcial
11111001	Soma do 1º com o 2º
<u>+ 0000000</u>	3º produto parcial
011111001	Soma
<u>+ 1010011</u>	4º produto parcial
1110010001	Soma
<u>+ 1010011</u>	5º produto parcial
100011000001	Soma
<u>+ 1010011</u>	6º produto parcial
1001100100001	Soma
<u>+ 0000000</u>	7º produto parcial
1001100100001	Produto final

**Passo 5** Como o sinal do produto é 1, conforme determinado no passo 1, obtenha o complemento de 2 do produto.

1001100100001 → 011001101111

Acrescente o bit de sinal

→ **1** 011001101111

**Problema relacionado** Verifique se a multiplicação está correta convertendo para números decimais e realizando a multiplicação.

## Divisão

Os termos de uma divisão são o **dividendo**, o **divisor** e o **quociente**. Eles são ilustrados no seguinte formato padrão de divisão.

$$\frac{\text{dividendo}}{\text{divisor}} = \text{quociente}$$

A operação de divisão nos computadores é realizada usando subtrações. Como as subtrações são feitas com um somador, a divisão também pode ser realizada com um somador.

O resultado de uma divisão é denominado *quociente*; o quociente é o número de vezes que o divisor ‘cabe dentro’ do dividendo. Isso significa que o divisor pode ser subtraído a partir do dividendo um número de vezes igual ao quociente, conforme ilustrado pela divisão de 21 por 7.

21	dividendo
$\begin{array}{r} - 7 \\ \hline \end{array}$	1ª subtração do divisor
14	1º resto parcial
$\begin{array}{r} - 7 \\ \hline \end{array}$	2ª subtração do divisor
7	2º resto parcial
$\begin{array}{r} - 7 \\ \hline \end{array}$	3ª subtração do divisor
0	Resto zero

Nesse exemplo, o divisor foi subtraído do dividendo três vezes antes que o resto zero fosse obtido. Portanto, o quociente é 3.

O sinal do quociente depende dos sinais do dividendo e do divisor de acordo com as seguintes regras:

- Se os sinais forem iguais, o quociente é positivo.
- Se os sinais forem diferentes, o quociente é negativo.

Quando dois números binários são divididos, os dois números têm que estar na forma verdadeira (não complementados). Os passos básicos no processo de divisão são os seguintes:

- Passo 1** Determine se os sinais do dividendo e do divisor são iguais ou diferentes. Isso determina o sinal do quociente. Inicialize o quociente com zero.
- Passo 2** Subtraia o divisor a partir do dividendo usando a adição do complemento de 2 para obter o primeiro resto parcial e somar 1 ao quociente. Se esse resto parcial for positivo, vá para o passo 3. Caso o resto parcial seja zero ou negativo, a divisão está completa.
- Passo 3** Subtraia o divisor a partir do dividendo e some 1 ao quociente. Se o resultado for positivo, repita a operação para o próximo resto parcial. Se o resultado for zero ou negativo, a divisão está completa.

Continue o divisor a partir do dividendo e os restos parciais até obter um resultado zero ou negativo. Conte o número de vezes que o divisor é subtraído e você terá o quociente. O Exemplo 2–23 ilustra esses passos usando números binários sinalizados de 8 bits.

### EXEMPLO 2–23

Efetue a divisão de 01100100 por 00011001.

**Solução** **Passo 1** Os sinais dos dois números são positivos, de forma que o quociente será positivo. O quociente é inicializado em zero: 00000000.

**Passo 2** Subtraia o divisor a partir do dividendo usando a adição do complemento de 2 (lembre-se que o carry final é descartado).

$$\begin{array}{r} 01100100 \quad \text{Dividendo} \\ + \underline{11100111} \quad \text{Complemento de 2 do divisor} \\ 01001011 \quad \text{1º resto parcial positivo} \end{array}$$

Some 1 ao quociente:  $00000000 + 00000001 = 00000001$ .

**Passo 3** Subtraia o divisor do 1º resto parcial usando a adição do complemento de 2.

$$\begin{array}{r} 01001011 \quad \text{1º resto parcial} \\ + \underline{11100111} \quad \text{Complemento de 2 do divisor} \\ 00110010 \quad \text{2º resto parcial positivo} \end{array}$$

**Passo 4** Subtraia o divisor do 2º resto parcial usando a adição do complemento de 2.

$$\begin{array}{r} 00110010 \quad \text{2º resto parcial} \\ + \underline{11100111} \quad \text{Complemento de 2 do divisor} \\ 00011001 \quad \text{3º resto parcial positivo} \end{array}$$

Some 1 ao quociente:  $00000010 + 00000001 = 00000011$ .

**Passo 5** Subtraia o divisor do 3º resto parcial usando a adição do complemento de 2.

$$\begin{array}{r} 00011001 \quad \text{3º resto parcial} \\ + \underline{11100111} \quad \text{Complemento de 2 do divisor} \\ 00000000 \quad \text{Resto zero} \end{array}$$

Some 1 ao quociente:  $00000011 + 00000001 = \mathbf{00000100}$  (quociente final). O processo está completo.

**Problema relacionado** Verifique se o processo está correto convertendo para números decimais e realizando a divisão.

## SEÇÃO 2-7 REVISÃO

1. Cite os quatro casos de sinalização quando os números são somados.
2. Some  $00100001$  e  $10111100$ .
3. Subtraia  $00110010$  de  $01110111$ .
4. Qual é o sinal do produto quando dois números negativos são multiplicados?
5. Multiplique  $01111111$  por  $00000101$ .
6. Qual é o sinal do quociente quando um número positivo é dividido por um número negativo?
7. Divida  $00110000$  por  $00001100$ .

## 2-8 NÚMEROS HEXADECIMAIS

O sistema de numeração hexadecimal tem dezesseis caracteres; ele é usado principalmente como uma forma compacta de apresentar ou escrever números binários, e é muito fácil realizar conversões entre binário e hexadecimal. Números binários longos são difíceis de serem lidos e escritos porque é fácil omitir ou trocar um bit. Como os computadores entendem apenas 1s e 0s, é necessário usar esses dígitos quando se programa em “linguagem de máquina”. Imagine escrever uma instrução de dezesseis bits para um sistema microprocessado em 1s e 0s. É muito mais eficiente usar hexadecimal ou octal; os números octais são abordados na Seção 2-9. O sistema hexadecimal é bastante usado em aplicações de computador e microprocessador.

Ao final do estudo desta seção você deverá ser capaz de:

- Fazer uma lista dos caracteres hexadecimais
- Contar em hexadecimal
- Converter de binário para hexadecimal
- Converter de hexadecimal para binário
- Converter de hexadecimal para decimal
- Converter de decimal para hexadecimal
- Somar números hexadecimais
- Determinar o complemento de 2 de um número hexadecimal
- Subtrair números hexadecimais

O sistema de numeração **hexadecimal** tem uma base de dezesseis; ou seja, ele é composto de 16 **caracteres numéricos** e alfabéticos. A maioria dos sistemas digitais processa dados binários em grupos que são múltiplos de quatro bits, tornando o número hexadecimal muito conveniente porque cada dígito hexadecimal representa um número binário de 4 bits (conforme vemos na Tabela 2-3).

O sistema de numeração hexadecimal consiste em dígitos de 0 a 9 e letras de A a F.

◀ TABELA 2-3

DECIMAL	BINÁRIO	HEXADECIMAL
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
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

O sistema de numeração hexadecimal é constituído de dez dígitos numéricos e seis caracteres alfabéticos. O uso das letras A, B, C, D, E e F para representar números pode inicialmente parecer estranho, mas tenha em mente que qualquer sistema de numeração é apenas um conjunto de símbolos sequenciais. Se entendermos que quantidades esses símbolos representam, então a forma que os símbolos apresentam é menos importante uma vez que nos acostumamos a usá-los. Usaremos o subscrito 16 para designar números hexadecimais para evitar confusões com os números decimais. Algumas vezes veremos uma letra “h” seguida de um número hexadecimal.

**NOTA: COMPUTAÇÃO**

Com as memórias dos computadores na faixa de gigabytes (GB), especificar um endereço de memória em binário é bastante incômodo. Por exemplo, precisamos de 32 bits para especificar um endereço numa memória de 4 GB. É muito mais fácil expressar um código de 32 bits usando 8 dígitos hexadecimais.

## Contagem em Hexadecimal

Como contar em hexadecimal uma vez atingida a contagem F? Simplesmente inicie uma nova coluna e continue como mostrado a seguir:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31...

Com dois dígitos hexadecimais, podemos contar até  $FF_{16}$ , que corresponde ao decimal 255. Para contar além desse valor, são necessários três dígitos hexadecimais. Por exemplo,  $100_{16}$  equivale ao decimal 256,  $101_{16}$  equivale ao decimal 257 e assim por diante. O maior número hexadecimal de três dígitos é  $FFF_{16}$ , que equivale ao decimal 4095. O maior número hexadecimal de quatro dígitos é  $FFFF_{16}$ , que equivale ao decimal 65.535.

## Conversão de Binário para Hexadecimal

A conversão de um número binário para hexadecimal é um procedimento direto. Simplesmente separe o número binário em grupos de 4 bits começando do bit mais à direita e substituindo cada grupo de 4 bits pelo símbolo hexadecimal equivalente.

### EXEMPLO 2-24

Converta os seguintes números binários para hexadecimal:

(a) 1100101001010111      (b) 111111000101101001

**Solução**

(a)  $\overbrace{1100}^C \overbrace{1010}^A \overbrace{0101}^5 \overbrace{0111}^7 = CA57_{16}$       (b)  $\overbrace{0011}^3 \overbrace{1111}^F \overbrace{1000}^1 \overbrace{1010}^6 \overbrace{1001}^9 = 3F169_{16}$

Dois zeros têm que ser acrescentados no item (b) para completar com 4 bits o grupo à esquerda.

**Problema relacionado** Converta o número binário 1001111011110011100 para hexadecimal.

## Conversão de Hexadecimal para Binário

Hexadecimal é uma forma conveniente de representar números binários.

Para converter um número de hexadecimal para binário, o processo é inverso, sendo que substituímos cada símbolo hexadecimal pelos quatro bits correspondentes.

### EXEMPLO 2-25

Determine os números binários correspondentes aos seguintes números hexadecimais:

(a)  $10A4_{16}$       (b)  $CF8E_{16}$       (b)  $9742_{16}$

**Solução**

(a)  $\begin{array}{cccc} 1 & 0 & A & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1000010100100 \end{array}$       (b)  $\begin{array}{cccc} C & F & 8 & E \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1100111110001110 \end{array}$       (c)  $\begin{array}{cccc} 9 & 7 & 4 & 2 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1001011101000010 \end{array}$

No item (a), considere o MSB precedido de três zeros, formando assim um grupo de 4 bits.

**Problema relacionado** Converta o número hexadecimal 6BD3 em binário.



Deve estar claro que é muito mais fácil lidar com um número hexadecimal do que com o equivalente em binário. Como a conversão é muito fácil, o sistema hexadecimal é amplamente usado na representação de números binários em programação, impressão e displays.

A conversão entre hexadecimal e binário é feita de forma direta e fácil.

### Conversão de Hexadecimal para Decimal

Uma forma de determinar o equivalente decimal de um número hexadecimal é primeiro converter o número hexadecimal em binário e em seguida converter de binário para decimal.

#### EXEMPLO 2-26

Converta o seguinte número hexadecimal em decimal:

(a)  $1C_{16}$       (b)  $A85_{16}$

**Solução** Lembre-se, converta primeiro o número hexadecimal em binário e em seguida converta o número binário para decimal.

(a) 
$$\begin{array}{cc} 1 & C \\ \downarrow & \downarrow \\ 0001 & 1100 \end{array} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$$

(b) 
$$\begin{array}{ccc} A & 8 & 5 \\ \downarrow & \downarrow & \downarrow \\ 1010 & 1000 & 101 \end{array} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$$

**Problema relacionado** Converta o número hexadecimal 6BD para decimal.

Outra forma de converter um número hexadecimal no seu equivalente decimal é multiplicar o valor decimal de cada dígito hexadecimal pelo seu peso e então realizar a soma desses produtos. Os pesos de um número hexadecimal são potências de 16 crescentes (da direita para a esquerda). Para um número hexadecimal de 4 dígitos, os pesos são:

$16^3$	$16^2$	$16^1$	$16^0$
4096	256	16	1

#### EXEMPLO 2-27

Converta os seguintes números hexadecimais em números decimais:

(a)  $E5_{16}$       (b)  $B2F8_{16}$

**Solução** Consultando a Tabela 2-3, vemos que as letras de A a F representam os números decimais de 10 a 15, respectivamente.

