E

Sistemas de numeração

Aqui estão apenas números ratificados.
—William Shakespeare

A natureza tem algum tipo de sistema de coordenadas geométrico-aritmético, porque a natureza tem todos os tipos de modelos. O que experimentamos de natureza está em modelos, e todos os modelos da natureza são belos. Surpreende-me que o sistema da natureza tenha tal beleza, uma vez que em química descobrimos que as associações ocorrem sempre em belos números inteiros — não há nenhuma fração.
—Richard Buckminster Fuller

OBJETIVOS

Neste capítulo, você aprenderá:

- Entender conceitos básicos de sistemas de numeração como base, valor posicional e valor de símbolo.
- Entender como trabalhar com números representados nos sistemas de numeração binário, octal e hexadecimal.
- Abreviar números binários como números octais ou hexadecimais.
- Converter números octais e hexadecimais em números binários.
- Converter nos dois sentidos entre números decimais e seus equivalentes binários, octais e hexadecimais.
- Entender a aritmética binária e como os números binários negativos são representados utilizando a notação de complemento de dois.

Sumário

E.1 Introdução

Neste apêndice, introduzimos os principais sistemas de numeração que os programadores de Java utilizam especialmente quando trabalham em projeto de software que requer íntima interação com hardware no nível de máquina. Projetos como esse incluem sistemas operacionais, software de rede de computador, compiladores, sistemas de banco de dados e aplicativos que requerem alto desempenho.

Quando escrevemos um inteiro como 227 ou –63 em um programa Java, o número é considerado como estando no sistema de numeração decimal (base 10). Os dígitos no sistema de numeração decimal são 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. O dígito mais baixo é 0 e o mais alto, 9 — um a menos que a base 10. Internamente, os computadores utilizam o sistema de numeração binário (base 2). O sistema de numeração binário tem apenas dois dígitos, 0 e 1. Seu dígito mais baixo é 0 e o mais alto é 1 — um a menos que a base 2.

Como veremos, números binários tendem a ser muito mais longos que seus equivalentes decimais. Programadores que trabalham em linguagens assembly e em linguagens de alto nível como Java, que permite que programadores alcancem o nível de máquina, acham desagradável trabalhar com números binários. Então dois outros sistemas de numeração, o octal (base 8) e o hexadecimal (base 16), são populares principalmente porque tornam conveniente abreviar números binários.

No sistema de numeração octal, os dígitos variam de 0 a 7. Uma vez que o sistema de numeração binário e o octal têm menos dígitos que o decimal, seus dígitos são os mesmos que os correspondentes em decimal.

O sistema de numeração hexadecimal apresenta um problema uma vez que requer 16 dígitos — um mais baixo, o dígito 0, e um mais alto, o dígito com um valor equivalente ao decimal 15 (um a menos que a base 16). Por convenção, utilizamos as letras A a F para representar os dígitos hexadecimais correspondentes aos valores decimais de 10 a 15. Portanto, em hexadecimal, podemos ter números como 876 consistindo unicamente em dígitos do tipo decimal, números como 8A55F consistindo em dígitos e letras, e números como FFE consistindo unicamente em letras. Ocasionalmente, um número hexadecimal forma uma palavra comum como FACE ou FOOD — isso pode parecer estranho para programadores acostumados a trabalhar com números. Os dígitos dos sistemas de numeração binário, octal, decimal e hexadecimal estão resumidos nas figuras E.1 e E.2.

Cada um desses sistemas de numeração utiliza notação posicional — cada posição na qual um dígito é escrito tem um valor posicional diferente. Por exemplo, no número decimal 937 (o 9, o 3 e o 7 são referidos como valores de símbolo), dizemos que o 7 é escrito na posição das unidades, o 3 é escrito na posição das dezenas e o 9 é escrito na posição das centenas. Observe que cada uma dessas posições é uma potência da base (base 10) e que essas potências iniciam em 0 e aumentam por 1 à medida que nos movemos para a esquerda no número (Figura E.3).

