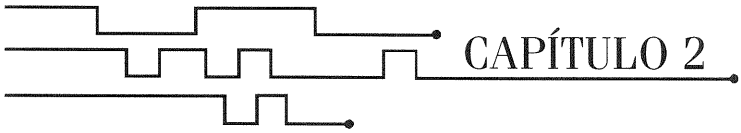
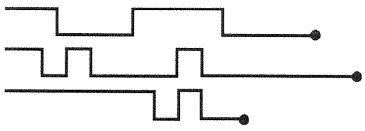


CAPÍTULO 2

Sistemas de Numeração e Códigos



SUMÁRIO

■ OBJETIVOS

Ao completar este capítulo, você deverá estar apto a:

- Usar dois métodos diferentes para realizar conversões de decimal para binário.
- Citar as diversas vantagens dos sistemas de numeração octal e hexadecimal.
- Converter do sistema de numeração hexadecimal ou octal tanto para o decimal quanto para o binário.
- Representar números decimais usando o código BCD.
- Compreender a diferença entre o código BCD e o código binário puro.
- Entender a necessidade dos códigos alfanuméricos, especialmente do código ASCII.
- Descrever o método da paridade para detecção de erros.
- Determinar a paridade (par ou ímpar) de dados digitais.

■ INTRODUÇÃO

O sistema de numeração binário é o mais importante em sistemas digitais, mas muitos outros também o são. O sistema decimal é importante porque é universalmente usado para representar quantidades externas a um sistema digital. Isto significa que existem situações nas quais valores decimais devem ser convertidos para valores binários antes de entrarem num sistema digital. Por exemplo, quando você digita um número decimal na calculadora (ou computador), os circuitos internos do dispositivo convertem o número decimal num valor binário.

Do mesmo modo, existem situações em que os valores binários nas saídas de um sistema digital devem ser convertidos em valores decimais para apresentação ao mundo exterior. Por exemplo, sua calculadora (ou computador) utiliza números binários para calcular as respostas de um problema e então converte-as para decimal antes de apresentá-las.

Além dos sistemas binário e decimal, dois outros sistemas de numeração encontram aplicações em diversas áreas de sistemas digitais. Os sistemas de numeração *octal* (base 8) e *hexadecimal* (base 16) são usados para o mesmo propósito — fornecer um meio eficiente de representar números binários grandes. Como veremos, os dois sistemas têm a vantagem de ser facilmente convertidos para o sistema binário e vice-versa.

Num sistema digital, três ou quatro desses sistemas de numeração podem estar em uso ao mesmo tempo, portanto o entendimento da operação do sistema requer a habilidade de converter de um sistema numérico para outro. Este capítulo apresenta como realizar estas conversões. Embora alguns deles não sejam de uso imediato no nosso estudo de sistemas digitais, você precisará deles quando começar a estudar microprocessadores.

Este capítulo também introduz alguns dos *códigos binários* que são usados para representar vários tipos de informação. Estes códigos binários utilizam 1s e 0s, mas de um modo que difere do sistema de numeração binário.

2-1 CONVERSÕES BINÁRIO-DECIMAL

Conforme explicado no Cap. 1, o sistema de numeração binário é um sistema posicional em que cada dígito binário (bit) tem um certo peso de acordo com sua posição relativa ao LSB. Qualquer número binário pode ser convertido para o seu equivalente decimal simplesmente somando-se os pesos das várias posições que contiverem 1 no número binário. Para ilustrar:

$$\begin{array}{rl} 1\quad 1\quad 0\quad 1\quad 1_2 & \textit{(binário)} \\ 2^4 + 2^3 + 0 + 2^1 + 2^0 = 16 + 8 + 2 + 1 & \\ = 27_{10} & \textit{(decimal)} \end{array}$$

Vejamos outro exemplo com um maior número de bits:

$$\begin{array}{l} 1\quad 0\quad 1\quad 1\quad 0\quad 1\quad 0\quad 1_2 = \\ 2^7 + 0 + 2^5 + 2^4 + 0 + 2^2 + 0 + 2^0 = 181_{10} \end{array}$$

Note que o procedimento é determinar os pesos (isto é, as potências de 2) para cada bit que contém 1, e então somá-los. Note também que o MSB tem um peso de 2^7, embora seja o oitavo bit; isto acontece porque o LSB é o primeiro bit e tem um peso de 2^0.

Questões de Revisão

1. Converta 100011011011_2 para seu equivalente decimal.
2. Qual é o peso do MSB de um número de 16 bits?

2-2 CONVERSÕES DECIMAL-BINÁRIO

Há dois modos de converter um número decimal *inteiro* para a representação equivalente no sistema binário. O primeiro método é o inverso do processo descrito na Seção 2-1. O número decimal é simplesmente representado como uma soma de potências de 2, e então 1s e 0s são escritos nas posições de bit apropriadas. Para ilustrar:

$$\begin{array}{rl} 45_{10} = 32 + 8 + 4 + 1 = 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0 & \\ = 1\quad 0\quad 1\quad 1\quad 0\quad 1_2 & \end{array}$$

Note que um 0 é colocado nas posições de 2^1 e 2^4, já que todas as posições devem ser levadas em conta. Um outro exemplo é o seguinte:

$$\begin{array}{rl} 76_{10} = 64 + 8 + 4 = 2^6 + 0 + 0 + 2^3 + 2^2 + 0 + 0 & \\ = 1\quad 0\quad 0\quad 1\quad 1\quad 0\quad 0_2 & \end{array}$$

Divisões Sucessivas

Um outro método para converter números decimais inteiros utiliza sucessivas divisões por 2. A conversão, ilustrada adiante para 25_{10}, requer repetidas divisões do número decimal por 2 e a escrita do resto de cada divisão até que o quociente 0 seja obtido. Note que o resultado binário é obtido escrevendo-se o primeiro resto como o LSB e o último resto como o MSB.

$$\frac{25}{2} = 12 + \text{o resto de } 1$$
$$\frac{12}{2} = 6 + \text{o resto de } 0$$
$$\frac{6}{2} = 3 + \text{o resto de } 0$$
$$\frac{3}{2} = 1 + \text{o resto de } 1$$
$$\frac{1}{2} = 0 + \text{o resto de } 1$$

MSB

$25_{10} = 1\ 1\ 0\ 0\ 1_2$

Este processo, mostrado no fluxograma da Fig. 2-1, também pode ser usado para converter de decimal para qualquer outro sistema de numeração.

Se uma calculadora for usada para realizar as divisões por 2, os restos podem ser determinados notando se o quociente tem ou não uma parte fracionária. Por exemplo, a calculadora deveria produzir 25/2 = 12,5. O ",5" indica que existe um resto 1. A calculadora deveria também produzir 12/2 = 6,0, que indica um resto 0. No exemplo seguinte é mostrado o que ocorreria usando-se uma calculadora.

$$\frac{37}{2} = 18,5 \longrightarrow \text{o resto de } 1 \text{ (LSB)}$$
$$\frac{18}{2} = 9,0 \longrightarrow 0$$
$$\frac{9}{2} = 4,5 \longrightarrow 1$$
$$\frac{4}{2} = 2,0 \longrightarrow 0$$
$$\frac{2}{2} = 1,0 \longrightarrow 0$$
$$\frac{1}{2} = 0,5 \longrightarrow 1 \text{ (MSB)}$$

Logo, $37_{10} = 100101_2$.