(a)  $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$

(b)  $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$   
 $= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$   
 $= 45.056 + 512 + 240 + 8 = \mathbf{45.816}_{10}$

**Problema relacionado** Converta  $60A_{16}$  em decimal.

TUTORIAL:  
CALCULADORA

## Potências de 16

**Exemplo** Determine o valor de  $16^4$ .

**TI-86**    **Passo 1**    1 6 ^

**Passo 2**    4 ENTER

 $16^4$   
65536

**TI-36X**    **Passo 1**    1 6  $y^x$

**Passo 2**    4 =

65536

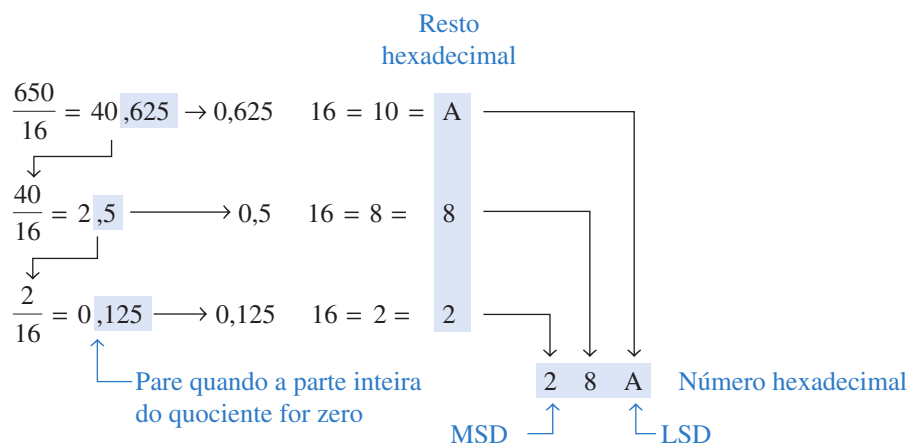
## Conversão de Decimal para Hexadecimal

Divisões sucessivas de um número decimal por 16 produzem o número hexadecimal equivalente, formado pelos restos das divisões. O primeiro resto produzido é o dígito menos significativo (LSD – *least significant digit*). Cada divisão sucessiva por 16 resulta num resto que se torna um dígito no número hexadecimal equivalente. Esse procedimento é similar à divisão sucessiva por 2 usada na conversão de decimal para binário que foi abordada na Seção 2–3. O Exemplo 2–28 ilustra esse procedimento. Observe que quando o quociente tem uma parte fracionária, essa parte é multiplicada pelo divisor para se obter o resto.

## EXEMPLO 2–28

Converta o número decimal 650 em hexadecimal por meio de divisões sucessivas por 16.

## Solução



**Problema relacionado** Converta o decimal 2591 em hexadecimal.

## Adição Hexadecimal

A adição pode ser feita diretamente com números hexadecimais lembrando que os dígitos hexadecimais de 0 a 9 são equivalentes aos dígitos decimais de 0 a 9 e que os dígitos hexadecimais de A a F são equivalentes aos números decimais de 10 a 15. Quando somar dois números hexadecimais, use as regras a seguir. (Os números decimais são indicados pelo subscrito 10).

1. Para qualquer coluna de um problema de adição, pense nos dois dígitos hexadecimais em termos dos seus valores decimais. Por exemplo,  $5_{16} = 5_{10}$  e  $C_{16} = 12_{10}$ .
2. Se a soma dos dois dígitos for  $15_{10}$  ou menos, registre o dígito hexadecimal correspondente.
3. Se a soma dos dois dígitos for maior que  $15_{10}$ , registre o valor da soma que excede a  $16_{10}$  e gere um carry de 1 para a próxima coluna.

Uma calculadora pode ser usada para realizar operações aritméticas com números hexadecimais.

### TUTORIAL: CALCULADORA

#### Conversão de um Número Decimal para Hexadecimal

**Exemplo** Converta o número decimal 650 para hexadecimal.

		BASE		
<b>TI-86</b>	Passo 1	2	1	F3
	Passo 2	6	5	0
	Passo 3	F2		
	Passo 4	ENTER		

		DEC		
<b>TI-36X</b>	Passo 1	3rd	EE	
	Passo 2	6	5	0
	Passo 3	3rd	(	

		HEX		
				28A

650 ► Hex 28Ah

A-F	TYPE	CONV	BOOL	BIT
► Bin	► Hex	► Oct	► Dec	

### EXEMPLO 2-29

Efetue a soma dos seguintes números hexadecimais:

- (a)  $23_{16} + 16_{16}$     (b)  $58_{16} + 22_{16}$     (c)  $2B_{16} + 84_{16}$     (d)  $DF_{16} + AC_{16}$

**Solução**

- (a)  $23_{16}$     coluna da direita:  $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$   
 $16_{16}$     coluna da esquerda:  $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$   
 $\underline{\phantom{00}39_{16}}$
- (b)  $58_{16}$     coluna da direita:  $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$   
 $+ 22_{16}$     coluna da esquerda:  $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$   
 $\underline{\phantom{00}7A_{16}}$
- (c)  $2B_{16}$     coluna da direita:  $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$   
 $+ 84_{16}$     coluna da esquerda:  $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$   
 $\underline{\phantom{00}AF_{16}}$
- (d)  $DF_{16}$     coluna da direita:  $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$   
 $+ AC_{16}$      $27_{10} - 16_{10} = 11_{10} = B_{16}$  com um carry de 1  
 $\underline{\phantom{00}18B_{16}}$     coluna da esquerda:  $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$   
 $\phantom{00}24_{10} - 16_{10} = 8_{10} = 8_{16}$  com um carry de 1

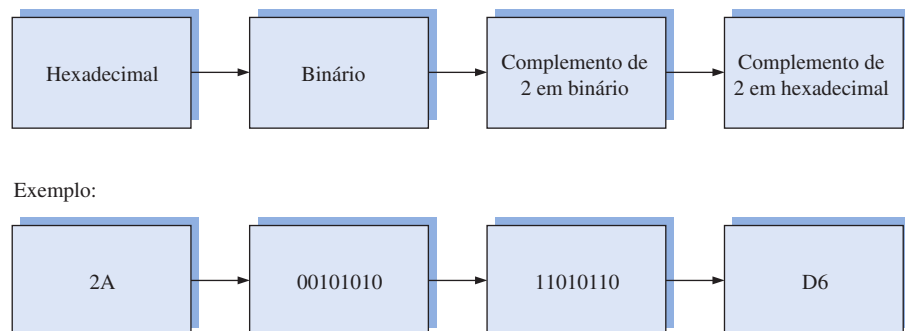
**Problema relacionado** Some  $4C_{16}$  com  $3A_{16}$ .

## Subtração Hexadecimal

Conforme estudamos, o complemento de 2 nos permite subtrair números binários por meio da adição. Como um número hexadecimal pode ser usado para representar um número binário, ele também pode ser usado para representar o complemento de 2 de um número binário.

Existem três formas de obter o complemento de 2 de um número hexadecimal. O método 1 é o mais comum e fácil de ser usado. Os métodos 2 e 3 são alternativos.

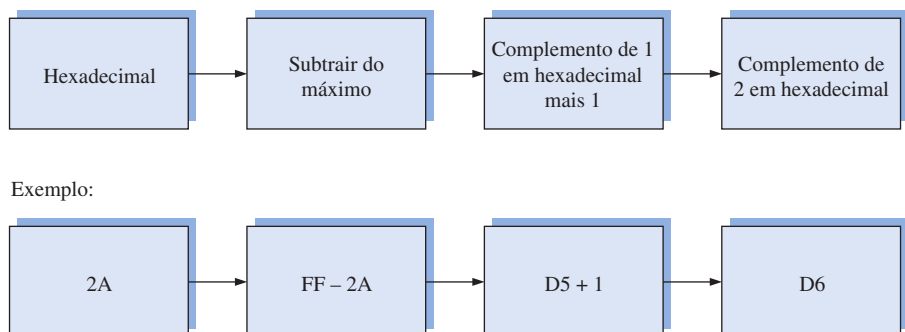
**Método 1** Converta o número hexadecimal para binário. Obtenha o complemento de 2 do número binário. Converta o resultado para hexadecimal. Esses passos estão ilustrados na Figura 2-4.



▲ FIGURA 2-4

Obtenção do complemento de 2 de um número hexadecimal, Método 1.

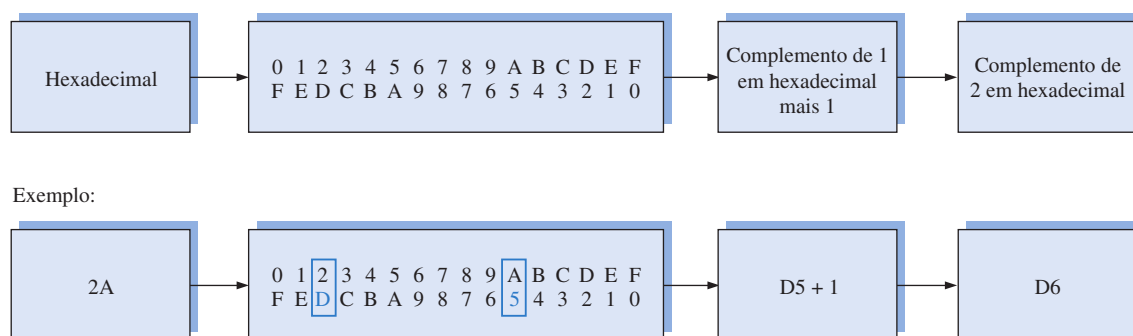
**Método 2** Subtraia o número hexadecimal do maior número hexadecimal (com a mesma quantidade de dígitos) e some 1. Isso está ilustrado na Figura 2-5.



▲ FIGURA 2-5

Obtenção do complemento de 2 de um número hexadecimal, Método 2.

**Método 3** Escreva a sequência dos números hexadecimais de um dígito. Escreva a sequência inversa abaixo da sequência direta. O complemento de 1 de cada dígito hexa é o dígito diretamente abaixo dele. Some 1 ao número resultante para obter o complemento de 2. Esses procedimentos estão ilustrados na Figura 2-6.



▲ FIGURA 2-6

Obtenção do complemento de 2 de um número hexadecimal, Método 3.

**EXEMPLO 2-30**

Efetue a subtração dos seguintes números hexadecimais:

(a)  $84_{16} + 2A_{16}$       (b)  $C3_{16} + 0B_{16}$

**Solução**

(a)  $2A_{16} = 00101010$

O complemento de 2 de  $2A_{16} = 11010110 = D6_{16}$  (usando o Método 1)

$$\begin{array}{r} 84_{16} \\ + D6_{16} \\ \hline \cancel{1}5A_{16} \end{array}$$

Soma  
Desconsiderar o carry, como na adição do complemento de 2

A diferença é  $5A_{16}$ .

(b)  $0B_{16} = 00001011$

O complemento de 2 de  $0B_{16} = 11110101 = F5_{16}$  (usando o Método 1)

$$\begin{array}{r} C3_{16} \\ + F5_{16} \\ \hline \cancel{1}B8_{16} \end{array}$$

Soma  
Desconsiderar o carry

A diferença é  $B8_{16}$ .**Problema relacionado** Subtraia  $173_{16}$  de  $BCD_{16}$ .**SEÇÃO 2-8  
REVISÃO**

1. Converta os seguintes números binários em hexadecimais.

(a)  $10110011$       (b)  $110011101000$

2. Converta os seguintes números hexadecimais em binários.

(a)  $57_{16}$       (b)  $3A5_{16}$       (c)  $F80B_{16}$

3. Converta  $9B30_{16}$  em decimal.

4. Converta o número decimal 573 em hexadecimal.

5. Some os seguintes números hexadecimais diretamente:

(a)  $18_{16} + 34_{16}$       (b)  $3F_{16} + 2A_{16}$

5. Efetue as seguintes subtrações de números hexadecimais.

(a)  $75_{16} - 21_{16}$       (b)  $94_{16} - 5C_{16}$

## 2-9 NÚMEROS OCTAIS

Assim como o sistema de numeração hexadecimal, o sistema de numeração octal proporciona uma forma conveniente de expressar números binários e códigos. Entretanto, ele é usado menos freqüentemente que o sistema hexadecimal em conjunção com computadores e microprocessadores para expressar quantidades binárias para fins de entrada e saída.

Ao final do estudo desta seção você deverá ser capaz de:

- Escrever os dígitos do sistema de numeração octal
- Converter de octal para decimal
- Converter de decimal para octal
- Converter de octal para binário
- Converter de binário para octal

O sistema de numeração **octal** é composto de oito dígitos, os quais são:

0, 1, 2, 3, 4, 5, 6, 7

Para contar acima de 7, inicie uma nova coluna e continue:

10, 11, 12, 13, 14, 15, 16, 17, 20, 21,...

O sistema de numeração octal tem uma base 8.

A contagem em octal é similar à contagem em decimal, exceto que os dígitos 8 e 9 não são usados. A fim de distinguir os números octais dos números decimais ou hexadecimais, usamos o subscrito 8 para indicar que o número em questão é octal. Por exemplo,  $15_8$  em octal é equivalente a  $13_{10}$  em decimal e D em hexadecimal. Algumas vezes podemos encontrar as letras “o” ou “Q” após o número, indicando que ele é octal.