Dígito binário	Dígito octal	Dígito decimal	Dígito hexadecimal
0	0	0	0
1	1	1	1
	2	2	2
	3	3	3
	4	4	4
	5	5	5
	6	6	6
	7	7	7
		8	8
		9	9
			A (valor decimal de 10)
			B (valor decimal de 11)
			C (valor decimal de 12)
			D (valor decimal de 13)
			E (valor decimal de 14)
			F (valor decimal de 15)

Figura E.1 Os dígitos dos sistemas de numeração binário, octal decimal e hexadecimal.

Atributo	Binário	Octal	Decimal	Hexadecimal
Base	2	8	10	16
Dígito mais baixo	0	0	0	0
Dígito mais alto	1	7	9	F

Figura E.2 Comparando os sistemas de numeração binário, octal, decimal e hexadecimal.

Valores posicionais no sistema de numeração decimal			
Dígito decimal	9	3	7
Posicione nome	centenas	dezenas	unidades
Valor posicional	100	10	1
Valor posicional como potência da base (10)	10^2	10^1	10^0

Figura E.3 Valores posicionais no sistema de numeração decimal.

Para números decimais mais longos, as próximas posições à esquerda seriam a dos milhares (10 à terceira potência), a da dezena de milhares (10 à quarta potência), a das centenas de milhares (10 à quinta potência), a dos milhões (10 à sexta potência), a das dezenas de milhões (10 à sétima potência) e assim por diante.

No número binário 101, o 1 mais à direita é escrito na posição das unidades, o 0 é escrito na posição dos 2s e o 1 mais à esquerda é escrito na posição dos 4s. Observe que cada posição é uma potência da base (base 2) e que essas potências iniciam em 0 e aumentam por 1 à medida que nos movemos à esquerda no número (Figura E.4). Portanto, $101 = 2^2 + 2^0 = 4 + 1 = 5$.

Para números binários mais longos, as próximas posições à esquerda seriam a dos 8s (2 elevado a 3), a dos 16s (2 elevado a 4), a dos 32s (2 elevado a 5), a dos 64s (2 elevado a 6) e assim por diante.

No número octal 425, dizemos que o 5 é escrito na posição das unidades, o 2 na posição dos 8s e o 4 na posição dos 64s. Observe que cada uma dessas posições representa uma potência da base (base 8) e que essas potências iniciam em 0 e aumentam por 1 quando movemos à esquerda no número (Figura E.5)

Para números octais mais longos, as próximas posições à esquerda seriam a dos 512s (8 elevado a 3), a dos 4096s (8 elevado a 4), a dos 32706s (8 elevado a 5) e assim por diante.

No número hexadecimal 3DA, dizemos que A é escrito na posição das unidades, o D na posição dos 16s e o 3 na posição dos 256s. Observe que cada uma dessas posições representa uma potência da base (base 16) e que essas potências iniciam em 0 e aumentam por 1 à medida que nos movemos à esquerda no número (Figura E.6)

Para números hexadecimais mais longos, as próximas posições à esquerda seriam a dos 4096(s) (16 elevado a 3), a dos 64536s (16 elevado a 4) e assim por diante.

Valores posicionais no sistema de numeração binário			
Dígito binário	1	0	1
Posicione nome	4s	2s	unidades
Valor posicional	4	2	1
Valor posicional como potência da base (2)	2^2	2^1	2^0

Figura E.4 Valores posicionais no sistema de numeração binário.

Valores posicionais no sistema de numeração octal			
Dígito decimal	4	2	5
Posicione nome	64s	8s	unidades
Valor posicional	64	8	1
Valor posicional como potência da base (8)	8^2	8^1	8^0

Figura E.5 Valores posicionais no sistema de numeração octal.

Valores posicionais no sistema de numeração hexadecimal			
Dígito decimal	3	D	A
Posicione nome	256s	16s	unidades
Valor posicional	256	16	1
Valor posicional como potência da base (16)	16^2	16^1	16^0

Figura E.6 Valores posicionais no sistema de numeração hexadecimal.