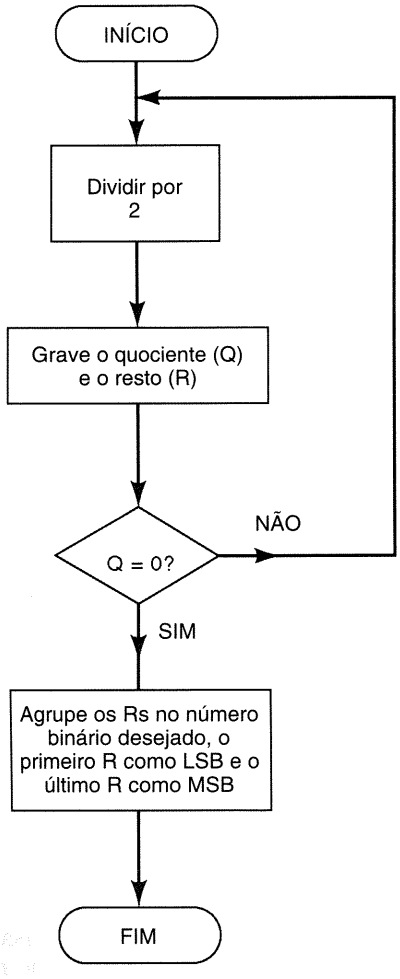


Fig. 2-1 Fluxograma do método das divisões sucessivas para a conversão decimal-binário de inteiros. O mesmo processo pode ser usado para converter um inteiro decimal para qualquer outro sistema de numeração.

Faixa de Contagem

Lembre-se de que usando N bits podemos contar 2^N valores decimais diferentes variando de 0 até $2^N - 1$. Por exemplo, para $N = 4$, podemos contar de 0000_2 até 1111_2, ou seja, de 0_{10} até 15_{10}, totalizando 16 números diferentes. O maior valor decimal é $2^4 - 1 = 15$, e existem 2^4 números diferentes.

Portanto, de um modo geral, podemos afirmar:

Usando N bits, podemos representar valores decimais variando de 0 até 2^N 2 1, num total de 2^N valores.

EXEMPLO 2-1

(a) Qual é a faixa de valores decimais que pode ser representada com oito bits?

(b) Quantos bits são necessários para representar valores decimais variando de 0 até 12.500?

Solução

(a) Aqui temos $N = 8$. Logo, podemos representar números decimais desde 0 até $2^8 - 1 = 255$. Podemos verificar isto constatando que 11111111_2 equivale a 255_{10}.

(b) Com 13 bits, podemos contar de 0 até $2^{13} - 1 = 8.191$. Com 14 bits, podemos contar de 0 até $2^{14} - 1 = 16.383$. Claramente, 13 bits não são suficientes, e com 14 bits obtemos além de 12.500. Portanto, o número de bits necessário é 14.

Questões de Revisão

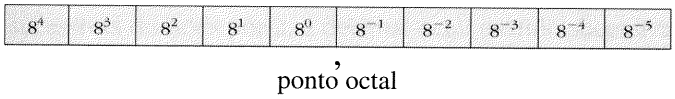
1. Converta 83_{10} para binário utilizando ambos os métodos.

2. Converta 729_{10} para binário utilizando ambos os mé-

todos. Verifique sua resposta convertendo de volta para decimal.
3. Quantos bits são necessários para contar até 1 milhão em decimal?

2-3 SISTEMA DE NUMERAÇÃO OCTAL

O sistema de numeração octal é muito importante no trabalho com computadores digitais. O **sistema de numeração octal** tem base *oito*, significando que tem oito dígitos possíveis: 0, 1, 2, 3, 4, 5, 6 e 7. Assim, cada dígito de um número octal pode ter valores de 0 a 7. As posições dos dígitos num número octal têm pesos, como segue:



Conversão Octal-Decimal

Um número octal pode ser facilmente convertido para seu equivalente decimal multiplicando-se cada dígito octal pelo seu peso posicional. Por exemplo:

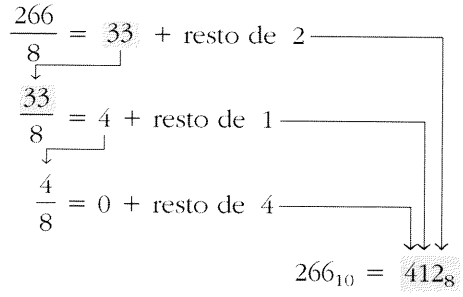
$$\begin{aligned} 372_8 &= 3 \times (8^2) + 7 \times (8^1) + 2 \times (8^0) \\ &= 3 \times 64 + 7 \times 8 + 2 \times 1 \\ &= 250_{10} \end{aligned}$$

Eis outro exemplo:

$$\begin{aligned} 24{,}6_8 &= 2 \times (8^1) + 4 \times (8^0) + 6 \times (8^{-1}) \\ &= 20{,}75_{10} \end{aligned}$$

Conversão Decimal-Octal

Um inteiro decimal pode ser convertido para octal utilizando o mesmo método das divisões sucessivas que foi usado na conversão decimal-binário (Fig. 2-1), mas com o fator de divisão 8 em vez de 2. Um exemplo é mostrado a seguir.



Note que o primeiro resto se torna o dígito menos significativo (LSD) do número octal, e o último resto se torna o dígito mais significativo (MSD).

Se uma calculadora é usada para realizar as divisões no processo anterior, o resultado incluirá uma fração decimal em vez de um resto. O resto pode ser obtido multiplicando-se a fração decimal por 8. Por exemplo, 266/8 produz 33,25. O resto é 0,25 × 8 = 2. Analogamente, 33/8 é 4,125, e o resto se torna 0,125 × 8 = 1.

Conversão Octal-Binário

A principal vantagem do sistema de numeração octal é a facilidade com que conversões podem ser feitas entre números binários e octais. A conversão de octal para binário é realizada convertendo-se *cada* dígito octal nos três bits binários equivalentes. Os oito dígitos possíveis são convertidos conforme indicado na Tabela 2-1.

TABELA 2-1

Dígito Octal	0	1	2	3	4	5	6	7
Equivalente Binário	000	001	010	011	100	101	110	111

Usando essas conversões, podemos converter qualquer número octal para binário convertendo individualmente cada dígito. Por exemplo, podemos converter 472_8 para binário como segue:

$$\begin{array}{ccc} 4 & 7 & 2 \\ \downarrow & \downarrow & \downarrow \\ 100 & 111 & 010 \end{array}$$

Portanto, o octal 472 é equivalente ao binário 100111010. Como outro exemplo, considere a conversão de 5431_8 para binário:

$$\begin{array}{cccc} 5 & 4 & 3 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 101 & 100 & 011 & 001 \end{array}$$

Assim, $5431_8 = 101100011001_2$.

Conversão Binário-Octal

Converter binários inteiros para octais inteiros é simplesmente o inverso do processo anterior. Os bits do número binário são reunidos em grupos de *três* bits iniciando-se do LSB. Então cada grupo é convertido para seu equivalente octal (Tabela 2-1). Para ilustrar, considere a conversão de 100111010_2 para octal.

$$\begin{array}{ccc} \underbrace{1\ 0\ 0} & \underbrace{1\ 1\ 1} & \underbrace{0\ 1\ 0} \\ \downarrow & \downarrow & \downarrow \\ 4 & 7 & 2_8 \end{array}$$

Algumas vezes, o número binário não tem grupos completos de três bits. Nesses casos, podemos adicionar um ou dois 0s à esquerda do MSB do número binário para preencher o último grupo. Isto é ilustrado adiante para o número binário 11010110.

$$\begin{array}{ccc} \underbrace{0\ 1\ 1} & \underbrace{0\ 1\ 0} & \underbrace{1\ 1\ 0} \\ \downarrow & \downarrow & \downarrow \\ 3 & 2 & 6_8 \end{array}$$

Note que um 0 foi colocado à esquerda do MSB para produzir grupos completos de três bits.