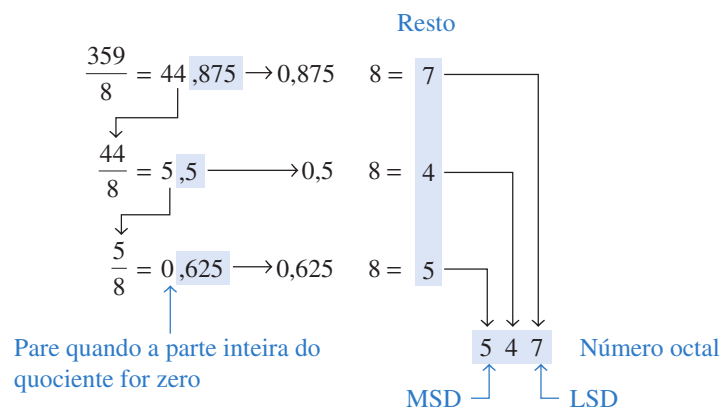
### Conversão de Octal para Decimal

Como o sistema de numeração octal tem uma base de oito, cada posição sucessiva de um dígito é uma potência crescente de oito, começando pela coluna mais à direita com  $8^0$ . O cálculo de um número octal em termos do seu equivalente decimal é realizado multiplicando-se cada dígito pelo seu peso e somando os produtos, conforme ilustrado a seguir para o número  $2374_8$ .

$$\begin{array}{r}
 \text{Peso: } 8^3 \ 8^2 \ 8^1 \ 8^0 \\
 \text{Número octal: } 2 \ 3 \ 7 \ 4 \\
 2374_8 = (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\
 = (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\
 = 1024 + 192 + 56 + 4 = 1276_{10}
 \end{array}$$

### Conversão de Decimal para Octal

Um método de conversão de um número decimal para octal é o da divisão sucessiva por 8, similar ao método usado na conversão de números decimais para binário ou para hexadecimal. Para mostrar como se faz, vamos converter o número decimal 359 para octal. Cada divisão sucessiva por 8 resulta num resto que se torna um dígito do número octal equivalente. O primeiro resto gerado é o dígito menos significativo (LSD).



TUTORIAL:  
CALCULADORA

## Conversão de um Número Decimal para Octal

**Exemplo** Converta o decimal 439 para octal.

**TI-86**

**Passo 1**      **2** **1** **F3**

**Passo 2**      **4** **3** **9**

**Passo 3**      **F3**

**Passo 4**      **ENTER**

BASE

**TI-36X**

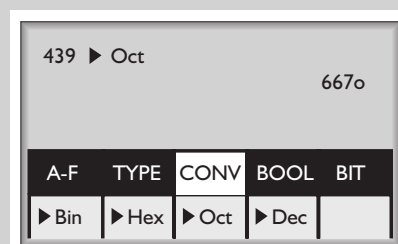
**Passo 1**      **3rd** **EE**

**Passo 2**      **4** **3** **9**

**Passo 3**      **3rd** **)**

DEC

OCT



667

## Conversão de Octal para Binário

Como o dígito octal pode ser representado por 3 bits, é muito fácil converter de octal para binário. Cada dígito octal é representado por três bits, conforme mostra a Tabela 2-4.

O sistema octal é uma forma conveniente de representar números binários, mas não é tão usado como o hexadecimal.

▼ TABELA 2-4

Conversão de octal para binário

DÍGITO OCTAL	0	1	2	3	4	5	6	7
BINÁRIO	000	001	010	011	100	101	110	111

Para converter um número octal para binário, simplesmente substitua cada dígito octal pelos três bits apropriados.

## EXEMPLO 2-3 I

Converta cada um dos seguintes números octais para binário:

(a)  $13_8$     (b)  $25_8$     (c)  $140_8$     (d)  $7526_8$ 

**Solução**

(a)  $\begin{matrix} 1 & 3 \\ \downarrow & \downarrow \\ 001 & 011 \end{matrix}$     (b)  $\begin{matrix} 2 & 5 \\ \downarrow & \downarrow \\ 010 & 101 \end{matrix}$     (c)  $\begin{matrix} 1 & 4 & 0 \\ \downarrow & \downarrow & \downarrow \\ 001 & 100 & 000 \end{matrix}$     (d)  $\begin{matrix} 7 & 5 & 2 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 111 & 101 & 010 & 110 \end{matrix}$

**Problema relacionado** Converta cada um dos números binários obtidos neste exemplo para decimal e verifique que cada valor está de acordo com o valor decimal correspondente ao número octal.

## Conversão de Binário para Octal

A conversão de binário para octal é a operação inversa da conversão de octal para binário. O procedimento é o seguinte: comece pelo grupo de três bits mais à direita e, percorrendo os grupos de bits da direita para a esquerda, converta cada grupo no seu dígito octal correspondente. Caso o grupo mais à esquerda não tiver três bits, acrescente um ou dois zeros para completar o grupo. Esses zeros à esquerda não afetam o valor do número binário.

**EXEMPLO 2-32**

Converta cada número binário a seguir no seu equivalente em octal:

- (a) 110101    (b) 101111001    (c) 100110011010    (d) 11010000100

**Solução**

$$\begin{array}{c} \text{(a)} \quad 110101 \\ \downarrow \quad \downarrow \\ 6 \quad 5 = 65_8 \end{array}$$

$$\begin{array}{c} \text{(b)} \quad 101111001 \\ \downarrow \quad \downarrow \quad \downarrow \\ 5 \quad 7 \quad 1 = 571_8 \end{array}$$

$$\begin{array}{c} \text{(c)} \quad 100110011010 \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ 4 \quad 6 \quad 3 \quad 2 = 4632_8 \end{array}$$

$$\begin{array}{c} \text{(d)} \quad 011010000100 \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ 3 \quad 2 \quad 0 \quad 4 = 3204_8 \end{array}$$

**Problema relacionado** Converta o número binário 1010101000111110010 para octal.

**SEÇÃO 2-9**  
**REVISÃO**

1. Converta os seguintes números octais em decimais:

- (a)  $73_8$     (b)  $125_8$

2. Converta os seguintes números decimais em octais:

- (a)  $98_{10}$     (b)  $163_{10}$

3. Converta os seguintes números octais em binários:

- (a)  $46_8$     (b)  $723_8$     (c)  $5624_8$

4. Converta os seguintes números binários em octais:

- (a) 110101111    (b) 1001100010    (c) 1011111001

**2-10**    **DECIMAL CODIFICADO EM BINÁRIO (BCD)**

Decimal codificado em binário (BCD – *binary coded decimal*) é uma forma de expressar cada dígito decimal com um código binário. Existem apenas dez grupos de códigos no sistema BCD, de forma que é muito fácil converter decimal em BCD. Como preferimos ler e escrever em decimal, o código BCD provê uma excelente interface com o sistema binário. Exemplos de tais interfaces são as entradas do teclado e leituras digitais.

Ao final do estudo desta seção você deverá ser capaz de:

- Converter cada dígito decimal em BCD
- Expressar números decimais em BCD
- Converter de BCD para decimal
- Somar números em BCD

**O Código 8421**

Em BCD, 4 bits representa cada dígito decimal.

O código 8421 é um tipo de código **BCD** (decimal codificado em binário). Decimal codificado em binário significa que cada dígito decimal, de 0 a 9, é representado por um código binário de quatro bits. A designação 8421 indica os pesos binários dos quatro bits ( $2^3$ ,  $2^2$ ,  $2^1$ ,  $2^0$ ). A facilidade de conversão entre números em código 8421 e números decimais é a principal vantagem desse código. Tudo o que precisamos fazer é lembrar as dez combinações binárias que representam os dez dígitos conforme mostra a Tabela 2-5. O código 8421 é o código BCD predominante, e quando nos referirmos a BCD, queremos dizer que o código é o 8421, a menos que seja relatado o contrário.



DÍGITO DECIMAL	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

TABELA 2-5

Conversão de decimal para BCD.

**Códigos inválidos** O leitor já deve ter percebido que, com quatro bits, podemos representar dezesseis números (de 0000 a 1111), porém, no código 8421, apenas dez deles são usados. As seis combinações do código que não são usadas (1010, 1011, 1100, 1101, 1110 e 1111) são inválidas no código BCD 8421.

Para expressar qualquer número decimal em BCD, substitua cada dígito decimal pelo código apropriado de 4 bits, conforme mostra o Exemplo 2-33.

**EXEMPLO 2-33**

Converta em BCD cada um dos seguintes números decimais:

- (a) 35      (b) 98      (c) 170      (d) 2469

**Solução**

- (a)  $\begin{array}{cc} 3 & 5 \\ \downarrow & \downarrow \\ 0011 & 0101 \end{array}$       (b)  $\begin{array}{cc} 9 & 8 \\ \downarrow & \downarrow \\ 1001 & 1000 \end{array}$
- (c)  $\begin{array}{ccc} 1 & 7 & 0 \\ \downarrow & \downarrow & \downarrow \\ 0001 & 0111 & 0000 \end{array}$       (d)  $\begin{array}{cccc} 2 & 4 & 6 & 9 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 0010 & 0100 & 0110 & 1001 \end{array}$

**Problema relacionado** Converta o número decimal 9673 em BCD.

Determinar um número decimal a partir de um número BCD é igualmente fácil. Comece pelo bit mais à direita separando o código em grupos de 4 bits. Em seguida, escreva o dígito representado por cada grupo de quatro bits.

**EXEMPLO 2-34**

Converta cada um dos seguintes códigos BCD em decimal:

- (a) 10000110      (b) 001101010001      (c) 1001010001110000

**Solução**

- (a)  $\begin{array}{cc} 1000 & 0110 \\ \downarrow & \downarrow \\ 8 & 6 \end{array}$       (b)  $\begin{array}{ccc} 0011 & 0101 & 0001 \\ \downarrow & \downarrow & \downarrow \\ 3 & 5 & 1 \end{array}$       (c)  $\begin{array}{cccc} 1001 & 0100 & 0111 & 0000 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 9 & 4 & 7 & 0 \end{array}$

**Problema relacionado** Converta o código BCD 10000010001001110110 em decimal.

**Adição em BCD**

BCD é um código numérico e pode ser usado em operações aritméticas. A adição é a operação mais importante porque as outras três operações (subtração, multiplicação e divisão) podem ser realizadas através da adição. Eis como dois números BCD são somados:

**Passo 1** Some os dois números BCD, usando as regras de adição binária dadas na Seção 2-4.

**Passo 2** Se um resultado de 4 bits for igual ou menor que 9, ele é um número BCD válido.

**Passo 3** Se um resultado de 4 bits for maior que 9, ou se um carry de saída de um grupo de 4 bits for gerado, ele será um resultado inválido. Some 6 (0110) ao resultado de 4 bits para “pular” os seis estados inválidos e retornar ao código 8421. Se ocorrer um carry quando 6 for somado, simplesmente acrescente o carry ao próximo grupo de 4 bits.

O Exemplo 2–35 ilustra adições BCD nas quais o resultado de cada coluna de 4 bits é igual ou menor que 9, sendo esses resultados números BCD válidos. O Exemplo 2–36 ilustra o procedimento no caso de resultados inválidos (maiores que 9 ou com carry).

### EXEMPLO 2–35

Some os seguintes números BCD:

- (a) 0011 + 0100                      (b) 00100011 + 00010101  
(c) 10000110 + 00010011          (d) 010001010000 + 010000010111

**Solução** As adições de números decimais são mostradas para comparação.

<p>(a)</p> $\begin{array}{r} 0011 \quad 3 \\ + 0100 \quad + 4 \\ \hline 0111 \quad 7 \end{array}$	<p>(b)</p> $\begin{array}{r} 0010 \quad 0011 \quad 23 \\ + 0001 \quad 0101 \quad + 15 \\ \hline 0011 \quad 1000 \quad 38 \end{array}$
<p>(c)</p> $\begin{array}{r} 1000 \quad 0110 \quad 86 \\ + 0001 \quad 0011 \quad + 13 \\ \hline 1001 \quad 1001 \quad 99 \end{array}$	<p>(d)</p> $\begin{array}{r} 0100 \quad 0101 \quad 0000 \quad 450 \\ + 0100 \quad 0001 \quad 0111 \quad + 417 \\ \hline 1000 \quad 0110 \quad 0111 \quad 867 \end{array}$

Observe que em cada caso o resultado é apenas uma coluna de 4 bits que não excede a 9, sendo números BCD válidos.

**Problema relacionado** Some os seguintes números BCD: 1001000001000011 + 0000100100100101.

### EXEMPLO 2–36

Some os seguintes números BCD:

- (a) 1001 + 0100                      (b) 1001 + 1001  
(c) 00010110 + 00010101          (d) 01100111 + 01010011

**Solução** As adições de números decimais são mostradas para comparação.

<p>(a)</p> $\begin{array}{r} 1001 \\ + 0100 \\ \hline 1101 \\ + 0110 \\ \hline 0001 \quad 0011 \end{array}$ <p style="text-align: center;">↓                  ↓</p> <p style="text-align: center;">1                  3</p>	<p>Número BCD inválido (&gt;9)</p> <p>Somar 6</p> <p>Número BCD válido</p>	$\begin{array}{r} 9 \\ + 4 \\ \hline 13 \end{array}$
<p>(b)</p> $\begin{array}{r} 1001 \\ + 1001 \\ \hline 1 \quad 0010 \\ + 0110 \\ \hline 0001 \quad 1000 \end{array}$ <p style="text-align: center;">↓                  ↓</p> <p style="text-align: center;">1                  8</p>	<p>Inválido por causa do carry</p> <p>Somar 6</p> <p>Número BCD válido</p>	$\begin{array}{r} 9 \\ + 9 \\ \hline 18 \end{array}$

(c)

0001	0110		16
+ 0001	0101		+ 15
0010	1011		31

+ 0110

0011	0001	
↓	↓	
3	1	

O grupo da direita é inválido (>9) e o grupo da esquerda é válido. Some 6 ao código inválido. Some o carry, 0001, ao próximo grupo. Número BCD válido

(d)

0110	0111		67
+ 0101	+ 0011		+ 53
1011	1010		120
+ 0110	0110		
0001	0010	0000	
↓	↓	↓	
1	2	0	

Ambos os grupos são inválidos (>9) 120. Some 6 aos dois grupos. Número BCD válido

**Problema relacionado** Some os seguintes números BCD: 01001000 + 00110100.

## SEÇÃO 2-10 REVISÃO

1. Qual é o peso binário de cada bit 1 nos números BCD a seguir?  
(a) 0010    (b) 1000    (c) 0001    (d) 0100
2. Converta os seguintes números decimais em números BCD:  
(a) 6    (b) 15    (c) 273    (d) 849
3. Quais números decimais são representados por cada código BCD?  
(a) 10001001    (b) 001001111000    (c) 000101010111
4. Na adição BCD, quando um resultado de 4 bits é inválido?