E.2 Abreviando números binários como octais e hexadecimais

A principal utilização para números hexadecimal e octal em computação é abreviar longas representações binárias. A Figura E. 7 destaca o fato de que números binários longos podem ser expressos concisamente em sistemas de numeração com bases mais altas que o sistema de numeração binário.

Número decimal	Representação binária	Representaçãooctal	Representaçãohexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
n0 9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Figura E.7 Equivalentes decimais, binários, octais e hexadecimais.

Um relacionamento particularmente importante que tanto o sistema de numeração octal como o hexadecimal têm com o binário é que as bases octal e hexadecimal (8 e 16, respectivamente) são potências da base do sistema de numeração binário (base 2). Considere o seguinte número binário de 12 algarismos e seus equivalentes octal e hexadecimal. Veja se você pode determinar como esse relacionamento torna conveniente a abreviação de número binário em octal ou hexadecimal. A resposta segue os números.

```
Número binário  Equivalente octal   Equivalente hexadecimal
100011010001    4321                8D1
```

Para ver como o número binário é facilmente convertido em octal, divida o número binário de 12 dígitos em grupos de três bits consecutivos cada e escreva esses grupos sobre os dígitos correspondentes do número octal como a seguir

```
100   011   010   001
4     3     2     1
```

Observe que o dígito octal que você escreveu sob cada grupo de bits corresponde precisamente ao equivalente octal desse número binário de 3 dígitos como mostrado na Figura E.7.

O mesmo tipo de relacionamento pode ser observado ao converter-se de binário para hexadecimal. Divida o número binário de 12 algarismos em grupos de quatro bits consecutivos cada e escreva esses grupos sobre os dígitos correspondentes do número hexadecimal como a seguir:

```
1000    1101    0001
8       D       1
```

Observe que o dígito hexadecimal que você escreveu sob cada grupo de quatro bits corresponde precisamente ao equivalente de hexadecimal do número binário de 4 algarismos como mostrado na Figura E.7.

E.3 Convertendo números octais e hexadecimais em binários

Na seção anterior, vimos como converter números binários em seus equivalentes octal e hexadecimal formando grupos de dígitos binários e simplesmente reescrevendo-os como seus valores equivalentes de dígito octal ou de dígito hexadecimal. Esse processo pode ser utilizado no sentido inverso para produzir o equivalente binário de determinado número octal ou hexadecimal.

Por exemplo, o número octal 653 é convertido em binário escrevendo-se o 6 como seu equivalente binário de 3 dígitos 110, o 5 como seu equivalente binário de 3 dígitos 101, e o 3 como seu equivalente binário de 3 dígitos 011 para formar o número binário de 9 dígitos 110101011.

O número hexadecimal FAD5 é convertido em binário escrevendo-se o F como seu equivalente binário de 4 dígitos 1111, o A como seu equivalente binário de 4 dígitos 1010, o D como seu equivalente binário de 4 dígitos 1101, e o 5 como seu equivalente binário de 4 dígitos 0101 para formar o número de 16 dígitos 1111101011010101.

E.4 Convertendo de octal, binário ou hexadecimal para decimal

Estamos acostumados a trabalhar em decimal e, portanto, torna-se conveniente converter um número octal, binário ou hexadecimal em decimal para obter um sentido de quanto o número 'realmente' vale. Nossos diagramas na Seção E.1 expressam os valores posicionais em decimal. Para converter um número em decimal por meio de outra base, multiplique o equivalente decimal de cada dígito por seu valor posicional e some esses produtos. Por exemplo, o número binário 110101 é convertido no decimal 53 como mostra a Figura E.8.

Para convertermos o octal 7614 no decimal 3980, utilizamos a mesma técnica, dessa vez usando valores posicionais octais apropriados, como mostra a Figura E.9.

Para convertermos hexadecimal AD3B em decimal 44347, utilizamos a mesma técnica, dessa vez usando valores posicionais hexadecimais apropriados como mostra a Figura E.10.