Contando em Octal

O maior dígito octal é 7, portanto na contagem em octal cada posição de dígito é incrementada de 0 a 7. Uma vez

alcançado o 7, ele retorna para 0 na próxima contagem e causa o incremento da próxima posição de dígito mais alta. Isto é ilustrado nas seguintes seqüências de contagem octal: (1) 65, 66, 67, 70, 71 e (2) 275, 276, 277, 300.

Com N posições de dígitos octais, podemos contar de 0 até $8^N - 1$, para um total de 8^N valores diferentes. Por exemplo, com três posições de dígitos octais podemos contar de 000_8 até 777_8, ou seja, de 0_{10} até 511_{10} para um total de $8^3 = 512_{10}$ números octais diferentes.

Utilidade do Sistema Octal

A facilidade com que as conversões podem ser feitas entre octal e binário torna o sistema octal atrativo como um modo "compacto" de expressar números binários grandes. No trabalho com computadores, números binários com até 64 bits não são incomuns. Estes números binários, conforme veremos, nem sempre representam uma quantidade numérica, mas são algum tipo de código que carregam freqüentemente informação não-numérica. Nos computadores, números binários podem representar (1) dados numéricos puros, (2) números correspondentes a posições (endereços) de memória, (3) um código de instrução, (4) um código representando caracteres alfabéticos e outros não-numéricos, ou (5) um grupo de bits representando o estado de dispositivos internos ou externos ao computador.

Quando lidamos com uma grande quantidade de números binários de vários bits, é conveniente e mais eficiente escrevermos os números em octal em vez de binário. Não devemos esquecer, no entanto, que circuitos e sistemas digitais trabalham exclusivamente em binário; usamos octal somente por conveniência para os operadores do sistema.

EXEMPLO 2-2

Converta 177_{10} para seu equivalente binário de oito bits, convertendo primeiramente para octal.

Solução

$$\frac{177}{8} = 22 + \text{resto de } 1$$

$$\frac{22}{8} = 2 + \text{resto de } 6$$

$$\frac{2}{8} = 0 + \text{resto de } 2$$

Assim, $177_{10} = 261_8$. Agora podemos converter este número octal para seu equivalente binário 010110001_2, e finalmente temos

$$177_{10} = 10110001_2$$

Note que descartamos o 0 à esquerda para expressar o resultado com 8 bits.

Este método de conversão decimal-octal-binário freqüentemente é mais rápido do que converter diretamente de decimal para binário, sobretudo para números grandes. De modo semelhante, freqüentemente é mais rápido converter de binário para decimal convertendo primeiro para octal.

Questões de Revisão

1. Converta 614_8 para decimal.
2. Converta 146_{10} para octal, e então de octal para binário.
3. Converta 10011101_2 para octal.
4. Escrever os três próximos números nesta seqüência de contagem octal: 624, 625, 626, ____, ____, ____.
5. Converta 975_{10} para binário, convertendo-o primeiramente para octal.
6. Converta o binário 1010111011 para decimal, convertendo-o primeiramente para octal.
7. Qual é a faixa de valores decimais que pode ser representada por um número octal de quatro dígitos?

2-4 SISTEMA DE NUMERAÇÃO HEXADECIMAL

O **sistema de numeração hexadecimal** usa a base 16. Assim, ele tem 16 símbolos possíveis. Ele usa os dígitos 0 a 9 mais as letras A, B, C, D, E e F como os 16 símbolos. A Tabela 2-2 mostra as relações entre hexadecimal, decimal e binário. Note que cada dígito hexadecimal representa um grupo de quatro dígitos binários. É importante lembrar que os dígitos hexa (abreviatura de "hexadecimal") A até F são equivalentes aos valores decimais 10 até 15.

TABELA 2-2

Hexadecimal	Decimal	Binário
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Conversão Hexadecimal-Decimal

Um número hexa pode ser convertido para seu equivalente decimal usando o fato de que cada posição de dígito hexa tem um peso que é uma potência de 16. O LSD tem um peso de $16^0 = 1$; a próxima posição de dígito mais alta tem um peso de $16^1 = 16$; a próxima tem um peso de $16^2 = 256$; e assim por diante. O processo de conversão é demonstrado nos exemplos a seguir:

$$356_{16} = 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0$$
$$= 768 + 80 + 6$$
$$= 854_{10}$$

$$2AF_{16} = 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0$$
$$= 512 + 160 + 15$$
$$= 687_{10}$$

Note que no segundo exemplo o valor 10 substituiu o A e o valor 15 o F na conversão para decimal.

Para praticar, verifique que $1BC2_{16}$ é igual a 7106_{10}.

Conversão Decimal-Hexadecimal

Relembre que fizemos conversões decimal-binário usando sucessivas divisões por 2, e decimal-octal usando sucessivas divisões por 8. Do mesmo modo, conversões decimal-hexadecimal podem ser feitas usando sucessivas divisões por 16 (Fig. 2-1). Os exemplos seguintes ilustram o método.

EXEMPLO 2-3

Converta 423_{10} para hexa.

Solução

$$\frac{423}{16} = 26 + \text{resto de } 7$$
$$\frac{26}{16} = 1 + \text{resto de } 10$$
$$\frac{1}{16} = 0 + \text{resto de } 1$$
$$423_{10} = 1A7_{16}$$

EXEMPLO 2-4

Converta 214_{10} para hexa.

Solução

$$\frac{214}{16} = 13 + \text{resto de } 6$$
$$\frac{13}{16} = 0 + \text{resto de } 13$$
$$214_{10} = D6_{16}$$

Repare novamente que os restos do processo de divisão formam os dígitos do número hexa. Note também que qualquer resto maior do que 9 é representado pelas letras de A até F.

Se uma calculadora está sendo usada para realizar as divisões do processo de conversão, os resultados incluirão uma fração decimal em vez de um resto. O resto pode ser obtido multiplicando-se a fração por 16. Para ilustrar, no Exemplo 2-4 uma calculadora produziria

$$\frac{214}{16} = 13{,}375$$

O resto se torna $(0{,}375) \times 16 = 6$.

Conversão Hexadecimal-Binário

Assim como o sistema de numeração octal, o sistema de numeração hexadecimal é usado principalmente como um método "compacto" para representação de números binários. É relativamente simples converter um número hexa em binário. *Cada* dígito hexa é convertido para seu equivalente de quatro bits (Tabela 2-2). Isto é ilustrado a seguir para $9F2_{16}$.

$$9F2_{16} = \begin{array}{ccccccccccccc} & 9 & & & & F & & & & 2 & & & \\ & \downarrow & & & & \downarrow & & & & \downarrow & & & \\ = & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$
$$= 100111110010_2$$

Para praticar, verifique que $BA6_{16} = 101110100110_2$.

Conversão Binário-Hexadecimal

A conversão de binário para hexa é apenas o inverso do processo anterior. O número binário é reunido em grupos de *quatro* bits, e cada grupo é convertido para seu equivalente dígito hexa. Zeros são adicionados, se necessário, para completar um grupo de quatro bits (vide sombreado).

$$1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0_2 = \underbrace{0\ 0\ 1\ 1}_{3}\ \underbrace{1\ 0\ 1\ 0}_{A}\ \underbrace{0\ 1\ 1\ 0}_{6}$$
$$= 3A6_{16}$$

De modo a realizar estas conversões entre hexa e binário, é necessário saber a equivalência entre os números binários de quatro bits (0000 até 1111) e os dígitos hexa. Uma vez dominadas, as conversões podem ser realizadas rapidamente sem necessidade de cálculos. Isto explica por que o hexa (e o octal) são tão úteis na representação de números binários grandes.