## 2-11 CÓDIGOS DIGITAIS

Muitos códigos específicos são usados em sistemas digitais. Acabamos de estudar o código BCD; agora vamos analisar outros códigos. Alguns códigos são estritamente numéricos, como o BCD, e outros são alfanuméricos; ou seja, são usados para representar números, letras, símbolos e instruções. Os códigos apresentados nesta seção são o Gray e o ASCII.

Ao final do estudo desta seção você deverá ser capaz de:

- Explicar a vantagem do código Gray
- Converter o código Gray em binário
- Usar o código ASCII

### O Código Gray

Os bits do **código Gray** não têm peso e ele não é um código aritmético; ou seja, não existem pesos associados às posições dos bits. A característica importante do código Gray é que *ele apresenta uma mudança de um único bit quando se passa de uma palavra do código para a seguinte na sequência*. Essa propriedade é importante em muitas aplicações, como em codificadores de posição de eixo, onde a suscetibilidade a erros aumenta com o número de mudanças de bits entre números adjacentes em uma sequência.

A alteração de um único bit, característica do código Gray, minimiza a chance de erro.

A Tabela 2–6 apresenta uma lista de um código Gray de 4 bits para os números decimais de 0 a 15. Os números binários são mostrados na tabela para referência. Assim como os números binários, o código Gray pode ter qualquer número de bits. Observe a mudança de apenas um bit entre as palavras do código Gray. Por exemplo, quando se passa do decimal 3 para o 4, o código Gray muda de 0010 para 0110, enquanto que o código binário muda de 0011 para 0100, uma mudança de três bits. Neste exemplo de código Gray, o único bit que muda é o terceiro bit da esquerda para a direita; os outros permanecem inalterados.

► TABELA 2–6

Código Gray de 4 bits

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

**Conversão de Binário para Código Gray** Às vezes é útil fazer conversões entre o código binário e o código Gray. As regras a seguir explicam como converter de um número binário para uma palavra em código Gray.

1. O bit mais significativo (mais à esquerda) no código Gray é o mesmo que o correspondente MSB no número binário.
2. Da esquerda para a direita, some cada par de bits adjacentes no código binário para obter o próximo bit do código Gray. Descarte os carries.

Por exemplo, a conversão do número binário 10110 para o código Gray é a seguinte:

1	–	+	→	0	–	+	→	1	–	+	→	1	–	+	→	0	Binário
↓				↓				↓				↓				↓	
1				1				1				0				1	Gray

O código Gray é 11101.

**Conversão de Código Gray para Binário** Para converter de código Gray para binário, usamos um método similar que, entretanto, apresenta algumas diferenças. As seguintes regras são aplicadas:

1. O bit mais significativo (mais à esquerda) no código binário é o mesmo que o correspondente bit no código Gray.
2. Some cada bit do código binário gerado ao bit do código Gray na próxima posição adjacente. Descarte os carries.

Por exemplo, a conversão do código Gray 11011 para binário é a seguinte:

1		1		0		1		1	Gray
↓	+	↓	+	↓	+	↓	+	↓	
1		0		0		1		0	Binário

O número binário é 10010.

**EXEMPLO 2-37**

- (a) Converta o número binário 11000110 para código Gray.  
 (b) Converta o código Gray 10101111 para binário.

**Solução** (a) De binário para código Gray:

1 + → 1 + → 0 + → 0 + → 0 + → 1 + → 1 + → 0  
 ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓  
 1 0 1 0 0 1 0 1

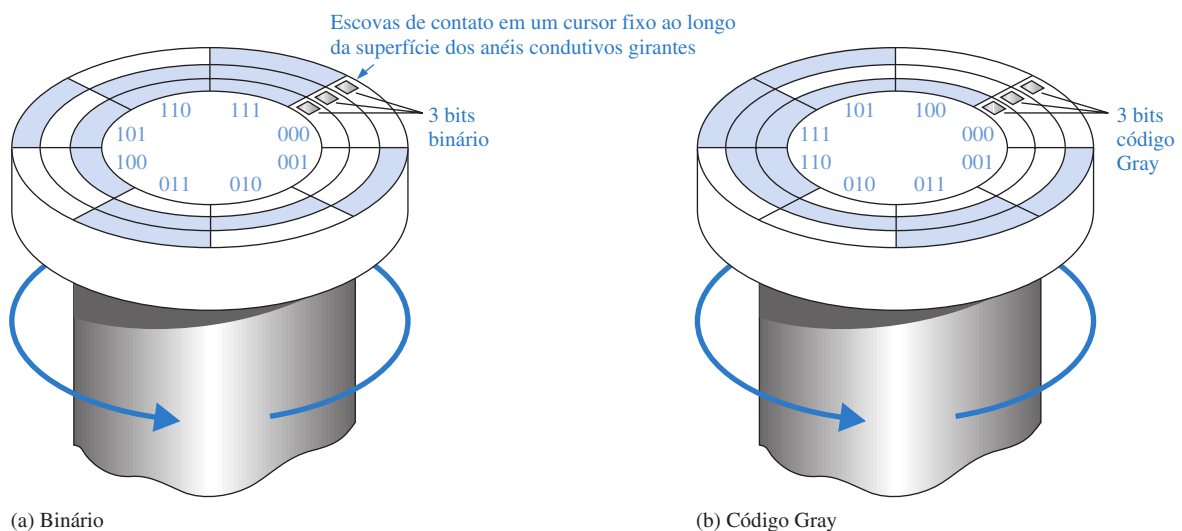
(b) De código Gray para binário:

1 ↓ + 0 ↓ + 1 ↓ + 0 ↓ + 1 ↓ + 1 ↓ + 1 ↓ + 1 ↓  
 1 1 0 0 1 0 1 0

- Problema relacionado** (a) Converta o binário 101101 para código Gray.  
 (b) Converta o código Gray 100111 para binário.

**Uma Aplicação**

Um diagrama simplificado de um mecanismo codificador de posição de eixo de três bits é mostrado na Figura 2-7. Basicamente, existem três anéis que são segmentados em oito setores. Quanto mais setores existirem, maior será a precisão do posicionamento que pode ser representada, porém estamos usando neste exemplo apenas oito para fins de ilustração. Cada setor de cada anel é fixado a uma tensão de nível alto ou a uma tensão de nível baixo para representar 1s e 0s. Um 1 é indicado por um setor colorido e um 0 por um setor branco. À medida que o eixo gira no sentido anti-horário ao longo dos 360°, os oito setores se movem sob três escovas produzindo uma saída binária de 3 bits que indicam a posição do eixo.



▲ FIGURA 2-7

Uma ilustração simplificada de como o código Gray resolve o problema de erro em codificadores da posição de eixo.

Na Figura 2–7(a), os setores são organizados de forma a produzir um padrão binário direto, gerando na passagem das escovas a sequência 000, 001, 010, 011, e assim por diante. Quando as escovas estão nos setores coloridos, a saída é 1 e quando elas estão nos setores brancos, a saída é 0. Se uma escova estiver um pouco a frente das outras durante a transição de um setor para o próximo, podem ocorrer erros na saída. Considere o que acontece quando as escovas estão no setor 111 e entram no setor 000. Se a escova relativa ao MSB estiver um pouco adiantada, a posição 011 seria indicada incorretamente em vez da transição direta de 111 para 000. Nesse tipo de aplicação, é praticamente impossível manter um alinhamento mecânico preciso para todas as escovas; portanto, alguns erros podem ocorrer em muitas das transições entre setores.

O código Gray é usado para eliminar o problema de erro que é inerente ao código binário. Conforme mostra a Figura 2–7(b), o código Gray garante que apenas um bit mude entre setores adjacentes. Isso significa que mesmo que as escovas não tenham um alinhamento preciso, nunca ocorrerá erros na transição. Por exemplo, vamos considerar novamente o que acontece quando as escovas estão no setor 111 e se movem para o próximo setor, 101. as duas únicas saídas possíveis durante a transição são 111 e 101, não importando como as escovas estão alinhadas. Uma situação similar ocorre na transição de cada um dos outros setores.

### Códigos Alfanuméricos

Para nos comunicarmos, não usamos apenas números, mas também letras e outros símbolos. No sentido rigoroso, os códigos **alfanuméricos** representam números e caracteres alfabéticos (letras). Entretanto, a maioria desses códigos representa também outros caracteres, como símbolos e várias instruções necessárias para transmissão de informações.

Um código alfanumérico representa pelo menos 10 dígitos decimais e 26 letras do alfabeto, num total de 36 itens. Esse número requer seis bits em cada representação de código, pois cinco bits não são suficientes ( $2^5 = 32$ ). Existe um total de 64 combinações de seis bits, sendo que 28 combinações do código não são usadas. Obviamente, em muitas aplicações, outros símbolos além de números e letras são necessários para uma comunicação completa. Precisamos de espaços, pontos, vírgulas, ponto-e-vírgulas, interrogação, etc. Precisamos também de instruções que digam ao sistema receptor o que fazer com a informação. Com códigos de seis bits de extensão, podemos operar com números decimais, o alfabeto e outros 28 símbolos. Isso deve ter despertado no leitor a idéia da necessidade de um código alfanumérico básico. O ASCII é o código alfanumérico mais comum e é abordado a seguir.

### ASCII

**ASCII** é a abreviação de American Standard Code for Information Interchange (Código Padrão Americano para Troca de Informações). O código ASCII (pronunciado “askii”), é um código alfanumérico aceito universalmente e usado na maioria dos computadores e outros equipamentos eletrônicos. A maioria dos teclados de computadores é padronizada com o código ASCII. Quando digitamos uma letra, um número ou um comando de controle, o código ASCII correspondente é enviado para o computador.

O ASCII tem 128 caracteres e símbolos representados por um código de 7 bits. Na verdade, o código ASCII pode ser considerado um código de 8 bits com o MSB sempre 0. Esse código de 8 bits vai de 00 até 7F em hexadecimal. Os primeiros 32 caracteres ASCII são comandos não-gráficos que não são impressos ou mostrados e são usados apenas para fins de controle. São exemplos desses caracteres: “nulo”, “próxima linha”, “início de texto” e “escape”. Os outros caracteres são símbolos gráficos que podem ser impresso ou mostrados e incluem as letras do alfabeto (minúsculas e maiúsculas), os dez dígitos decimais, sinais de pontuação e outros símbolos normalmente usados.

A Tabela 2–7 é uma lista do código ASCII mostrando a representação de cada caractere e símbolo em decimal, hexadecimal e binário. A seção à esquerda da tabela é uma lista dos nomes dos 32 caracteres de controle (de 00 a 1F em hexadecimal). Os símbolos gráficos são apresentados no restante da tabela (de 20 a 7F em hexadecimal).

#### NOTA: COMPUTAÇÃO



O teclado de um computador tem um microprocessador dedicado que constantemente escaneia (“lê”) o circuito do teclado para detectar quando uma tecla foi pressionada e liberada. Uma única varredura é gerada pelo software do computador representando aquela tecla em particular. O código de varredura é então convertido em código alfanumérico (ASCII) para ser usado pelo computador.

TABELA 2-7  
Código Padrão Americano para Troca de Informações (ASCII)

CARACTERES DE CONTROLE				SÍMBOLO GRÁFICO											
NOME	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA
NUL	0	0000000	00	espaço	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	\$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	-	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C		124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

**EXEMPLO 2-38**

Determine o código binário ASCII que é inserido pelo teclado do computador quando a seguinte linha de comando em BASIC é digitada. Expresse também cada código em hexadecimal.

20 PRINT "A=";X

**Solução** O código ASCII para cada símbolo é encontrado na Tabela 2-7.

Símbolo	Binário	Hexadecimal
2	0110010	32 <sub>16</sub>
0	0110000	30 <sub>16</sub>
Espaço	0100000	20 <sub>16</sub>
P	1010000	50 <sub>16</sub>
R	1010010	52 <sub>16</sub>
I	1001001	49 <sub>16</sub>
N	1001110	4E <sub>16</sub>
T	1010100	54 <sub>16</sub>
Espaço	0100000	20 <sub>16</sub>
"	0100010	22 <sub>16</sub>
A	1000001	41 <sub>16</sub>
=	0111101	3D <sub>16</sub>
"	0100010	22 <sub>16</sub>
;	0111011	3B <sub>16</sub>
X	1011000	58 <sub>16</sub>

**Problema relacionado** Determine a sequência de códigos ASCII necessária para expressar a seguinte linha de comando de um programa em hexadecimal:

80 INPUT Y

**Os Caracteres de Controle do Código ASCII** Os primeiros trinta e dois códigos da tabela ASCII (Tabela 2-7) representam os caracteres de controle. Esses caracteres são usados para permitir que dispositivos como um computador e uma impressora se comuniquem um com o outro quando passam informações e dados. A Tabela 2-8 é uma lista dos caracteres de controle e da função da tecla de controle que permite que os caracteres sejam inseridos diretamente a partir de um teclado ASCII pressionando a tecla de controle (CTRL) e o símbolo correspondente. Também é dada uma breve descrição de cada caractere de controle.