E.5 Convertendo de decimal para octal, binário ou hexadecimal

As conversões na Seção E.4 seguem-se naturalmente das convenções de notação posicional. A conversão de decimal em octal, binário ou hexadecimal também segue essas convenções.

Convertendo um número binário em decimal						
Valores posicionais:	32	16	8	4	2	1
Valores de símbolo:	1	1	0	1	0	1
Produtos:	1*32=32	1*16=16	0*8=0	1*4=4	0*2=0	1*1=1
Soma:	= 32 + 16 + 0 + 4 + 0s + 1 = 53					

Figura E.8 Convertendo um número binário em decimal.

Convertendo um número octal em decimal				
Valores posicionais:	512	64	8	1
Valores de símbolo:	7	6	1	4
Produtos	7*512=3584	6*64=384	1*8=8	4*1=4
Soma:	= 3584 + 384 + 8 + 4 = 3980			

Figura E.9 Convertendo um número octal em decimal.

Convertendo um número hexadecimal em decimal				
Valores posicionais:	4096	256	16	1
Valores de símbolo:	A	D	3	B
Produtos	A*4096=40960	D*256=3328	3*16=48	B*1=11
Soma:	= 40960 + 3328 + 48 + 11 = 44347			

Figura E.10 Convertendo um número hexadecimal em decimal.

Suponha que quiséssemos converter o decimal 57 em binário. Iniciamos escrevendo os valores posicionais das colunas da direita para a esquerda até alcançarmos uma coluna cujo valor posicional seja maior que o número decimal. Não precisamos dessa coluna, então a descartamos. Portanto, primeiro escrevemos:

Valores posicionais: 64 32 16 8 4 2 1

Então descartamos a coluna com valor posicional 64, deixando:

Valores posicionais: 32 16 8 4 2 1

Em seguida trabalhamos da coluna mais à esquerda para a direita. Dividimos 32 em 57 e observamos que há um 32 em 57 com um resto de 25, então escrevemos 1 na coluna 32. Dividimos 16 em 25 e observamos que há um 16 em 25 com um resto de 9 e escrevemos 1 na coluna 16. Dividimos 8 em 9 e observamos que há um 8 em 9 com um resto de 1. Cada uma das duas próximas colunas produz cocientes de 0 quando seus valores posicionais são divididos em 1, portanto escrevemos 0s nas colunas 4 e 2. Por fim, 1 em 1 é 1, então escrevemos 1 na coluna 1. Isso resulta em:

Valores posicionais:	32	16	8	4	2	1
Valores de símbolo:	1	1	1	0	0	1

E assim o decimal 57 é equivalente ao binário 111001.

Para converter o decimal 103 em octal, iniciamos escrevendo os valores posicionais das colunas até alcançarmos uma coluna cujo valor posicional seja maior que o número decimal. Não precisamos dessa coluna, então a descartamos. Portanto, primeiro escrevemos:

```
Valores posicionais:   512   64   8   1
```

Então descartamos a coluna com valor posicional 512, o que resulta em:

```
Valores posicionais:         64   8   1
```

Em seguida trabalhamos da coluna mais à esquerda para a direita. Dividimos 64 em 103 e observamos que há um 64 em 103 com um resto de 39, então escrevemos 1 na coluna 32. Dividimos 8 em 39 e observamos que há quatro 8s em 39 com um resto de 7 e escrevemos 4 na coluna 8. Por fim, dividimos 1 em 7 e observamos que há sete 1s em 7 sem resto e, então, escrevemos 7 na coluna 1. Isso resulta em:

```
Valores posicionais:   64   8   1
Valores de símbolo:    1    4   7
```

E assim o decimal 103 é equivalente ao octal 147.