Para praticar, verifique que $101011111_2 = 15F_{16}$.

Contando em Hexadecimal

Quando contamos em hexa, cada posição de dígito pode ser incrementada (aumentada de 1) de 0 até F. Uma vez que uma posição de dígito alcance o valor F, ela volta a 0, e a próxima posição de dígito é incrementada. Isto é ilustrado nas seguintes seqüências de contagem hexa:

(a) 38, 39, 3A, 3B, 3C, 3D, 3E, 3F, 40, 41, 42
(b) 6F8, 6F9, 6FA, 6FB, 6FC, 6FD, 6FE, 6FF, 700

Note que quando existe um 9 numa posição de dígito, ele se torna um A quando é incrementado.

Com N posições de dígitos hexa podemos contar de 0 a $16^N - 1$ em decimal, para um total de 16^N valores diferentes. Por exemplo, com três dígitos hexa podemos contar de 000_{16} até FFF_{16}, que é de 0_{10} até 4095_{10}, para um total de 4096 $= 16^3$ valores diferentes.

EXEMPLO 2-5

Converta o decimal 378 para um binário de 16 bits, primeiramente convertendo-o para hexa.

Solução

$$\frac{378}{16} = 23 + \text{resto de } 10$$

$$\frac{23}{16} = 1 + \text{resto de } 7$$

$$\frac{1}{16} = 0 + \text{resto de } 1$$

Logo, $378_{10} = 17A_{16}$. Este valor hexa pode ser facilmente convertido para o binário 000101111010. Finalmente, podemos expressar 378_{10} como um número binário de 16 bits adicionando-se quatro 0s à esquerda:

$$378_{10} = 0000\ \ 0001\ \ 0111\ \ 1010_2$$

EXEMPLO 2-6

Converta $B2F_{16}$ para octal.

Solução

É mais fácil primeiro converter de hexa para binário, e então para octal.

$B2F_{16}$	= 1011	0010	1111		{converter para binário}
	= 101	100	101	111	{reunir em grupo de três bits}
	= 5	4	5	7_8	{converter para octal}

Resumo das Conversões

Neste ponto, sua cabeça provavelmente está rodando enquanto você tenta guardar todos estes sistemas — binário, decimal, octal, hexa — e todas as diferentes conversões de um para o outro. Você pode não acreditar, mas à medida que você usar mais e mais estes vários sistemas, você acabará conhecendo-os muito bem. Por enquanto, o seguinte resumo deve ajudá-lo a fazer as diferentes conversões:

1. Quando converter de binário [ou octal ou hexa] para decimal, use o método da soma ponderada para cada posição de dígito.
2. Quando converter de decimal para binário [ou octal ou hexa], use o método das divisões sucessivas por 2 [ou 8 ou 16], agrupando os restos (Fig. 2-1).
3. Quando converter de binário para octal [ou hexa], reúna os bits em grupos de três [ou quatro] e converta cada grupo no dígito octal [ou hexa] correto.
4. Quando converter de octal [ou hexa] para binário, converta cada dígito para o seu equivalente de três [ou quatro] bits.
5. Quando converter de octal para hexa [ou vice-versa], primeiramente converta para binário; então converta o binário para o sistema de numeração desejado.

Questões de Revisão

1. Converta $24CE_{16}$ para decimal.
2. Converta 3117_{10} para hexa, e depois para binário.
3. Converta 1001011110110101_2 para hexa.
4. Escreva os próximos quatro números nesta seqüência de contagem hexa: E9A, E9B, E9C, E9D, _____, _____, _____, _____.
5. Converta 3527_8 para hexa.
6. Qual é a faixa de valores decimais que pode ser representada por um número hexa de quatro dígitos?

2-5 CÓDIGO BCD

Quando números, letras ou palavras são representados por um grupo especial de símbolos, dizemos que estão codificados, e o grupo de símbolos é chamado de *código*. Provavelmente um dos códigos mais conhecidos é o código Morse, em que uma série de traços e pontos representam as letras do alfabeto.

Já vimos que qualquer número decimal pode ser representado por um número binário equivalente. O grupo de 0s e 1s no número binário pode ser imaginado como um código representando o número decimal. Quando um número decimal é representado por seu número binário equivalente, denomina-se **codificação binária pura.**

Todos os sistemas digitais utilizam alguma forma de números binários para suas operações internas, mas o mundo exterior é decimal por natureza. Isto significa que conversões entre os sistemas decimal e binário são realizadas freqüentemente. Vimos que conversões entre decimal e binário podem se tornar longas e complicadas para números grandes. Por essa razão, um meio de codificar números decimais que combina algumas características tanto do sistema decimal quanto do sistema binário é usado em certas situações.

Código Decimal Codificado em Binário

Se *cada* dígito de um número decimal é representado por seu equivalente binário, o resultado é um código chamado **decimal codificado em binário** (daqui para a frente abreviado como BCD, do inglês Binary-Coded-Decimal). Como um dígito decimal pode assumir o valor 9, quatro bits são necessários para codificar cada dígito (o código binário para 9 é 1001).

Para ilustrar o código BCD, considere um número decimal como 874. Cada *dígito* é substituído pelo seu equivalente binário do seguinte modo:

8	7	4	(decimal)
↓	↓	↓	
1000	0111	0100	(BCD)

Como um outro exemplo, vamos transformar 943 para sua representação no código BCD:

9	4	3	(decimal)
↓	↓	↓	
1001	0100	0011	(BCD)

Mais uma vez, cada dígito decimal é trocado pelo seu binário equivalente puro. Note que *sempre* são usados quatro bits para cada dígito.

O código BCD, portanto, representa cada dígito do número decimal por um número binário de quatro bits. Obviamente apenas os números binários de quatro bits de 0000 até 1001 são usados. O código BCD não utiliza os números 1010, 1011, 1100, 1101, 1110 e 1111. Em outras palavras, somente 10 dos 16 grupos possíveis de quatro bits são usados. Se algum número de quatro bits "proibido" ocorrer numa máquina usando o código BCD, usualmente é uma indicação de que um erro aconteceu.

EXEMPLO 2-7

Converta 0110100000111001 (BCD) para seu equivalente decimal.

Solução

Divida o número BCD em grupos de quatro bits e converta cada um deles para decimal.

$$\underbrace{0110}_{6}\ \underbrace{1000}_{8}\ \underbrace{0011}_{3}\ \underbrace{1001}_{9}$$

EXEMPLO 2-8

Converta o número BCD 011111000001 para seu equivalente decimal.

Solução

$$\underbrace{0111}_{7}\ 1100\ \underbrace{0001}_{1}$$

↓ Este grupo de bits é proibido e indica um erro no número BCD.

Comparação entre BCD e Binário

É importante ressaltar que o BCD não é um outro sistema de numeração tal como o binário, o octal, o decimal ou o hexadecimal. Ele é, na verdade, um sistema decimal com cada dígito codificado no seu equivalente binário. Também é importante compreender que um número BCD *não* é o mesmo que um número binário puro. O código binário puro considera o número decimal *completo* e o representa em binário; o código BCD converte *cada dígito* decimal para binário individualmente. Para ilustrar, considere o número 137 e compare seus códigos binário puro e BCD:

$137_{10} = 10001001_2$ (binário)
$137_{10} = 0001\ 0011\ 0111$ (BCD)

O código BCD requer 12 bits, enquanto o código binário puro necessita de apenas 8 bits para representar 137. O BCD necessita de mais dígitos do que o binário puro para representar números decimais com mais de um dígito. Isto é porque o BCD não usa todos os grupos possíveis de quatro bits, conforme ressaltado anteriormente, e por isso é um tanto ineficiente.