### Caracteres Estendidos ASCII

Além dos 128 caracteres padrão ASCII, existem 128 caracteres adicionais que foram adotados pela IBM para uso em seus PCs (computadores pessoais). Devido à popularidade do PC, esses caracteres estendidos ASCII também são usados em outras aplicações além de PCs, e tornaram-se um padrão não-oficial.

Os caracteres estendidos ASCII são representados por um código 8 bits a partir do hexadecimal 80 até FF.



◀ TABELA 2-8

Caracteres de controle ASCII

NOME	DECIMAL	HEXA	TECLAS	DESCRIÇÃO
NUL	0	00	CTRL @	caractere nulo
SOH	1	01	CTRL A	início do cab. de transmissão
STX	2	02	CTRL B	início de texto
ETX	3	03	CTRL C	fim de texto
EOT	4	04	CTRL D	fim de transmissão
ENQ	5	05	CTRL E	interroga
ACK	6	06	CTRL F	confirmação
BEL	7	07	CTRL G	sinal sonoro
BS	8	08	CTRL H	volta um caractere
HT	9	09	CTRL I	tabulação horizontal
LF	10	0A	CTRL J	próxima linha
VT	11	0B	CTRL K	tabulação vertical
FF	12	0C	CTRL L	próxima página
CR	13	0D	CTRL M	início da linha
SO	14	0E	CTRL N	liberação de shift
SI	15	0F	CTRL O	ativação de shift
DLE	16	10	CTRL P	escape da conexão de dados
DC1	17	11	CTRL Q	dipositivo de controle 1
DC2	18	12	CTRL R	dipositivo de controle 2
DC3	19	13	CTRL S	dipositivo de controle 3
DC4	20	14	CTRL T	dipositivo de controle 4
NAK	21	15	CTRL U	negativa de confirmação
SYN	22	16	CTRL V	sincronismo
ETB	23	17	CTRL W	fim de transmissão de bloco
CAN	24	18	CTRL X	cancela
EM	25	19	CTRL Y	fim de meio de transmissão
SUB	26	1A	CTRL Z	substitui
ESC	27	1B	CTRL [	escape
FS	28	1C	CTRL /	separador de arquivo
GS	29	1D	CTRL ]	separador de grupo
RS	30	1E	CTRL ^	separador de registro
US	31	1F	CTRL _	separador de unidade




O código ASCII estendido contém caracteres nas seguintes categorias gerais:

1. Caracteres alfabéticos estrangeiros (idioma diferente do inglês)
2. Símbolos de moeda estrangeira
3. Letras gregas
4. Símbolos matemáticos
5. Caracteres gráficos
6. Caracteres de gráfico de barras
7. Caracteres de sombreamento

A Tabela 2-9 é uma lista do conjunto de caracteres ASCII estendidos com as representações decimal e hexadecimal.

▼ TABELA 2-9

Caracteres ASCII estendidos

SÍMBOLO	DEC	HEXA	SÍMBOLO	DEC	HEXA	SÍMBOLO	DEC	HEXA	SÍMBOLO	DEC	HEXA
Ç	128	80	á	160	A0	Ł	192	C0	α	224	E0
ü	129	81	í	161	A1	┐	193	C1	β	225	E1
é	130	82	ó	162	A2	└	194	C2	Γ	226	E2
â	131	83	ú	163	A3	┌	195	C3	π	227	E3
ä	132	84	ñ	164	A4	—	196	C4	Σ	228	E4
à	133	85	Ñ	165	A5	÷	197	C5	σ	229	E5
â	134	86	ä	166	A6	ƒ	198	C6	μ	230	E6
ç	135	87	ö	167	A7	‡	199	C7	τ	231	E7
ê	136	88	ı	168	A8	ℒ	200	C8	Φ	232	E8
ë	137	89	ƒ	169	A9	℔	201	C9	Θ	233	E9
è	138	8A	¬	170	AA	≡	202	CA	Ω	234	EA
ï	139	8B	$\frac{1}{2}$	171	AB	≡	203	CB	δ	235	EB
î	140	8C	$\frac{1}{4}$	172	AC	‡	204	CC	∞	236	EC
ì	141	8D	ı	173	AD	≡	205	CD	φ	237	ED
Ä	142	8E	«	174	AE	‡	206	CE	€	238	EE
Å	143	8F	»	175	AF	≡	207	CF	∩	239	EF
É	144	90		176	B0	┐	208	D0	≡	240	F0
æ	145	91		177	B1	≡	209	D1	±	241	F1
Æ	146	92		178	B2	└	210	D2	≥	242	F2
ô	147	93		179	B3	┌	211	D3	≤	243	F3
ö	148	94	¬	180	B4	┐	212	D4	/	244	F4
ò	149	95	≡	181	B5	℔	213	D5		245	F5
û	150	96	┐	182	B6	℔	214	D6	÷	246	F6
ù	151	97	└	183	B7	‡	215	D7	≈	247	F7
ÿ	152	98	≡	184	B8	≡	216	D8	°	248	F8
Ö	153	99	‡	185	B9	┐	217	D9	•	249	F9
Ü	154	9A		186	BA	ƒ	218	DA	·	250	FA
ç	155	9B	≡	187	BB	■	219	DB	√	251	FB
£	156	9C	≡	188	BC	■	220	DC	η	252	FC
¥	157	9D	┐	189	BD	┐	221	DD	²	253	FD
Pτ	158	9E	┐	190	BE	┐	222	DE	■	254	FE
f	159	9F	└	191	BF	■	223	DF	□	255	FF

SEÇÃO 2-11  
REVISÃO

1. Converta os seguintes números binários para o código Gray:  
(a) 1100    (b) 1010    (c) 11010
2. Converta as seguintes representações em código Gray para binário:  
(a) 1000    (b) 1010    (c) 11101
3. Qual é a representação ASCII para cada um dos seguintes caracteres? Expresse cada um como um padrão de bits e em notação hexadecimal.  
(a) K    (b) r    (c) \$    (d) +

## 2-12 CÓDIGOS DE DETECÇÃO E CORREÇÃO DE ERRO

Nesta seção, discutiremos dois métodos que acrescentam bits aos códigos com a finalidade de detectar erro num único bit ou detectar e corrigir erro num único bit. O método da paridade de detecção de erro é introduzido e o método Hamming de detecção e correção de erro num único bit é abordado com mais detalhes. Quando é identificado que um bit numa dada palavra de código está errado, ele pode ser corrigido fazendo simplesmente a inversão do bit.

Ao final do estudo desta seção você deverá ser capaz de:

- Determinar se existe um erro num código baseado no bit de paridade
- Associar a um código o bit de paridade apropriado
- Usar o código de Hamming para detecção e correção de erro num único bit
- Associar os bits de paridade adequados para correção de erro num único bit

### Método da Paridade para Detecção de Erro

Muitos sistemas usam um bit de paridade como um meio de **detecção de erro** de bit. Qualquer grupo de bits possui um número de 1s par ou ímpar. Um bit de paridade é acrescentado a um grupo de bits para tornar o número de 1s no grupo sempre par ou sempre ímpar. Um bit de paridade par torna o número de 1s par e um bit de paridade ímpar torna ímpar o total de bits.

Um dado sistema pode operar com **paridade** par ou ímpar, porém não ambas. Por exemplo, se um sistema opera com paridade par, é feita uma verificação em cada grupo de bits recebido para certificar-se de que o número total de 1s no grupo seja par. Caso exista um número ímpar de 1s, ocorreu um erro.

Como uma ilustração da forma com que os bits de paridade são acrescentados a um código, a Tabela 2-10 apresenta uma lista dos bits de paridade para cada número BCD tanto para a paridade par quanto para a ímpar. O bit de paridade para cada número BCD está na coluna *P*.

Um bit de paridade diz se o número de 1s é ímpar ou par.

PARIDADE PAR		PARIDADE ÍMPAR	
<i>P</i>	BCD	<i>P</i>	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

◀ TABELA 2-10

O código BCD com bits de paridade

O bit de paridade pode ser acrescentado ao início ou ao final do código, dependendo do projeto do sistema. Observe que o número total de 1s, incluindo o bit de paridade, é sempre par para a paridade par e sempre ímpar para a paridade ímpar.

**Detecção de um Erro** Um bit de paridade provê a detecção de erro num único bit (ou qualquer número ímpar de erros, que é bem pouco provável) mas não pode verificar dois erros num grupo. Por exemplo, vamos admitir que desejamos transmitir o código BCD 0101. (A paridade pode ser

usada com qualquer número de bits; estamos usando quatro bits como ilustração.) O código total transmitido, incluindo o bit de paridade par, é:

Bit de paridade par  
 00101  
 Código BCD

Agora vamos admitir que ocorra um erro no terceiro bit a partir da esquerda (o 1 vira 0).

Bit de paridade par  
 00001  
 Bit errado

Quando esse código é recebido, o circuito de verificação de paridade determina que existe apenas um único 1 (paridade ímpar), quando deveria haver um número par de 1s. Devido ao número par de 1s não aparecer no código recebido, é indicado um erro.

Um bit de paridade ímpar também provê uma forma de detecção de erro num único bit num dado grupo de bits.

### EXEMPLO 2-39

Associe o bit de paridade par apropriado para os seguintes grupos de códigos:

- (a) 1010                      (b) 111000                      (c) 101101  
 (d) 1000111001001                      (e) 101101011111

**Solução** Faça o bit de paridade 0 ou 1 conforme necessário para tornar o número total de 1s par. O bit de paridade será o bit mais à esquerda (colorido).

- (a) **0**1010                      (b) **1**111000                      (c) **0**101101  
 (d) **0**100011100101                      (e) **1**101101011111

**Problema relacionado** Acrescente um bit de paridade par ao código ASCII de 7 bits para a letra K.

### EXEMPLO 2-40

Um sistema de paridade ímpar recebe os seguintes grupos de código: 10110, 11010, 110011, 110101110100 e 1100010101010. Determine quais grupos, se houver algum, estão com erro.

**Solução** Como é informado que a paridade é ímpar, qualquer grupo com um número par de 1s está incorreto. Os seguintes grupos estão com erro: **110011** e **1100010101010**.

**Problema relacionado** O seguinte caractere ASCII é recebido por um sistema de paridade ímpar: 00110111. Ele está correto?

## O Código de Correção de Erro Hamming

Conforme estudado, um único bit de paridade permite a detecção de erro num único bit numa palavra de código. Um único bit de paridade pode indicar que existe um erro num certo grupo de bits. Para corrigir um erro detectado, mais informação é necessária porque a posição do bit errado tem que ser identificada antes que ele possa ser corrigido. Mais do que um bit de paridade tem que ser incluído no grupo de bits para tornar possível a correção do erro detectado. Em um código de 7 bits, existem sete possibilidades de erro num único bit. Nesse caso, três bits de paridade podem não apenas detectar um erro mas podem especificar a posição do bit errado. O **código Hamming** provê a correção de um único erro. A abordagem a seguir ilustra a construção de um código Hamming de 7 bits para a correção de um único erro.

**Número de Bits de Paridade** Se o número de bits de dados projetado for  $d$ , então o número de bits de paridade,  $p$ , é determinado pela seguinte relação:

$$2^p \geq d + p + 1$$

Por exemplo, se temos quatro bits de dados, então  $p$  é determinado por tentativa e erro por meio da Equação 2-1. Façamos  $p = 2$ . Então:

$$2^p = 2^2 = 4$$

e

$$d + p + 1 = 4 + 2 + 1 = 7$$

Como  $2^p$  tem que ser igual ou maior a  $d + p + 1$ , a relação na Equação 2-1 não é satisfeita. Temos que tentar novamente. Façamos  $p = 3$ . Então:

$$2^p = 2^3 = 8$$

e

$$d + p + 1 = 4 + 3 + 1 = 8$$

Esse valor de  $p$  satisfaz a Equação 2-1, assim são necessários três bits de paridade para proporcionar a correção de um único erro para quatro bits de dados. Deve-se notar que a detecção e correção são proporcionadas por todos os bits, de paridade e de dados, no grupo de código; ou seja, os bits de paridade também são verificados.

**Inserção de Bits de Paridade no Código** Agora que sabemos determinar o número de bits de paridade necessários no nosso exemplo particular, temos que arranjar os bits adequadamente no código. Devemos saber que nesse exemplo o código é composto de quatro bits de dados e três bits de paridade. O bit mais à esquerda é designado como bit 1, o próximo bit é o 2 e assim por diante, conforme a seguir:

bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7

Os bits de paridade estão localizados nas posições que são numeradas em correspondência às potências de dois ascendentes (1, 2, 4, 8,...), conforme indicado:

$P_1, P_2, D_1, P_3, D_2, D_3, D_4$

O símbolo  $P_n$  designa um bit de paridade em particular e  $D_n$  designa um bit de dado em particular.