Para converter o decimal 375 em hexadecimal, iniciamos escrevendo os valores posicionais das colunas até alcançarmos uma coluna cujo valor posicional seja maior que o número decimal. Não precisamos dessa coluna, então a descartamos. Portanto, primeiro escrevemos:

```
Valores posicionais:   4096   256   16   1
```

Então descartamos a coluna com valor posicional 4096, o que resulta em:

```
Valores posicionais:          256   16   1
```

Em seguida trabalhamos da coluna mais à esquerda para a direita. Dividimos 256 em 375 e observamos que há um 256 em 375 com um resto de 119, então escrevemos 1 na coluna 256. Dividimos 16 em 119 e observamos que há sete 16s em 119 com um resto de 7 e escrevemos 7 na coluna 16. Por fim, dividimos 1 em 7 e observamos que há sete 1s em 7 sem resto, portanto então escrevemos 7 na coluna 1. Isso resulta em:

```
Valores posicionais:          256   16   1
Valores de símbolo:           1     7    7
```

E assim o decimal 375 é equivalente ao hexadecimal 177.

E.6 Números binários negativos: notação de complemento de dois

A discussão até agora neste apêndice focalizou números positivos. Nesta seção, explicamos como os computadores representam números negativos utilizando-se *notação de complemento de dois*. Inicialmente mostramos como o complemento de dois de um número binário é formado e, então, mostramos por que ele representa o valor negativo do número binário dado.

Considere uma máquina com inteiros de 32 bits. Suponha

```
int value = 13;
```

A representação de 32 bits de `value` é

```
00000000 00000000 00000000 00001101
```

Para formarmos o negativo de `value`, primeiro formamos seu complemento de um aplicando operador de complemento de bitwise do Java (~) :

```
onesComplementOfValue = ~value;
```

Internamente, `~value` é agora `value` com cada um de seus bits invertidos — 1s tornam-se 0s e 0s tornam-se 1s como se segue:

```
value:
00000000 00000000 00000000 00001101
~value (isto é, o complemento de um do valor):
11111111 11111111 11111111 11110010
```

Para formarmos complemento de dois de `value`, adicionamos 1 ao complemento de um de `value`. Assim

```
Complemento de dois de value:
11111111 11111111 11111111 11110011
```

Agora se isso é de fato igual a –13, devemos ser capazes de adicioná-lo a binário 13 e obter um resultado do 0. Vejamos:

```
 00000000 00000000 00000000 00001101
+11111111 11111111 11111111 11110011
------------------------------------
 00000000 00000000 00000000 00000000
```

O bit de transporte que vem da coluna mais à esquerda é descartado e, de fato, obtemos 0 como resultado. Se adicionássemos o complemento de um número ao número, o resultado seria todos os dígitos como 1. A chave para obter um resultado de todos os dígitos como zero é que o complemento de dois seja 1 maior que o complemento de um. A adição de 1 faz com que cada coluna adicione 0 a um transportador de 1. O transportador continua se movendo na esquerda até que seja descartado do bit mais à esquerda e daí o número resultante é todos os dígitos como zero.

Os computadores na realidade realizam uma subtração, como

```
x = a - value;
```

adicionando o complemento de dois de `value` a `a` como a seguir:

```
x = a + (~value + 1);
```

Suponha que `a` seja 27 e `value` seja 13 como antes. Se o complemento de dois de `value` for realmente o negativo de `value`, adicionar o complemento de dois de `value` a `a` deve produzir o resultado 14. Vejamos:

```
a (i.e., 27)      00000000 00000000 00000000 00011011
+(~value + 1)    +11111111 11111111 11111111 11110011
                 ------------------------------------
                  00000000 00000000 00000000 00001110
```

que é de fato igual a 14.