A principal vantagem do código BCD é a relativa facilidade de conversão para o decimal e vice-versa. Apenas os códigos de quatro bits para os dígitos decimais de 0 até 9 precisam ser lembrados. Esta facilidade de conversão é especialmente importante sob o ponto de vista do hardware porque num sistema digital são os circuitos lógicos que realizam as conversões de e para decimal.

Questões de Revisão

1. Represente o valor decimal 178 pelo seu equivalente binário puro. Depois codifique o mesmo número usando BCD.
2. Quantos bits são necessários para representar um número decimal de oito dígitos em BCD?
3. Qual é a vantagem de codificar um número decimal em BCD quando comparado com binário puro? Qual é a desvantagem?

2-6 RELACIONANDO AS REPRESENTAÇÕES

A Tabela 2-3 mostra a representação dos números decimais de 0 até 15 nos sistemas de numeração binário, octal, hexadecimal e no código BCD. Examine-a cuidadosamente e esteja certo de compreender como ela foi obtida. Observe especialmente como a representação BCD sempre usa quatro bits para cada dígito decimal.

TABELA 2-3

Decimal	Binário	Octal	Hexadecimal	BCD
0	0	0	0	0000
1	1	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101

2-7 O BYTE

A maioria dos microcomputadores manipula e armazena dados binários e informações em grupos de oito bits; assim, um nome especial é dado para uma cadeia (ou seqüência) de oito bits: é o chamado **byte**. Um byte sempre corresponde a oito bits, e pode representar numerosos tipos de dados ou informações. Os exemplos seguintes ilustram isso.

EXEMPLO 2-9

Quantos bytes existem numa cadeia de 32 bits?

Solução

32/8 = 4, logo existem **quatro** bytes numa cadeia de 32 bits.

EXEMPLO 2-10

Qual é o maior valor decimal que pode ser representado em binário usando dois bytes?

Solução

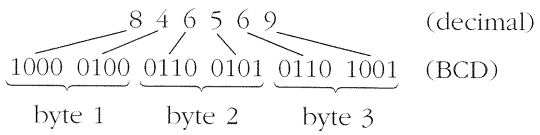
Dois bytes são 16 bits, logo o maior valor binário será equivalente ao decimal $2^{16} - 1 = 65.535$.

EXEMPLO 2-11

Quantos bytes são necessários para representar o valor decimal 846.569 em BCD?

Solução

Cada dígito decimal é convertido para um código BCD de quatro bits. Assim, um número decimal de seis dígitos requer 24 bits. Estes 24 bits equivalem a **três** bytes. Isto é ilustrado a seguir.



Questões de Revisão

1. Quantos bytes são necessários para representar 235_{10} em binário?
2. Qual é o maior valor decimal que pode ser representado em BCD usando-se dois bytes?

2-8 CÓDIGOS ALFANUMÉRICOS

Além de dados numéricos, um computador deve ser capaz de manipular informação não-numérica. Em outras palavras, um computador deve reconhecer códigos que representam letras do alfabeto, sinais de pontuação e outros caracteres especiais, como também os números. Estes códigos são chamados de **códigos alfanuméricos**. Um código alfanumérico completo deve incluir as 26 letras minúsculas, as 26 letras maiúsculas, 10 dígitos numéricos, 7 sinais de pontuação, e entre 20 e 40 outros caracteres, tais como +, /, #, %, *, e assim por diante.* Podemos dizer que um código alfanumérico representa todos os caracteres e funções encontrados num teclado de computador.

*N. do T.: Considerações feitas para a língua inglesa.

Código ASCII

O código alfanumérico mais amplamente usado é o **American Standard Code for Information Interchange (ASCII)**. O código ASCII (pronuncia-se "asquii") é um código de sete bits, e portanto tem $2^7 = 128$ codificações possíveis. Isto é mais do que suficiente para representar todos os caracteres de um teclado padrão, como também as funções de controle, tais como as funções de <RETURN> e <LINEFEED>. A Tabela 2-4 mostra uma listagem parcial do código ASCII. Além do código binário para cada caracter, a tabela apresenta os equivalentes octal e hexadecimal.

TABELA 2-4 Listagem parcial do código ASCII.

Caracter	ASCII de sete bits	Octal	Hexa
A	100 0001	101	41
B	100 0010	102	42
C	100 0011	103	43
D	100 0100	104	44
E	100 0101	105	45
F	100 0110	106	46
G	100 0111	107	47
H	100 1000	110	48
I	100 1001	111	49
J	100 1010	112	4A
K	100 1011	113	4B
L	100 1100	114	4C
M	100 1101	115	4D
N	100 1110	116	4E
O	100 1111	117	4F
P	101 0000	120	50
Q	101 0001	121	51
R	101 0010	122	52
S	101 0011	123	53
T	101 0100	124	54
U	101 0101	125	55
V	101 0110	126	56
W	101 0111	127	57
X	101 1000	130	58
Y	101 1001	131	59
Z	101 1010	132	5A
0	011 0000	060	30
1	011 0001	061	31
2	011 0010	062	32
3	011 0011	063	33
4	011 0100	064	34
5	011 0101	065	35
6	011 0110	066	36
7	011 0111	067	37
8	011 1000	070	38
9	011 1001	071	39
espaço	010 0000	040	20
.	010 1110	056	2E
(	010 1000	050	28
+	010 1011	053	2B
$	010 0100	044	24
*	010 1010	052	2A
)	010 1001	051	29
—	010 1101	055	2D
/	010 1111	057	2F
,	010 1100	054	2C
=	011 1101	075	3D
<RETURN>	000 1101	015	0D
<LINEFEED>	000 1010	012	0A

EXEMPLO 2-12

A mensagem a seguir é uma mensagem codificada em código ASCII. Qual é a mensagem?

1001000 1000101 1001100 1010000

Solução

Converta cada código de sete bits para seu hexa equivalente. Os resultados são

48 45 4C 50

Agora localize estes valores hexa na Tabela 2-4 e determine o caracter representado por cada um. Os resultados são

H E L P

O código ASCII é usado para a transferência de informações entre um computador e dispositivos de entrada e saída como terminais de vídeo e impressoras. Um computador também o utiliza internamente para armazenar informações que um operador digita no teclado. O exemplo seguinte ilustra isso.

EXEMPLO 2-13

Um operador está digitando um programa em BASIC no teclado de um certo microcomputador. O computador converte cada tecla digitada para o código ASCII e armazena o código como um byte na memória. Determine as cadeias de bits que serão armazenadas na memória quando o operador digitar o seguinte comando BASIC:

GOTO 25

Solução

Localize cada caracter (incluindo o espaço) na Tabela 2-4 e registre seu código ASCII.

G	01000111
O	01001111
T	01010100
O	01001111
(espaço)	00100000
2	00110010
5	00110101

Observe que um 0 foi adicionado para o bit mais à esquerda de cada código ASCII porque os códigos devem ser armazenados como bytes (oito bits). Este acréscimo de um bit extra é chamado *preenchimento com 0s.*

Questões de Revisão

1. Codifique a seguinte mensagem em código ASCII usando a representação hexa:

"COST = $72."

2. A seguinte mensagem em código ASCII preenchido está armazenada em posições de memória consecutivas em um computador:

01010011 01010100 01001111 01010000

Qual é a mensagem?