**Determinação dos Valores dos Bits de Paridade** Finalmente, temos que designar adequadamente o valor 0 ou 1 a cada bit de paridade. Como cada bit de paridade provê uma verificação em outros determinados bits no código total, temos que saber o valor desses outros bits para determinar o valor do bit de paridade. Para determinar o valor do bit, primeiro numere cada posição de bit em binário, ou seja, escreva o número binário para cada número decimal da posição, conforme mostra a segunda e terceira linhas da Tabela 2-11. Em seguida, indique a localização dos bits de dados e de paridade, conforme mostra a primeira linha da Tabela 2-11. Observe que o número da posição em binário do bit de paridade  $P_1$  tem um 1 no dígito mais à direita. *Esse bit de paridade verifica as posições de todos os bits, incluindo ele mesmo, que têm 1s na mesma posição nos números de posição em binário.* Portanto, o bit de paridade  $P_1$  verifica as posições de bit 1, 3, 5 e 7.

▼ TABELA 2-11

Tabela de posicionamento dos bits para um código de correção de erro de 7 bits

DESIGNAÇÃO DOS BITS	$P_1$	$P_2$	$D_1$	$P_3$	$D_2$	$D_3$	$D_4$
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados ( $D_n$ )							
Bits de paridade ( $P_n$ )							

O número da posição em binário do bit de paridade  $P_2$  tem um 1 no bit do meio. Ele verifica todas as posições de bit, incluindo ele mesmo, que têm 1 na mesma posição. Portanto, o bit de paridade  $P_2$  verifica os bits das posições 2, 3, 6 e 7.

O número da posição em binário para o bit de paridade  $P_3$  tem um 1 no bit mais à esquerda. Ele verifica todas as posições de bit, incluindo ele mesmo, que têm 1s na mesma posição. Portanto, o bit de paridade  $P_3$  verifica as posições de bit 4, 5, 6 e 7.

Em cada caso, ao bit de paridade é designado um valor que torna a quantidade de 1s, no conjunto de bits que ele verifica, par ou ímpar, dependendo do que for especificado. Os exemplos a seguir devem tornar esse procedimento mais claro.

### EXEMPLO 2-41

Determine o código de Hamming para o número BCD 1001 (bits de dados), usando paridade par.

**Solução** **Passo 1** Determine o número de bits de paridade necessários. Façamos  $p = 3$ . então:

$$2^p = 2^3 = 8$$

$$d + p + 1 = 4 + 3 + 1 = 8$$

Três bits de paridade são suficientes.

$$\text{Total de bits do código} = 4 + 3 = 7$$

**Passo 2** Construa uma tabela de posições de bits, conforme mostra a Tabela 2-12 e insira os bits de dados. Os bits de paridade são determinados nos passos a seguir.

▼ TABELA 2-12

DESIGNAÇÃO DOS BITS	$P_1$	$P_2$	$D_1$	$P_3$	$D_2$	$D_3$	$D_4$
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados			1		0	0	1
Bits de paridade	0	0		1			

**Passo 3** Determine os bits de paridade como a seguir:

O bit  $P_1$  verifica os bits das posições 1, 3, 5 e 7 e tem que ser 0 para que o número de 1s (2) seja par nesse grupo.

O bit  $P_2$  verifica os bits das posições 2, 3, 6 e 7 e tem que ser 0 para que o número de 1s (2) seja par nesse grupo.

O bit  $P_3$  verifica os bits das posições 4, 5, 6 e 7 e tem que ser 1 para que o número de 1s (2) seja par nesse grupo.

**Passo 4** Esses bits de paridade são inseridos na Tabela 2-12 e o código combinado resultante é 0011001.

**Problema relacionado** Determine o código Hamming para o número BCD 1000 usando a paridade par.

**EXEMPLO 2-42**

Determine o código Hamming para os bits de dados 10110 usando a paridade ímpar.

**Solução** **Passo 1** Determine o número de bits de paridade necessários. Nesse caso o número de bits de dados,  $d$ , é cinco. A partir do exemplo anterior sabemos que fazendo  $p = 3$  não adianta. Experimente fazer  $p = 4$ :

$$2^p = 2^4 = 16$$

$$d + p + 1 = 5 + 4 + 1 = 10$$

Quatro bits de paridade são suficientes.

$$\text{Total de bits do código} = 5 + 4 = 9$$

**Passo 2** Construa uma tabela de posições de bits, Tabela 2-13, e insira os bits de dados. Os bits de paridade são determinados de acordo com os passos a seguir. Observe que  $P_4$  está na posição do bit 8.

**Passo 3** Determine os bits de paridade como a seguir:

▼ **TABELA 2-13**

DESIGNAÇÃO DOS BITS	$P_1$	$P_2$	$D_1$	$P_3$	$D_2$	$D_3$	$D_4$	$P_4$	$D_5$
POSIÇÃO DOS BITS	1	2	3	4	5	6	7	8	9
NÚMERO DA POS. EM BINÁRIO	0001	0010	0011	0100	0101	0110	0111	1000	1001
Bits de dados			1		0	1	1		0
Bits de paridade	1	0		1				1	

O bit  $P_1$  verifica os bits das posições 1, 3, 5, 7 e 9 e tem que ser 1 para que o número de 1s (3) seja ímpar nesse grupo.

O bit  $P_2$  verifica os bits das posições 2, 3, 6 e 7 e tem que ser 0 para que o número de 1s (3) seja ímpar nesse grupo.

O bit  $P_3$  verifica os bits das posições 4, 5, 6 e 7 e tem que ser 1 para que o número de 1s (3) seja ímpar nesse grupo.

O bit  $P_4$  verifica os bits das posições 8 e 9 e tem que ser 1 para que o número de 1s (1) seja ímpar nesse grupo.

**Passo 4:** Esses bits de paridade são inseridos na Tabela 2-13 e o código combinado resultante é 101101110.

**Problema relacionado** Determine o código Hamming para 11001 usando paridade ímpar.

## Detecção e Correção de Erro com o Código de Hamming

Agora que o método Hamming para construção de um código de erro foi abordado, como o usamos para localizar e corrigir um erro? Cada bit de paridade, ao longo dos seu grupos de bits correspondentes, tem que ser verificado para a paridade adequada. Caso existam três bits de paridade na palavra de código, são geradas três verificações. Caso existam quatro bits de paridade, são geradas quatro verificações, e assim por diante. Cada verificação de paridade apresenta um resul-

tado bom ou ruim. O resultado total de todas as verificações de paridade indica o bit, se houver algum, que está errado, como a seguir:

- Passo 1** Comece com o grupo verificado por  $P_1$ .
- Passo 2** Verifique o grupo quanto a paridade correta. Um 0 representa uma verificação de paridade correta e um 1 representa uma verificação incorreta.
- Passo 3** Repita o passo 2 para cada grupo de paridade.
- Passo 4** O número binário formado pelo resultado de todas as verificações de paridade determina a posição do bit do código que está errado. Esse é o *código de posição de erro*. A primeira verificação de paridade gera o bit menos significativo (LSB). Se todas as verificações forem corretas, não há erro.

EXEMPLO 2-43

Considere que a palavra de código dada no Exemplo 2-41 (0011001) seja transmitida e que 0010001 seja recebida. O receptor não “sabe” o que foi transmitido e tem que testar as paridades para determinar se o código está correto. Determine qualquer erro que tenha ocorrido na transmissão se a paridade usada foi a par.

**Solução** Primeiro, faça uma tabela de posição de bit, conforme indicado na Tabela 2-14.

▼ TABELA 2-14

DESIGNAÇÃO DOS BITS	$P_1$	$P_2$	$D_1$	$P_3$	$D_2$	$D_3$	$D_4$
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POSIÇÃO EM BINÁRIO	001	010	011	100	101	110	111
Código recebido	0	0	1	0	0	0	1

*Primeira verificação de paridade:*

O bit  $P_1$  verifica as posições 1, 3, 5 e 7.  
Existem dois 1s nesse grupo.  
A verificação de paridade é correta. → 0 (LSB)

*Segunda verificação de paridade:*

O bit  $P_2$  verifica as posições 2, 3, 6 e 7.  
Existem dois 1s nesse grupo.  
A verificação de paridade é correta. → 0

*Terceira verificação de paridade:*

O bit  $P_3$  verifica as posições 4, 5, 6 e 7.  
Existe um 1 nesse grupo.  
A verificação de paridade é incorreta. → 1 (MSB)

*Resultado:*

O código de posição de erro é 100 (binário quatro). Isso diz que o bit na posição 4 está errado. Ele é 0 e deveria ser 1. O código corrigido é 0011001, que está de acordo com o código transmitido.

**Problema relacionado** Repita o processo ilustrado nesse exemplo se o código recebido for 0111001.



**EXEMPLO 2-44**

O código 101101010 é recebido. Corrija qualquer erro. Existem quatro bits de paridade, sendo que a paridade usada é a ímpar.

**Solução** Primeiro, faça uma tabela de posição de bit como a Tabela 2-15.

▼ **TABELA 2-15**

DESIGNAÇÃO DOS BITS	$P_1$	$P_2$	$D_1$	$P_3$	$D_2$	$D_3$	$D_4$	$P_4$	$D_5$
POSIÇÃO DOS BITS	1	2	3	4	5	6	7	8	9
NÚMERO DA POS. EM BINÁRIO	0001	0010	0011	0100	0101	0110	0111	1000	1001
Código recebido	1	0	1	1	0	1	0	1	0

*Primeira verificação de paridade:*

O bit  $P_1$  verifica as posições 1, 3, 5, 7 e 9.

Existem dois 1s nesse grupo.

A verificação de paridade é incorreta. —————→ 1 (LSB)

*Segunda verificação de paridade:*

O bit  $P_2$  verifica as posições 2, 3, 6 e 7.

Existem dois 1s nesse grupo.

A verificação de paridade é incorreta. —————→ 1

*Terceira verificação de paridade:*

O bit  $P_3$  verifica as posições 4, 5, 6 e 7.

Existem dois 1s nesse grupo.

A verificação de paridade é incorreta. —————→ 1

*Quarta verificação de paridade:*

O bit  $P_4$  verifica as posições 8 e 9.

Existe um 1 nesse grupo.

A verificação de paridade é correta. —————→ 0 (MSB)

*Resultado:*

O código de posição de erro é 0111 (binário sete). Isso diz que o bit na posição 7 está errado. O código correto é portanto 101101110.

**Problema relacionado** O código 101111001 é recebido. Corrija qualquer erro se a paridade ímpar foi usada.

**SEÇÃO 2-12**  
**REVISÃO**

- Qual código de paridade ímpar está errado?  
(a) 1011 (b) 1110 (c) 0101 (d) 1000
- Qual código de paridade par está errado?  
(a) 11000110 (b) 00101000 (c) 10101010 (d) 11111011
- Acrescente um bit de paridade par no final de cada um dos seguintes códigos:  
(a) 1010100 (b) 0100000 (c) 1110111 (d) 10001100
- Quantos bits de paridade são necessários para os bits de dados 11010 usando o código de Hamming?
- Crie o código de Hamming para os bits de dados 0011 usando a paridade par.

## RESUMO

- O número binário é um número posicional em que o peso de cada dígito de um número inteiro é uma potência positiva de dois e o peso de cada dígito da parte fracionária é uma potência de dois negativa. Os pesos num número inteiro aumentam da direita para a esquerda (do dígito menos significativo para o mais significativo).
- Um número binário pode ser convertido para um número decimal somando os valores decimais dos pesos de todos os 1s no número binário.
- Um número inteiro decimal pode ser convertido em binário usando a soma dos pesos ou o método da divisão sucessiva por 2.
- Um número decimal fracionário pode ser convertido para binário usando a soma dos pesos ou o método da multiplicação sucessiva por 2.
- As regras básicas para a adição binária são:

$$\begin{aligned}0 + 0 &= 0 \\0 + 1 &= 1 \\1 + 0 &= 1 \\1 + 1 &= 10\end{aligned}$$

- As regras básicas para a subtração binária são:

$$\begin{aligned}0 - 0 &= 0 \\1 - 1 &= 0 \\1 - 0 &= 1 \\10 - 1 &= 1\end{aligned}$$

- O complemento de 1 de um número binário é obtido trocando 1s por 0s e 0s por 1s.
- O complemento de 2 de um número binário é obtido somando 1 ao complemento de 1.
- A subtração binária pode ser realizada por meio de adição usando o método do complemento de 1 ou de 2.
- Um número binário positivo é representado por um bit de sinal 0.
- Um número binário negativo é representado por um bit de sinal 1.
- Para operações aritméticas, os números binários negativos são representados na forma do complemento de 2 ou complemento de 1.
- Em operações de adição, um overflow é possível quando os dois números são positivos ou quando os dois números são negativos. Um bit de sinal incorreto numa soma indica a ocorrência de um overflow.
- O sistema de numeração hexadecimal consiste de 16 dígitos e caracteres, de 0 a 9 seguidos de A até F.
- Um dígito hexadecimal representa um número de 4 bits sendo a sua principal finalidade a simplificação de padrões de bits tornando-os de fácil leitura.
- Um número decimal pode ser convertido para hexadecimal usando o método da divisão sucessiva por 16.
- O sistema de numeração octal consiste de oito dígitos, de 0 a 7.
- Um número decimal pode ser convertido para octal usando o método da divisão sucessiva por 8.
- A conversão de octal para binário é realizada simplesmente substituindo cada dígito octal pelo seu equivalente binário de 3 bits. O processo é invertido na conversão de binário para octal.
- Um número decimal é convertido para BCD substituindo cada dígito decimal pelo código binário de 4 bits apropriado.
- ASCII é um código alfanumérico de 7 bits que é amplamente usado em sistemas de computador para entrada e saída de informação.
- Um bit de paridade é usado para detectar um erro num código.
- O código Hamming provê a detecção e correção de um único erro numa palavra de código.