Resumo

- Um inteiro como 19 ou 227 ou –63 em um programa Java é considerado como estando no sistema de numeração decimal (base 10). Os dígitos no sistema de numeração decimal são 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. O dígito mais baixo é 0 e o mais alto é 9 — um a menos que a base 10.
- Internamente, computadores utilizam o sistema de numeração binário (base 2). O sistema de numeração binário tem apenas dois dígitos, 0 e 1. Seu dígito mais baixo é 0 e o mais alto é 1 — um a menos que a base 2.
- O sistema de numeração octal (base 8) e o sistema de numeração hexadecimal (base 16) são populares principalmente porque tornam conveniente a abreviação de números binários.
- Os dígitos do sistema de numeração octal variam de 0 a 7.
- O sistema de numeração hexadecimal apresenta um problema uma vez que ele requer dezesseis dígitos — um dígito mais baixo que 0 e um mais alto com um valor equivalente ao decimal 15 (um a menos que a base 16). Por convenção, utilizamos as letras A a F para representar os dígitos hexadecimais correspondentes aos valores decimais 10 a 15.
- Cada sistema de numeração utiliza notação posicional — cada posição em que um dígito é escrito tem um valor posicional diferente.
- Um relacionamento particularmente importante, que tanto o sistema de numeração octal como o sistema de numeração hexadecimal têm com o sistema binário, é que as bases octal e hexadecimal (8 e 16, respectivamente) são potências da base do sistema de numeração binário (base 2).
- Para converter um número octal em um binário, substitua cada dígito octal por seu equivalente binário de três dígitos.
- Para converter um número hexadecimal em um binário, substitua cada dígito hexadecimal por seu equivalente binário de quatro dígitos.
- Uma vez que estejamos acostumados a trabalhar com decimal, é conveniente converter um número binário, octal, ou hexadecimal em decimal para obter um sentido do valor 'real' do número.
- Para converter um número em decimal por meio de outra base, multiplique o equivalente decimal de cada dígito por seu valor posicional e some os produtos.
- Os computadores representam números negativos utilizando-se a notação de complemento de dois.
- Para formar o negativo de um valor em binário, primeiro forme o complemento de um aplicando o operador de complemento de bitwise do Java (~). Isso inverte os bits do valor. Para formar o complemento de dois de um valor, adicione um ao complemento de um do valor.

Terminologia

base
bitwise, operador de
complemento (~)
complemento de dois, notação
conversões
dígito
notação de complemento de um
notação posicional
sistema de numeração binário
sistema de numeração de base 10
sistema de numeração de base 16
sistema de numeração de base 2
sistema de numeração de base 8
sistema de numeração decimal
sistema de numeração hexadecimal
sistema de numeração octal
valor de símbolo
valor negativo

Exercícios de revisão

E.1 As bases dos sistemas de numeração binários, octais, decimais e hexadecimais são ________, ________, ________ e ________ , respectivamente.

E.2 Em geral, as representações decimal, octal e hexadecimal de um número binário dado contêm (mais/menos) dígitos que o número binário.

E.3 (*VerdadeiroFalso*) Uma razão popular para utilizar o sistema de numeração decimal é que ele forma uma notação conveniente para abreviar número binário simplesmente substituindo-se um dígito decimal por grupo de quatro bits binários.

E.4 A representação (octal / hexadecimal / decimal) de um valor binário grande é a mais concisa (das alternativas dadas).

E.5 (*Verdadeiro/Falso*) O dígito mais alto em qualquer base é um maior que a base.

E.6 (*Verdadeiro/Falso*) O dígito mais baixo em qualquer base é um menor que a base.

E.7 O valor posicional do dígito mais à direita de qualquer número em octal, binário, hexadecimal ou decimal é sempre ________.

E.8 O valor posicional do dígito à esquerda do dígito mais à direita de qualquer número em octal binário, hexadecimal ou decimal é sempre igual a ________.

E.9 Preencha os valores ausentes neste gráfico de valores posicionais para as quatro posições mais à direita em cada um dos sistemas de número indicados:

```
decimal      1000    100    10    1
hexadecimal  ...     256    ...   ...
binário      ...     ...    ...   ...
octal        512     ...    8     ...
```

E.10 Converta o binário 110101011000 em octal e hexadecimal.

E.11 Converta o hexadecimal FACE em binário.

E.12 Converta o octal 7316 em binário.

E.13 Converta o octal 4FEC em hexadecimal. (*Dica:* Primeiro converta 4FEC em binário então converta esse número binário em octal.)

E.14 Converta o binário 1101110 em decimal.

E.15 Converta o octal 317 em decimal.

E.16 Converta o hexadecimal EFD4 em decimal.

E.17 Converta o decimal 177 em binário, em octal e em hexadecimal.