2-9 MÉTODO DA PARIDADE PARA DETECÇÃO DE ERROS

A movimentação de dados binários e de códigos de um lugar para outro é a operação mais freqüentemente realizada em sistemas digitais. Aqui estão alguns exemplos:

- A transmissão de voz digitalizada através de um enlace de microondas
- A gravação e recuperação de dados de dispositivos de memória externa como fitas e discos magnéticos
- A transmissão de informação de um computador para um terminal de um usuário remoto ou para outro computador através das linhas telefônicas (usando um modem)

Sempre que uma informação é transmitida de um dispositivo (o transmissor) para outro dispositivo (o receptor), existe a possibilidade de que erros ocorram de modo que o receptor não receba a informação idêntica àquela que foi enviada pelo transmissor. A causa principal de erros de transmissão são *ruídos elétricos*, que consistem em flutuações espúrias de tensão ou corrente que estão presentes em diferentes graus em todos os sistemas eletrônicos. A Fig. 2-2 é uma ilustração simples de um tipo de erro de transmissão.

O transmissor envia um sinal digital serial relativamente livre de ruídos através de uma linha de sinal para o receptor. Entretanto, quando o sinal atinge o receptor, ele contém um certo nível de ruído sobreposto ao sinal original. Ocasionalmente, o ruído é grande o suficiente em amplitude e altera o nível lógico do sinal como acontece no ponto *x*. Quando isso ocorre, o receptor pode interpretar incorretamente aquele bit como 1 lógico, que não foi o que o transmissor enviou.

A maioria dos equipamentos digitais modernos é projetada para ser relativamente livre de erros, e a probabilidade de erros como mostrado na Fig. 2-2 é muito baixa. Entretanto, devemos compreender que sistemas digitais freqüentemente transmitem milhares, ou mesmo milhões, de bits por segundo, e assim mesmo uma taxa de ocorrência de erros muito baixa pode produzir um erro ocasional que pode ser um incômodo, ou mesmo um desastre. Por essa razão, muitos sistemas digitais empregam algum método para detecção (e algumas vezes correção) de erros. Um dos esquemas mais simples e mais amplamente usados para a detecção de erros é o **método da paridade.**

Bit de Paridade

Um **bit de paridade** é um bit extra que é anexado ao grupo de bits do código que está sendo transferido de um lugar para outro. O bit de paridade é 0 ou 1, dependendo do número de 1s contido no grupo. Dois métodos diferentes são usados.

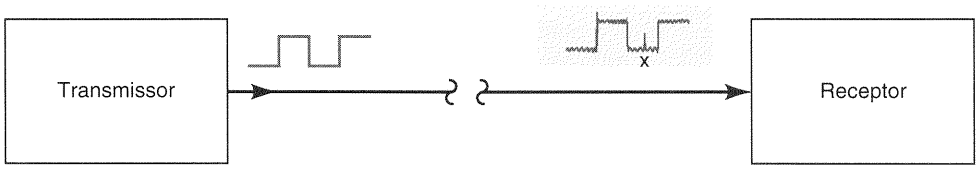


Fig. 2-2 Exemplo de ruído causando um erro na transmissão de dados digitais.

No método da *paridade par*, o valor do bit de paridade é escolhido de tal modo que o número total de 1s no grupo de bits do código (incluindo o bit de paridade) seja um número *par*. Por exemplo, suponha que o grupo é 1000011. Este é o caracter "C" em ASCII. Este grupo possui *três* 1s. Portanto, adicionamos um bit de paridade de 1 para fazermos o número total de 1s um número par. O *novo* grupo de bits do código, incluindo o bit de paridade, torna-se

1 1 0 0 0 0 1 1
↑ bit de paridade adicionado*

Se o grupo de bits do código contém inicialmente um número par de 1s, o bit de paridade assume o valor 0. Por exemplo, se o grupo for 1000001 (o código ASCII para "A"), o bit de paridade deve ser 0, e o novo código, incluindo o bit de paridade, deve ser 01000001.

O método da *paridade ímpar* é usado exatamente do mesmo modo, com exceção de que o bit de paridade é escolhido de tal maneira que o número total de 1s (incluindo o bit de paridade) seja um número *ímpar*. Por exemplo, para o grupo 1000001, o bit de paridade deve ser 1. Para o grupo 1000011, o bit de paridade deve ser 0.

Independentemente de ser usada paridade par ou paridade ímpar, o bit de paridade torna-se parte integrante da palavra de código. Por exemplo, adicionando-se um bit de paridade ao código ASCII de sete bits, produz-se um código de oito bits. Assim, o bit de paridade é tratado como qualquer outro bit no código.

O bit de paridade é usado para detectar qualquer erro de *apenas um bit* que ocorra durante a transmissão de um código de um lugar para outro (e. g., de um computador para um terminal de vídeo). Por exemplo, suponha que o caracter "A" esteja sendo transmitido e que a paridade *ímpar* esteja sendo usada. O código transmitido deveria ser

1 1 0 0 0 0 0 1

Quando o circuito receptor receber esse código, ele verificará que o código contém um número ímpar de 1s (incluindo o bit de paridade). Se for este o caso, o receptor suporá que o código foi recebido corretamente. Agora, suponha que devido a algum ruído ou mau funcionamento o receptor receba o seguinte código:

1 1 0 0 0 0 0 0

O receptor constatará que esse código tem um número *par* de 1s. Isto revela ao receptor que deve haver um erro no código, já que presumivelmente o transmissor e o receptor tinham concordado em usar paridade ímpar. Não existe maneira, entretanto, de o receptor indicar qual bit está com erro, já que ele não sabe que código deveria ser.

*O bit de paridade pode ser colocado no início ou no fim do grupo, mas normalmente é colocado à esquerda do MSB.

Deveria ficar claro que este método da paridade não funcionaria se *dois* bits estivessem errados, porque dois erros não mudariam a paridade par ou ímpar do código. Na prática, o método da paridade é usado apenas em situações em que a probabilidade de erros simples é muito baixa e a probabilidade de erros duplos é essencialmente zero.

Quando o método da paridade está sendo usado, o transmissor e o receptor devem concordar, antecipadamente, se será usada a paridade par ou a paridade ímpar. Não existe vantagem de uma sobre a outra, embora a paridade par pareça ser usada mais freqüentemente. O transmissor deve anexar um bit de paridade apropriado para cada unidade de informação que transmite. Por exemplo, se o transmissor está enviando dados codificados em ASCII, ele anexa um bit de paridade para cada grupo do código ASCII de sete bits. Quando o receptor examinar os dados que recebeu do transmissor, ele verificará cada grupo de bits do código para constatar que o número total de 1s (incluindo o bit de paridade) é compatível com o tipo de paridade acordado previamente. Isto é comumente denominado de *verificação da paridade* dos dados. Na circunstância de ele detectar um erro, o receptor pode enviar uma mensagem de volta ao transmissor solicitando a retransmissão do último conjunto de dados. O procedimento exato que é seguido quando um erro é detectado dependerá do projeto do sistema em particular.

EXEMPLO 2-14

Computadores freqüentemente se comunicam com outros computadores remotos através de linhas telefônicas. Por exemplo, é assim que a comunicação pela Internet acontece. Quando um computador está transmitindo uma mensagem para outro, a informação é usualmente codificada em ASCII. Qual é a cadeia de bits real que um computador transmite para enviar a mensagem HELLO, usando ASCII com paridade par?

Solução

Primeiro procure o código ASCII para cada caracter da mensagem. Então para cada código conte o número de 1s. Se for um número par, anexe um 0 como MSB. Se for um número ímpar, anexe um 1. Assim, todos os códigos de oito bits (bytes) resultantes terão um número par de 1s (incluindo a paridade).

	bits de paridade par anexados
H-	0 1 0 0 1 0 0 0
E-	1 1 0 0 0 1 0 1
L-	1 1 0 0 1 1 0 0
L-	1 1 0 0 1 1 0 0
O-	1 1 0 0 1 1 1 1

Questões de Revisão

1. Anexe o bit de paridade ímpar para o código ASCII do símbolo $ e expresse o resultado em hexadecimal.
2. Anexe o bit de paridade par ao código BCD do número decimal 69.
3. Por que o método da paridade não pode detectar um erro duplo nos dados transmitidos?