**TERMOS IMPORTANTES**

Os termos importantes e outros termos em **negrito** destacados no capítulo são definidos no glossário que se encontra no final do livro.

**Alfanumérico** Consiste de numerais, letras e outros caracteres.

**ASCII** Código Padrão Americano para Troca de Informações; o código alfanumérico mais amplamente usado.

**BCD** Decimal codificado em binário; um código digital no qual cada um dos dígitos decimais, de 0 a 9, é representado por um grupo de quatro bits.

**Byte** Um grupo de oito bits.

**Código Hamming** Um tipo de código de correção de erro.

**Hexadecimal** Descreve um sistema de numeração com base 16.

**LSB** Bit menos significativo; o bit mais à direita num código ou número inteiro binário.

**MSB** Bit mais significativo; o bit mais à esquerda num código ou número inteiro binário.

**Número em ponto flutuante** Uma representação numérica baseada em notação científica na qual o número consiste de um expoente e uma mantissa.

**Octal** Descreve um sistema de numeração com base oito.

**Paridade** Em relação aos códigos binários, a condição de paridade par ou ímpar do número de 1s num grupo de código.

**AUTOTESTE**

As respostas estão no final do capítulo.

- $2 \cdot 10^1 + 8 \cdot 10^0$  é igual a  
(a) 10 (b) 280 (c) 2,8 (d) 28
- O número binário 1101 é igual ao número decimal  
(a) 13 (b) 49 (c) 11 (d) 3
- O número binário 1101101 é igual ao número decimal  
(a) 121 (b) 221 (c) 441 (d) 256
- O número decimal 17 é igual ao número binário  
(a) 10010 (b) 11000 (c) 10001 (d) 01001
- O número decimal 175 é igual ao número binário  
(a) 11001111 (b) 10101110 (c) 10101111 (d) 11101111
- O resultado da soma de  $11010 + 01111$  é igual a  
(a) 101001 (b) 101010 (c) 110101 (d) 101000
- A diferença de  $110 - 010$  é igual a  
(a) 001 (b) 010 (c) 101 (d) 100
- O complemento de 1 de 10111001 é  
(a) 01000111 (b) 01000110 (c) 11000110 (d) 10101010
- O complemento de 2 de 11001000 é  
(a) 00110111 (b) 00110001 (c) 01001000 (d) 00111000
- O número decimal +122 é expresso na forma do complemento de 2 como  
(a) 01111010 (b) 11111010 (c) 01000101 (d) 10000101
- O número decimal -34 é expresso na forma do complemento de 2 como  
(a) 01011110 (b) 10100010 (c) 11011110 (d) 01011101
- Um número binário de ponto flutuante de precisão simples tem um total de  
(a) 8 bits (b) 16 bits (c) 24 bits (d) 32 bits
- Na forma do complemento de 2, o número binário 10010011 é igual ao número decimal  
(a) -19 (b) +109 (c) +91 (d) -109
- O número binário 101100111001010100001 pode ser escrito em octal como  
(a) 5471230<sub>8</sub> (b) 5471241<sub>8</sub> (c) 2634521<sub>8</sub> (d) 23162501<sub>8</sub>

15. O número binário 10001101010001101111 pode ser escrito em hexadecimal como  
 (a) AD467<sub>16</sub> (b) 8C46F<sub>16</sub> (c) 8D46F<sub>16</sub> (d) AE46F<sub>16</sub>
16. O número binário equivalente a F7A9<sub>16</sub> é  
 (a) 1111011110101001 (b) 1110111110101001  
 (c) 1111111010110001 (d) 1111011010101001
17. O número BCD para o decimal 473 é  
 (a) 111011010 (b) 110001110011 (c) 010001110011 (d) 010011110011
18. Consulte a Tabela 2-7. O comando STOP em ASCII é  
 (a) 1010011101010010011111010000 (b) 1010010100110010011101010000  
 (c) 1001010110110110011101010001 (d) 1010011101010010011101100100
19. O código que tem erro de paridade par é  
 (a) 1010011 (b) 1101000 (c) 1001000 (d) 1110111

## PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

### SEÇÃO 2-1 Números Decimais

- Qual é o peso do dígito 6 em cada um dos seguintes números decimais?  
 (a) 1386 (b) 54.692 (c) 671.920
- Expresse cada um dos seguintes números decimais como uma potência de dez:  
 (a) 10 (b) 100 (c) 10.000 (d) 1.000.000
- Determine o valor de cada dígito nos números decimais a seguir:  
 (a) 471 (b) 9356 (c) 125.000
- Até que valor é possível contar com números decimais de 4 dígitos?

### SEÇÃO 2-2 Números Binários

- Converta para decimal os números binários a seguir:  
 (a) 11 (b) 100 (c) 111 (d) 1000  
 (e) 1001 (f) 1100 (g) 1011 (h) 1111
- Converta os seguintes números binários para decimal:  
 (a) 1110 (b) 1010 (c) 11100 (d) 10000  
 (e) 10101 (f) 11101 (g) 10111 (h) 11111
- Converta cada número binário a seguir para decimal:  
 (a) 110011,11 (b) 101010,01 (c) 1000001,111  
 (d) 1111000,101 (e) 1011100,10101 (f) 1110001,0001  
 (g) 1011010,1010 (h) 1111111,11111
- Qual o maior número decimal que pode ser representado pelas seguintes quantidades de dígitos binários (bits)?  
 (a) dois (b) três (c) quatro (d) cinco (e) seis  
 (f) sete (g) oito (h) nove (i) dez (j) onze
- Quantos bits são necessários para representar os seguintes números decimais?  
 (a) 17 (b) 35 (c) 49 (d) 68  
 (e) 81 (f) 114 (g) 132 (h) 205
- Determine a sequência binária para cada sequência decimal a seguir:  
 (a) 0 a 7 (b) 8 a 15 (c) 16 a 31  
 (d) 32 a 63 (e) 64 a 75

**SEÇÃO 2-3 Conversão de Decimal para Binário**

11. Converta cada número decimal a seguir para binário usando o método da soma dos pesos:  
 (a) 10    (b) 17    (c) 24    (d) 48  
 (e) 61    (f) 93    (g) 125    (h) 186
12. Converta cada fração decimal para binário usando o método da soma dos pesos:  
 (a) 0,32    (b) 0,246    (c) 0,0981
13. Converta cada número decimal para binário usando o método da divisão sucessiva por 2.  
 (a) 15    (b) 21    (c) 28    (d) 34  
 (e) 40    (f) 59    (g) 65    (h) 73
14. Converta cada fração decimal para binário usando o método da multiplicação sucessiva por 2:  
 (a) 0,98    (b) 0,347    (c) 0,9028

**SEÇÃO 2-4 Aritmética Binária**

15. Some os seguintes números binários:  
 (a)  $11 + 01$     (b)  $10 + 10$     (c)  $101 + 11$   
 (d)  $111 + 110$     (e)  $1001 + 101$     (f)  $1101 + 1011$
16. Use a subtração direta para os seguintes números binários:  
 (a)  $11 - 1$     (b)  $101 - 100$     (c)  $110 - 101$   
 (d)  $1110 - 11$     (e)  $1100 - 1001$     (f)  $11010 - 10111$
17. Realize as seguintes multiplicações binárias:  
 (a)  $11 \times 11$     (b)  $100 \times 10$     (c)  $111 \times 101$   
 (d)  $1001 \times 110$     (e)  $1101 \times 1101$     (f)  $1110 \times 1101$
18. Faça a operação de divisão binária conforme indicado:  
 (a)  $100 \div 10$     (b)  $1001 \div 11$     (c)  $1100 \div 100$

**SEÇÃO 2-5 Complementos de 1 e de 2 de Números Binários**

19. Determine o complemento de 1 de cada número binário:  
 (a) 101    (b) 110    (c) 1010  
 (d) 11010111    (e) 1110101    (f) 00001
20. Determine o complemento de 2 de cada número binário a seguir usando qualquer método:  
 (a) 10    (b) 111    (c) 1001    (d) 1101  
 (e) 11100    (f) 10011    (g) 10110000    (h) 00111101

**SEÇÃO 2-6 Números Sinalizados**

21. Expresse cada número decimal a seguir em um número binário do tipo sinal-magnitude de 8 bits:  
 (a) +29    (b) -85    (c) +100    (d) -123
22. Expresse cada número decimal a seguir como um número de 8 bits na forma do complemento de 1:  
 (a) -34    (b) +57    (c) -99    (d) +115
23. Expresse cada número decimal a seguir como um número de 8 bits na forma do complemento de 2:  
 (a) +12    (b) -68    (c) +101    (d) -125
24. Determine o valor decimal de cada número binário sinalizado a seguir na forma sinal-magnitude:  
 (a) 10011001    (b) 01110100    (c) 10111111
25. Determine o valor decimal de cada número binário sinalizado a seguir na forma do complemento de 1:  
 (a) 10011001    (b) 01110100    (c) 10111111

26. Determine o valor decimal de cada número binário sinalizado a seguir na forma do complemento de 2:
- (a) 10011001    (b) 01110100    (c) 10111111
27. Expresse cada um dos seguintes números binários no formato de ponto flutuante de precisão simples:
- (a) 0111110000101011    (b) 100110000011000
28. Determine os valores dos números em ponto flutuante de precisão simples a seguir:
- (a) 1 10000001 010010011100010000000000
- (b) 0 11001100 100001111101001000000000

### SEÇÃO 2-7 Operações Aritméticas com Números Sinalizados

29. Converta cada par de números decimais para binário e some-os usando a forma do complemento de 2.
- (a) 33 e 15    (b) 56 e -27    (c) -46 e 25    (d) -110 e -84
30. Realize cada adição a seguir na forma do complemento de 2:
- (a) 00010110 + 00110011    (b) 01110000 + 10101111
31. Realize cada adição a seguir na forma do complemento de 2:
- (a) 10001100 + 00111001    (b) 11011001 + 11100111
32. Realize cada subtração a seguir na forma do complemento de 2:
- (a) 00110011 - 00010000    (b) 01100101 - 11101000
33. Multiplique 01101010 por 11110001 na forma do complemento de 2.
34. Divida 01000100 por 00011001 na forma do complemento de 2:

### SEÇÃO 2-8 Números Hexadecimais

35. Converta para binário cada número hexadecimal a seguir:
- (a)  $38_{16}$     (b)  $59_{16}$     (c)  $A14_{16}$     (d)  $5C8_{16}$
- (e)  $4100_{16}$     (f)  $FB17_{16}$     (g)  $8A9D_{16}$
36. Converta para hexadecimal cada número binário a seguir:
- (a) 1110    (b) 10    (c) 10111
- (d) 10100110    (e) 1111110000    (f) 100110000010
37. Converta para decimal cada número hexadecimal a seguir:
- (a)  $23_{16}$     (b)  $92_{16}$     (c)  $1A_{16}$     (d)  $8D_{16}$
- (e)  $F3_{16}$     (f)  $EB_{16}$     (g)  $5C2_{16}$     (h)  $700_{16}$
38. Converta para hexadecimal cada número decimal a seguir:
- (a) 8    (b) 14    (c) 33    (d) 52
- (e) 284    (f) 2890    (g) 4019    (h) 6500
39. Realize as seguintes adições:
- (a)  $37_{16} + 29_{16}$     (b)  $A0_{16} + 6B_{16}$     (c)  $FF_{16} + BB_{16}$
40. Realize as seguintes subtrações:
- (a)  $51_{16} - 40_{16}$     (b)  $C8_{16} - 3A_{16}$     (c)  $FD_{16} - 88_{16}$

### SEÇÃO 2-9 Números Octais

41. Converta para decimal cada número octal a seguir:
- (a)  $12_8$     (b)  $27_8$     (c)  $56_8$     (d)  $64_8$     (e)  $103_8$
- (f)  $557_8$     (g)  $163_8$     (h)  $1024_8$     (i)  $7765_8$
42. Converta para octal cada número decimal a seguir fazendo divisões sucessivas por 8:
- (a) 15    (b) 27    (c) 46    (d) 70
- (e) 100    (f) 142    (g) 219    (h) 435

43. Converta para binário cada número octal a seguir:
- (a)  $13_8$       (b)  $57_8$       (c)  $101_8$       (d)  $321_8$       (e)  $540_8$   
 (f)  $4653_8$       (g)  $13271_8$       (h)  $45600_8$       (i)  $100213_8$
44. Converta para octal cada número binário a seguir:
- (a) 111      (b) 10      (c) 110111  
 (d) 101010      (e) 1100      (f) 1011110  
 (g) 101100011001      (h) 10110000011      (i) 111111101111000