E.18 Mostre a representação binária do decimal 417. Então, mostre o complemento de 417 e o complemento de dois de 417.

E.19 Qual o resultado quando um número e seu complemento de dois são somados?

Respostas dos exercícios de revisão

E.1 10, 2, 8, 16.

E.2 Menos.

E.3 Falso. O hexadecimal faz isso.

E.4 Hexadecimal.

E.5 Falso. O dígito mais alto em qualquer base é um menor que a base.

E.6 Falso. O dígito mais baixo em qualquer base é zero.

E.7 1 (a base elevada à potência zero).

E.8 A base do sistema de numeração.

E.9 Preencha os valores ausentes nesse gráfico de valores posicionais para as quatro posições mais à direita em cada um dos sistemas de número indicados:

```
decimal      1000    100    10    1
hexadecimal  4096    256    16    1
binário      8       4      2     1
octal        512     64     8     1
```

E.10 Octal 6530; hexadecimal D58.

E.11 Binário 1111 1010 1100 1110.

E.12 Binário 111 011 001 110.

E.13 Binário 0 100 111 111 101 100; octal 47754.

E.14 Decimal 2+4+8+32+64= decimal 110.

E.15 Decimal 7+1*8+3*64=7+8+192=207.

E.16 Decimal 4+13*16+15*256+14*4096=61396.

E.17 Decimal 177
em binário:

```
256 128 64 32 16 8 4 2 1
128 64 32 16 8 4 2 1
(1*128)+(0*64)+(1*32)+(1*16)+(0*8)+(0*4)+(0*2)+(1*1)
10110001
```

em octal:

```
512 64 8 1
64 8 1
(2*64)+(6*8)+(1*1)
261
```

em hexadecimal:

```
256 16 1
16 1
(11*16)+(1*1)
(B*16)+(1*1)
B1
```

E.18 Binário:

```
512 256 128 64 32 16 8 4 2 1
256 128 64 32 16 8 4 2 1
(1*256)+(1*128)+(0*64)+(1*32)+(0*16)+(0*8)+(0*4)+(0*2)+(1*1)
110100001
```

Complemento de um: `001011110`
Complemento de dois: `001011111`
Verificação: Número binário original + seu complemento de dois

```
110100001
001011111
---------
000000000
```

E.19 Zero.

Exercícios

E.20 Algumas pessoas argumentam que muitos de nossos cálculos seriam mais fáceis no sistema de numeração de base 12 porque 12 é divisível por muito mais números que 10 (para a base 10). Qual o dígito mais baixo na base 12? Qual seria o símbolo mais alto para o dígito na base 12? Quais os valores posicionais das quatro posições mais à direita de qualquer número no sistema de numeração de base 12?

E.21 Como é o valor mais alto de símbolo nos sistemas de numeração que discutimos em relação ao valor posicional do primeiro dígito à esquerda do dígito mais à direita de qualquer número nesses sistemas de numeração?

E.22 Complete o seguinte gráfico de valores posicionais para as quatro posições mais à direita em cada um dos sistemas de número indicados:

```
decimal   1000     100     10      1
base 6     ...     ...      6    ...
base 13    ...     169    ...    ...
base 3      27    ...     ...    ...
```

E.23 Converta o binário 100101111010 em octal e hexadecimal.

E.24 Converta o hexadecimal 3A7D em binário.

E.25 Converta o hexadecimal 765F em octal. (*Dica:* Primeiro converta 765F em binário, então converta esse número binário em octal.)

E.26 Converta o binário 1011110 em decimal.

E.27 Converta o octal 426 em decimal.

E.28 Converta o hexadecimal FFFF em decimal.

E.29 Converta o decimal 299 em binário, em octal e em hexadecimal.

E.30 Mostre a representação binária do decimal 417. Então, mostre o complemento de um de 779 e o complemento de dois de 779.

E.31 Qual o resultado quando o complemento de dois de um número é adicionado a ele mesmo?

E.32 Mostre o complemento de dois do valor inteiro -1 em uma máquina com inteiros de 32 bits.