2-10 REVISÃO

A título de revisão, aqui estão alguns exemplos a mais para ilustrar as operações apresentadas neste capítulo.

EXEMPLO 2-15

(a) Converta o decimal 135 para binário.

$$\frac{135}{2} = 67 + R1$$
$$\frac{67}{2} = 33 + R1$$
$$\frac{33}{2} = 16 + R1$$
$$\frac{16}{2} = 8 + R0$$
$$\frac{8}{2} = 4 + R0$$
$$\frac{4}{2} = 2 + R0$$
$$\frac{2}{2} = 1 + R0$$
$$\frac{1}{2} = 0 + R1$$

$$1\ 0\ 0\ 0\ 0\ 1\ 1\ 1_2$$

(b) Converta o decimal 76 para octal.

$$\frac{76}{8} = 9 + R4$$
$$\frac{9}{8} = 1 + R1$$
$$\frac{1}{8} = 0 + R1$$

$$1\ 1\ 4_8$$

(c) Converta o decimal 541 para hexadecimal.

$$\frac{541}{16} = 33 + R13$$
$$\frac{33}{16} = 2 + R1$$
$$\frac{2}{16} = 0 + R2$$

$$2\ 1\ D_{16}$$

(d) Converta o decimal 479 para BCD.

$$\underbrace{0100}_{4}\ \underbrace{0111}_{7}\ \underbrace{1001}_{9}\ \text{BCD}$$

(e) Converta o binário 101101 para decimal.

$$101101_2 = 1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$$
$$= 32 + 8 + 4 + 1$$
$$= 45_{10}$$

(f) Converta o octal 6254 para decimal.

$$6254_8 = 6\times8^3 + 2\times8^2 + 5\times8^1 + 4\times8^0$$
$$= 6\times512 + 2\times64 + 5\times8 + 4\times1 = 3244_{10}$$

(g) Converta o hexa 1A3F para decimal.

$$1A3F = 1\times16^3 + 10\times16^2 + 3\times16^1 + 15\times16^0$$
$$= 4096 + 2560 + 48 + 15 = 6719_{10}$$

(h) Converta 010010010110 (BCD) para decimal.

$$\underbrace{0100}_{4}\ \underbrace{1001}_{9}\ \underbrace{0110}_{6}\ \text{(BCD)}$$
$$4\ \ 9\ \ 6_{10}$$

(i) Converta o binário 10110111 para octal e para hexa.

$$\underbrace{010}_{2}\ \underbrace{110}_{6}\ \underbrace{111}_{7_8}$$
$$\underbrace{1011}_{B}\ \underbrace{0111}_{7_{16}}$$

(j) Converta o hexadecimal E61 para binário.

$$\underbrace{1110}_{E}\ \underbrace{0110}_{6}\ \underbrace{0001}_{1}{}_2$$

(k) Converta o octal 724 para binário.

$$\overbrace{111}^{7}\ \overbrace{010}^{2}\ \overbrace{100}^{4}{}_2$$

(l) Adicione o bit de paridade ímpar no código ASCII de "Z".

Da Tabela 2-4, o código para "Z" é 1011010. O número de 1s neste grupo é quatro, um número par. Portanto, para achar a paridade ímpar, temos que anexar um 1 como bit de paridade (MSB) como segue:

$$1\,1011010$$

Observe que o grupo de bits do código completo — incluindo o bit de paridade — agora tem um número ímpar de 1s.

RESUMO

1. Os sistemas de numeração octal e hexadecimal são usados em sistemas digitais e computadores como modos eficientes de representar quantidades binárias.
2. Em conversões entre octal e binário, um dígito octal corresponde a três bits. Em conversões entre hexa e binário, cada dígito hexa corresponde a quatro bits.
3. O método das divisões sucessivas é usado para converter números decimais para binário, octal ou hexadecimal.
4. Usando um número binário de N bits, podemos representar valores decimais de 0 até $2^N - 1$.
5. O código BCD para um número decimal é formado convertendo-se cada dígito do número decimal para o seu equivalente binário de quatro bits.
6. Um byte é uma cadeia de oito bits.
7. Um código alfanumérico usa grupos de bits para representar todos os vários caracteres e funções que fazem parte de um típico teclado de computador. O código ASCII é o código alfanumérico mais amplamente usado.
8. O método da paridade para detecção de erros anexa um bit de paridade especial para cada grupo de bits transmitido.

TERMOS IMPORTANTES*

sistema de numeração octal
sistema de numeração hexadecimal
codificação binária pura
decimal codificado em binário (código BCD)
byte
código alfanumérico
American Standard Code for Information Interchange (ASCII)
método da paridade
bit de paridade

PROBLEMAS

SEÇÕES 2-1 E 2-2

2-1. Converta os seguintes números binários para decimal.
(a) 10110 (d) 1111010111
(b) 10001101 (e) 10111111
(c) 100100001001

2-2. Converta os seguintes valores decimais para binário.
(a) 37 (d) 205
(b) 14 (e) 2313
(c) 189 (f) 511

2-3. Qual é o maior valor decimal que pode ser representado por um número binário de oito bits? E por um de 16 bits?

SEÇÃO 2-3

2-4. Converta cada número octal para seu equivalente decimal.
(a) 743 (d) 257
(b) 36 (e) 1204
(c) 3777

2-5. Converta cada um dos seguintes números decimais para octal.
(a) 59 (d) 65.536
(b) 372 (e) 255
(c) 919

2-6. Converta cada um dos valores octais do Problema 2-4 para binário.

2-7. Converta os números binários do Problema 2-1 para octal.

2-8. Relacione os números octais em seqüência desde 165_8 até 200_8.

2-9. Quando um número decimal grande tem que ser convertido para binário, às vezes é mais fácil convertê-lo primeiro para octal e depois de octal para binário. Tente este procedimento para 2313_{10} e compare com o procedimento usado no Problema 2-2 (e).

2-10. Quantos dígitos octais são necessários para representar números decimais até 20.000?

SEÇÃO 2-4

2-11. Converta os seguintes valores hexadecimais em decimal.
(a) 92 (d) 2C0
(b) 1A6 (e) 7FF
(c) 37FD

2-12. Converta os seguintes valores decimais para hexa.
(a) 75 (d) 25.619
(b) 314 (e) 4095
(c) 2048

2-13. Converta os números binários do Problema 2-1 para hexadecimal.

2-14. Converta os valores em hexa do Problema 2-11 para binário.

2-15. Liste os números hexadecimais em seqüência desde 280 até 2A0.

2-16. Quantos dígitos hexadecimais são necessários para representar números decimais até um milhão?

SEÇÃO 2-5

2-17. Codifique estes números decimais em BCD.
(a) 47 (d) 42.689.627
(b) 962 (e) 1204
(c) 187

2-18. Quantos bits são necessários para representar os números decimais na faixa de 0 até 999 usando-se a codificação binária pura? E usando o código BCD?

2-19. Os números seguintes estão em BCD. Converta-os para decimal.
(a) 1001011101010010 (c) 0111011101110101
(b) 000110000100 (d) 010010010010

SEÇÃO 2-7

2-20. (a) Quantos bits estão contidos em oito bytes?
(b) Qual é o maior número hexa que pode ser representado em quatro bytes?
(c) Qual é o maior valor decimal codificado em BCD que pode ser representado em três bytes?

SEÇÕES 2-8 E 2-9

2-21. Represente a instrução "X = 25/Y" no código ASCII (excluindo as aspas). Anexe o bit de paridade ímpar.