## SEÇÃO 2-10 Decimal Codificado em Binário (BCD)

45. Converta para BCD 8421 cada um dos seguintes números decimais:
- (a) 10      (b) 13      (c) 18      (d) 21      (e) 25      (f) 36  
 (g) 44      (h) 57      (i) 69      (j) 98      (k) 125      (l) 156
46. Converta para binário direto cada um dos números do Problema 45 e compare o número de bits necessários nesses dois problemas.
47. Converta para BCD os seguintes números decimais:
- (a) 104      (b) 128      (c) 132      (d) 150      (e) 186  
 (f) 210      (g) 359      (h) 547      (i) 1051
48. Converta para decimal os números BCD a seguir:
- (a) 0001      (b) 0110      (c) 1001  
 (d) 00011000      (e) 00011001      (f) 00110010  
 (g) 01000101      (h) 10011000      (i) 100001110000
49. Converta para decimal cada um dos números BCD a seguir:
- (a) 10000000      (b) 001000110111  
 (c) 001101000110      (d) 010000100001  
 (e) 011101010100      (f) 100000000000  
 (g) 100101111000      (h) 0001011010000011  
 (i) 1001000000011000      (j) 0110011001100111
50. Some os seguintes números BCD:
- (a)  $0010 + 0001$       (b)  $0101 + 0011$   
 (c)  $0111 + 0010$       (d)  $1000 + 0001$   
 (e)  $00011000 + 00010001$       (f)  $01100100 + 00110011$   
 (g)  $01000000 + 01000111$       (h)  $10000101 + 00010011$
51. Some os seguintes números BCD:
- (a)  $1000 + 0110$       (b)  $0111 + 0101$   
 (c)  $1001 + 1000$       (d)  $1001 + 0111$   
 (e)  $00100101 + 00100111$       (f)  $01010001 + 01011000$   
 (g)  $10011000 + 10010111$       (h)  $010101100001 + 011100001000$
52. Converta para BCD cada par de números decimais e faça a soma conforme indicado:
- (a)  $4 + 3$       (b)  $5 + 2$       (c)  $6 + 4$       (d)  $17 + 12$   
 (e)  $28 + 23$       (f)  $65 + 58$       (g)  $113 + 101$       (h)  $295 + 157$

## SEÇÃO 2-11 Códigos Digitais

53. Numa determinada aplicação, uma sequência de 4 bits varia ciclicamente de 1111 a 0000. Existe uma alteração de 4 bits, e em função de atrasos no circuito, essas alterações podem não ocorrer no mesmo instante. Por exemplo, se o LSB mudar primeiro, o número aparecerá como 1110 durante a transição de 1111 para 0000 podendo ser interpretado erroneamente pelo sistema. Ilustre como o código Gray evita esse problema.

54. Converta para código Gray cada número binário a seguir:  
 (a) 11011 (b) 1001010 (c) 1111011101110
55. Converta para binário cada código Gray a seguir:  
 (a) 1010 (b) 00010 (c) 11000010001
56. Converta para ASCII cada um dos seguintes números decimais. Consulte a Tabela 2–7.  
 (a) 1 (b) 3 (c) 6 (d) 10 (e) 18  
 (f) 29 (g) 56 (h) 75 (i) 107
57. Determine cada caractere codificado a seguir em ASCII. Consulte a Tabela 2–7.  
 (a) 0011000 (b) 1001010 (c) 0111101  
 (d) 0100011 (e) 0111110 (f) 1000010
58. Decodifique a seguinte mensagem codificada em ASCII:  
 1001000 1100101 1101100 1101100 1101111 0101110  
 0100000 1001000 1101111 1110111 0100000 1100001  
 1110010 1100101 0100000 1111001 1101111 1110101  
 0111111
59. Escreva em hexadecimal a mensagem apresentada no Problema 58.
60. Converta para ASCII a seguinte linha comando de um programa de computador:  
 30 INPUT A,B

## SEÇÃO 2–12 Códigos de Detecção e Correção de Erro

61. Determine qual dos seguintes códigos com paridade par apresenta erro:  
 (a) 100110010 (b) 011101010 (c) 10111111010001010
62. Determine qual dos seguintes códigos com paridade ímpar apresenta erro:  
 (a) 11110110 (b) 00110001 (c) 010101010101010
63. Acrescente um bit de paridade par aos seguintes bytes de dados:  
 (a) 10100100 (b) 00001001 (c) 11111110
64. Determine o código de Hamming com paridade par para os bits de dados 1100.
65. Determine o código de Hamming com paridade ímpar para os bits de dados 1101.
66. Corrija qualquer erro em cada um dos seguintes códigos de Hamming com paridade par.  
 (a) 1110100 (b) 1000111
67. Corrija qualquer erro em cada um dos seguintes códigos de Hamming com paridade ímpar.  
 (a) 110100011 (b) 100001101

## RESPOSTAS

### SEÇÕES DE REVISÃO

#### SEÇÃO 2–1 Números Decimais

1. (a) 1370: 10 (b) 6725: 100 (c) 7051: 1000 (d) 58,72: 0,1
2. (a)  $51 = (5 \text{ } 10) + (1 \text{ } 1)$  (b)  $137 = (1 \text{ } 100) + (3 \text{ } 10) + (7 \text{ } 1)$  (c)  $1492 = (1 \text{ } 1000) + (4 \text{ } 100) + (9 \text{ } 10) + (2 \text{ } 1)$  (d)  $106,58 = (1 \text{ } 100) + (0 \text{ } 10) + (6 \text{ } 1) + (5 \text{ } 0,1) + (8 \text{ } 0,01)$

#### SEÇÃO 2–2 Números Binários

1.  $2^8 - 1 = 255$
2. O peso é 16.
3.  $10111101,011 = 189,375$



**SEÇÃO 2-3 Conversão de Decimal para Binário**

1. (a)  $23 = 10111$  (b)  $57 = 111001$  (c)  $45,5 = 101101,1$
2. (a)  $14 = 1110$  (b)  $21 = 10101$  (c)  $0,375 = 0,011$

**SEÇÃO 2-4 Aritmética Binária**

1. (a)  $1101 + 1010 = 10111$  (b)  $10111 + 01101 = 100100$
2. (a)  $1101 - 0100 = 1001$  (b)  $1001 - 0111 = 0010$
3. (a)  $110 \cdot 111 = 101010$  (b)  $1100 \div 011 = 100$

**SEÇÃO 2-5 Complementos de 1 e de 2 de Números Binários**

1. (a) Compl. de 1 de  $00011010 = 11100101$  (b) Compl. de 1 de  $11110111 = 0000100$   
(c) Compl. de 1 de  $10001101 = 01110010$
2. (a) Compl. de 2 de  $00010110 = 11101010$  (b) Compl. de 2 de  $11111100 = 0000010$   
(c) Compl. de 2 de  $10010001 = 01101111$

**SEÇÃO 2-6 Números Sinalizados**

1. Sinal-magnitude:  $+9 = 00001001$
2. Complemento de 1:  $-33 = 11011110$
3. Complemento de 2:  $-46 = 11010010$
4. Bit de sinal, expoente e mantissa.

**SEÇÃO 2-7 Operações Aritméticas com Números Sinalizados**

1. Casos da adição: o número positivo é maior, o número negativo é maior, ambos são positivos, ambos são negativos.
2.  $00100001 + 10111100 = 11011101$
3.  $01110111 - 00110010 = 01000101$
4. O sinal do produto é positivo.
5.  $00000101 \cdot 01111111 = 01001111011$
6. O sinal do quociente é negativo.
7.  $00110000 \div 00001100 = 00000100$

**SEÇÃO 2-8 Números Hexadecimais**

1. (a)  $10110011 = B3_{16}$  (b)  $110011101000 = CE8_{16}$
2. (a)  $57_{16} = 01010111$  (b)  $3A5_{16} = 001110100101$   
(c)  $F80B_{16} = 1111100000001011$
3.  $9B30_{16} = 39.728_{10}$
4.  $573_{10} = 23D_{16}$
5. (a)  $18_{16} + 34_{16} = 4C_{16}$  (b)  $3F_{16} + 2A_{16} = 69_{16}$
6. (a)  $75_{16} - 21_{16} = 54_{16}$  (b)  $94_{16} - 5C_{16} = 38_{16}$

**SEÇÃO 2-9 Números Octais**

1. (a)  $73_8 = 59_{10}$  (b)  $125_8 = 85_{10}$
2. (a)  $98_{10} = 142_8$  (b)  $163_{10} = 243_8$
3. (a)  $46_8 = 100110$  (b)  $723_8 = 111010011$  (c)  $5624_8 = 101110010100$
4. (a)  $110101111 = 657_8$  (b)  $1001100010 = 1142_8$  (c)  $10111111001 = 2771_8$

**SEÇÃO 2-10 Decimal Codificado em Binário (BCD)**

1. (a)  $0010: 2$  (b)  $1000: 8$  (c)  $0001: 1$  (d)  $0100: 4$

2. (a)  $6_{10} = 0110$  (b)  $15_{10} = 00010101$  (c)  $273_{10} = 001001110011$   
 (d)  $849_{10} = 100001001001$   
 3. (a)  $10001001 = 89_{10}$  (b)  $001001111000 = 278_{10}$  (c)  $000101010111 = 157_{10}$   
 4. Um resultado de 4 bits é inválido quando ele for maior que  $9_{10}$ .

### SEÇÃO 2-11 Códigos Digitais

1. (a)  $1100_2 = 1010$  Gray (b)  $1010_2 = 1111$  Gray (c)  $11010_2 = 10111$  Gray  
 2. (a)  $1000$  Gray =  $1111_2$  (b)  $1010$  Gray =  $1100_2$  (c)  $11101$  Gray =  $10110_2$   
 3. (a) K:  $1001011 \rightarrow 4B_{16}$  (b) r:  $1110010 \rightarrow 72_{16}$   
 (c) \$:  $0100100 \rightarrow 24_{16}$  (d) +:  $0101011 \rightarrow 2B_{16}$

### SEÇÃO 2-12 Códigos de Detecção e Correção de Erro

1. (c) 0101 tem um erro.  
 2. (d) 11111011 tem um erro.  
 3. (a) 10101001 (b) 01000001 (c) 11101110 (d) 10001101  
 4. Quatro bits de paridade.  
 5. 1000011 (os bits de paridade estão em cor)

### PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

- 2-1. 9 tem um valor de 900, 3 tem um valor de 30, 9 tem um valor de 9.  
 2-2. 6 tem um valor de 60, 7 tem um valor de 7, 9 tem um valor de 9/10 (0,9), 2 tem um valor de 2/100 (0,02), 4 tem um valor de 4/1000 (0,004).  
 2-3.  $10010001 = 128 + 16 + 1 = 145$  2-4.  $10,111 = 2 + 0,5 + 0,25 + 0,125 = 2,875$   
 2-5.  $125 = 64 + 32 + 16 + 8 + 4 + 1 = 1111101$  2-6.  $39 = 100111$   
 2-7.  $1111 + 1100 = 11011$  2-8.  $111 - 100 = 011$  2-9.  $110 - 101 = 001$   
 2-10.  $1101 \quad 1010 = 10000010$  2-11.  $1100 \div 100 = 11$  2-12.  $00110101$   
 2-13.  $01000000$  2-14. Veja a Tabela 2-16. 2-15.  $01110111 = +119_{10}$

► TABELA 2-16

	SINAL - MAGNITUDE	COMPL. DE 1	COMPL. DE 2
+19	00010011	00010011	00010011
-19	10010011	11101100	11101101

- 2-16.  $11101011 = -20_{10}$  2-17.  $11010111 = -41_{10}$   
 2-18.  $11000010001010011000000000$  2-19.  $01010101$  2-20.  $00010001$   
 2-21.  $1001000110$  2-22.  $(83)(-59) = -4897$  ( $10110011011111$  em complemento de 2)  
 2-23.  $100 \div 25 = 4$  (0100) 2-24.  $4F79C_{16}$  2-25.  $0110101111010011_2$   
 2-26.  $6BD_{16} = 011010111101 = 2^{10} + 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0$   
 $= 1024 + 512 + 128 + 32 + 16 + 8 + 4 + 1 = 1725_{10}$   
 2-27.  $60A_{16} = (6 \quad 256) + (0 \quad 16) + (10 \quad 1) = 1546_{10}$   
 2-28.  $2591_{10} = A1F_{16}$  2-29.  $4C_{16} + 3A_{16} = 86_{16}$   
 2-30.  $BCD_{16} - 173_{16} = A5A_{16}$   
 2-31. (a)  $001011_2 = 11_{10} = 13_8$  (b)  $010101_2 = 21_{10} = 25_8$   
 (c)  $001100000_2 = 96_{10} = 140_8$  (d)  $111101010110_2 = 3926_{10} = 7526_8$

- 2-32.  $1250762_8$     2-33.  $1001011001110011$     2-34.  $82,276_{10}$   
 2-35.  $1001100101101000$     2-36.  $10000010$     2-37. (a)  $111011$  (Gray)    (b)  $111010_2$   
 2-38. A sequência de códigos para 80 INPUT Y é  $38_{16}30_{16}20_{16}49_{16}4E_{16}50_{16}55_{16}54_{16}20_{16}59_{16}$   
 2-39.  $01001011$     2-40. Sim    2-41.  $1110000$     2-42.  $001010001$   
 2-43. O bit na posição 010 (2) está errado. O correto é  $0011001$ .  
 2-44. O bit na posição 0010 (2) está errado. O correto é  $111111000$ .

**AUTOTESTE**

1. (d)    2. (a)    3. (b)    4. (c)    5. (c)    6. (a)    7. (d)    8. (b)  
 9. (d)    10. (a)    11. (c)    12. (d)    13. (d)    14. (b)    15. (c)    16. (a)  
 17. (c)    18. (a)    19. (b)