2-22. Anexe o bit de paridade *par* para cada um dos códigos ASCII do Problema 2-21 e forneça os resultados em hexa.

2-23. Os bytes a seguir (em hexadecimal) representam o nome de uma pessoa do modo como deveriam estar armazenados na memória de um computador. Cada byte é um código ASCII preenchido. Determine o nome da pessoa.

42 45 4E 20 53 4D 49 54 48

*Estes termos podem ser encontrados em negrito no capítulo e estão definidos no Glossário ao final do livro.

2-24. Converta os seguintes números decimais para o código BCD e depois anexe um bit de paridade *ímpar*.
(a) 74 (c) 165
(b) 38 (d) 9201

2-25. Em determinado sistema digital, os números decimais de 000 até 999 são representados no código BCD. Um bit de paridade *ímpar* também é incluído no fim de cada grupo de bits do código. Examine cada um dos grupos a seguir e admita que eles acabaram de ser transferidos de um lugar para outro. Alguns grupos contêm erros. Suponha que *não mais do que* dois erros ocorreram para cada grupo. Determine quais os grupos do código têm um erro apenas e quais deles têm *definitivamente* um erro duplo. (Sugestão: Lembre-se de que é utilizado o código BCD.)

(a) 1001010110000
↑ bit de paridade
(b) 0100011101100
(c) 0111110000011
(d) 1000011000101

2-26. Suponha que o receptor recebeu os seguintes dados a seguir do transmissor do Exemplo 2-14:

0 1 0 0 1 0 0 0
1 1 0 0 0 1 0 1
1 1 0 0 1 1 0 0
1 1 0 0 1 0 0 0
1 1 0 0 1 1 0 0

Que erros o receptor pode detectar nestes dados recebidos?

QUESTÕES DE FIXAÇÃO

2-27. Realize cada uma das conversões a seguir. Para algumas delas, você pode querer tentar vários métodos para ver qual deles é melhor para você. Por exemplo, uma conversão binário-decimal pode ser feita diretamente, ou pode ser feita como uma conversão binário-octal seguida de uma conversão octal-decimal.
(a) 1417_{10} = _____ $_2$
(b) 255_{10} = _____ $_2$
(c) 11010001_2 = _____ $_{10}$
(d) 1110101000100111_2 = _____ $_{10}$
(e) 2497_{10} = _____ $_8$
(f) 511_{10} = _____ $_8$
(g) 235_8 = _____ $_{10}$
(h) 4316_8 = _____ $_{10}$
(i) $7A9_{16}$ = _____ $_{10}$
(j) $3E1C_{16}$ = _____ $_{10}$
(k) 1600_{10} = _____ $_{16}$
(l) 38.187_{10} = _____ $_{16}$
(m) 865_{10} = _____ (BCD)
(n) 100101000111 (BCD) = _____ $_{10}$
(o) 465_8 = _____ $_{16}$
(p) $B34_{16}$ = _____ $_8$
(q) 01110100 (BCD) = _____ $_2$
(r) 111010_2 = _____ (BCD)

2-28. Represente o valor decimal 37 em cada um dos seguintes modos.
(a) binário puro
(b) BCD
(c) hexa
(d) ASCII (isto é, trate cada dígito como um caracter)
(e) octal

2-29. Complete os espaços com a(s) palavra(s) correta(s).
(a) A conversão de decimal para _____ requer sucessivas divisões por 8.
(b) A conversão de decimal para hexadecimal requer sucessivas divisões por _____.
(c) No código BCD, cada _____ é convertido para o seu binário equivalente de quatro bits.
(d) Um transmissor anexa um _____ num grupo do código para permitir ao receptor detectar _____.
(e) O código _____ é o código alfanumérico mais comum usado em sistemas computacionais.
(f) _____ e _____ são freqüentemente utilizados como um modo conveniente de representar números binários grandes.
(g) Uma cadeia de oito bits é denominada um _____.

2-30. Escreva o número binário que resulta quando cada um dos números a seguir é incrementado de um.
(a) 0111
(b) 010000
(c) 1110

2-31. Repita o Problema 2-30 para a operação de decremento.

2-32. Escreva o número que resulta quando cada um dos valores a seguir é incrementado.
(a) 7777_8 (d) 2000_{16}
(b) 7777_{16} (e) $9FF_{16}$
(c) 2000_8 (f) 1000_{16}

2-33. Repita o Problema 2-32 para a operação de decremento.

EXERCÍCIOS DESAFIADORES

2-34. Em um microcomputador os *endereços* das posições de memória são números binários que identificam cada circuito de memória onde um byte está armazenado. O número de bits que compõem um endereço depende de quantas posições de memória existem. Como o número de bits pode ser muito grande, os endereços são freqüentemente especificados em hexa em vez de binário.
(a) Se um microcomputador usa um endereço de 20 bits, quantas posições de memória diferentes existem?
(b) Quantos dígitos hexadecimais são necessários para representar o endereço de uma posição de memória?
(c) Qual é o endereço hexa da 256ª posição de memória?
(*Nota*: o primeiro endereço é sempre 0.)

2-35. Em um CD de áudio, o sinal de tensão do áudio é geralmente amostrado cerca de 44.000 vezes por segundo, e o valor de cada amostra é gravado na superfície do CD como um número binário. Em outras palavras, cada número binário gravado representa o valor da tensão em um ponto da forma de onda do sinal de áudio.
(a) Se os números binários têm seis bits de tamanho, quantos valores diferentes de tensão podem ser representados por apenas um número binário? Repita para oito bits e para dez bits.
(b) Se números de dez bits são usados, quantos bits devem ser gravados num CD em 1 segundo?
(c) Se um CD pode geralmente armazenar 5 bilhões de bits, quantos segundos de áudio podem ser gravados quando números de dez bits são usados?

2-36. Uma câmera eletrônica em preto-e-branco constrói um reticulado sobre uma imagem e então mede e grava um número binário representando o nível de cinza que ela detecta em cada célula do reticulado. Por exemplo, se números de quatro bits são usados, o valor do preto é ajustado para 0000 e o valor do branco para 1111, e qualquer nível de cinza fica entre 0000 e 1111. Se números de seis bits são usados, o preto é 000000, o branco é 111111, e todos os cinzas estão entre eles.

Suponha que queiramos distinguir entre 254 níveis diferentes de cinza em cada célula do reticulado. Quantos bits teríamos que usar para representar esses níveis?

2-37. Construa uma tabela mostrando as representações binária, octal, hexadecimal e BCD para todos os números decimais de 0 até 15. Compare sua tabela com a Tabela 2-3.

RESPOSTAS PARA AS QUESTÕES DE REVISÃO DAS SEÇÕES

SEÇÃO 2-1
1. 2267
2. 32768

SEÇÃO 2-2
1. 1010011
2. 1011011001
3. 20 bits

SEÇÃO 2-3
1. 396
2. 222; 010010010
3. 235
4. 627, 630, 631
5. 1111001111
6. 699
7. 0 até 4095

SEÇÃO 2-4
1. 9422
2. C2D; 110000101101
3. 97B5
4. E9E, E9F, EA0, EA1
5. 757
6. 0 até 65.535

SEÇÃO 2-5
1. 10110010_2; 000101111000 (BCD)
2. 32
3. Vantagem: facilidade de conversão. Desvantagem: BCD requer mais bits.

SEÇÃO 2-7
1. Dois
2. 99

SEÇÃO 2-8
1. 43, 4F, 53, 54, 20, 3D, 20, 24, 37, 32
2. STOP

SEÇÃO 2-9
1. A4
2. 001101001
3. Dois erros nos dados não mudariam a paridade par ou ímpar dos dados.