# Conversão de bases numéricas

## Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Converter bases numéricas (binário, octal, decimal e hexadecimal).
- Reconhecer a utilidade das bases numéricas em tecnologia da informação.
- Resolver exercícios de conversão de base numérica.

## Introdução

Um sistema numeral é um sistema de escrita para expressar números; isto é, uma notação matemática para representar números de um determinado conjunto, usando dígitos ou outros símbolos de uma maneira consistente. A mesma sequência de símbolos pode representar números diferentes em diferentes sistemas numéricos. Por exemplo, "11" representa o número três no sistema numérico binário e o número onze no sistema numeral decimal.

Na computação, temos diferentes usos de sistemas numéricos para representar os valores. Por exemplo, é comum vermos cores sendo expressas com o uso de valores hexadecimais. Já no baixo nível, próximo ao *hardware*, o computador opera com o sistema binário. No entanto, é comum que, na programação de sistemas de informação, por exemplo, utilizemos amplamente o sistema decimal, que é comum no nosso dia a dia.

Assim, neste capítulo, você vai aprender a converter bases numéricas (binário, octal, decimal e hexadecimal), reconhecer a utilidade das bases numéricas em tecnologia da informação e resolver exercícios de conversão de base numérica.

## Bases numéricas e conversões

As bases numéricas representam os símbolos possíveis que temos disponíveis para demonstrar ou representar um número ou contagem. No Quadro 1, você pode ver quais são os símbolos que podemos usar em cada sistema de numeração (THURSTON, 2007).

**Quadro 1.** Símbolos para cada uma das bases numéricas mais utilizadas

| Base numérica | Símbolos |
|---|---|
| Decimal | 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9 |
| Octal | 0, 1, 2, 3, 4, 5, 6 e 7 |
| Binário | 0 e 1 |
| Hexadecimal | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F |

Dependendo da base numérica utilizada, podemos diferenciar os símbolos a partir de uma representação diferente. Veja, a seguir, alguns exemplos de representação no Quadro 2.

**Quadro 2.** Representação de bases numéricas

| Base numérica | Representação |
|---|---|
| Decimal | 1000 |
| Binário | $1111101000_2$ |
| Octal | $1750_8$ |
| Hexadecimal | $3E8_{16}$ |

Note que dificilmente conseguimos identificar a base numérica de um número se ele não informar, do lado direito inferior, qual é a sua base. Por exemplo, o número 111110100 poderia ser interpretado como um decimal, já que números decimais não exigem que a base esteja explícita. No entanto,

vimos, no Quadro 2, que se trata de um número de base binária $1111101000_2$; logo, trata-se do valor 1000, mas com outra representação.

## Conversões: binário

Para converter números decimais para números binários, devemos realizar divisões consecutivas, dividindo o número da base decimal por 2 até que não seja mais divisível ao final (THURSTON, 2007). O número binário é o resultado da última divisão combinado com todos os restos de divisão de baixo para cima. Veja, na Figura 1, como fazer essa conversão.

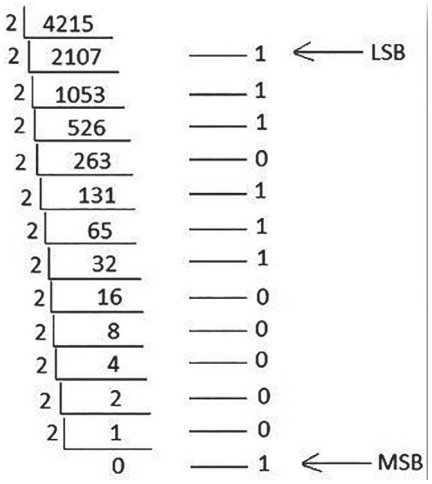



**Figura 1.** Conversão de base decimal para binária.
*Fonte:* Math Only Math (2018, documento on-line).

Podemos verificar, então, que o número decimal $4215_{10}$ torna-se o $1000001110111_2$ em binário, já que fizemos a leitura de baixo para cima dos resultados da divisão. Agora, como fazemos o contrário? Partindo de um número binário, queremos descobrir qual é o seu valor em decimal. Como exemplo, veja a Figura 2, que mostra a conversão do número binário $11011001_2$ para decimal.

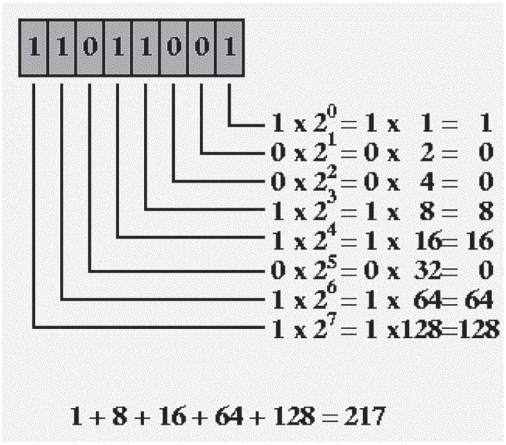



**Figura 2.** Conversão de base binária para decimal.
*Fonte:* Javarevisited (2017, documento on-line).

Podemos verificar que sempre precisamos multiplicar cada símbolo do número binário por 2 elevado à sua posição da esquerda para a direita, iniciando em zero sempre. Logo, o resultado final é $217_{10}$.

## Conversões: hexadecimal

Hexadecimal é um sistema numérico de base 16, e decimal é um número de base 10. Para saber o equivalente decimal de cada dígito de número hexadecimal (THURSTON, 2007), siga os seguintes passos:

1. Obtenha o equivalente decimal em uma tabela hexadecimal para decimal.
2. Multiplique cada dígito por 16 elevado no número de posição (localização) do dígito (por exemplo, iniciando em zero, 7DE: a localização E é 0, a localização D é 1 e a localização 7 é 2).
3. Some todos os multiplicadores.

Observe, na Figura 3, a demonstração dessa conversão com o número hexadecimal $A59C_{16}$, que representa o número decimal $42{,}396_{10}$.

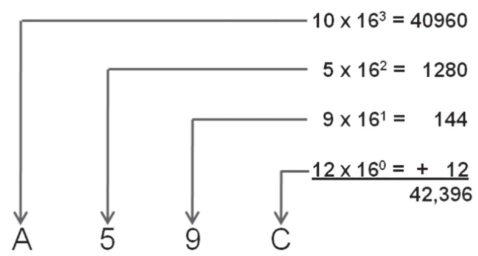



**Figura 3.** Conversão de base hexadecimal para decimal.
*Fonte:* Math Only Math (2018, documento on-line).

## Conversões: octal

Um número decimal regular é a soma dos dígitos multiplicados por $10^n$ (THURSTON, 2007).

**Exemplo 1**

137 na base 10 é igual a cada dígito multiplicado pelo seu correspondente $10^n$:

$137^{10} = 1 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 = 100 + 30 + 7$

Os números octais são lidos da mesma maneira, mas cada dígito conta $8^n$ em vez de $10^n$.

Multiplique cada dígito do número hexadecimal pelo correspondente $8^n$.

**Exemplo 2**

37 na base 8 é igual a cada dígito multiplicado pelo seu correspondente $8^n$:

$378 = 3 \times 8^1 + 7 \times 8^0 = 24 + 7 = 31$

**Exemplo 3**

7014 na base 8 é igual a cada dígito multiplicado pela sua potência correspondente de 8:

$70148 = 7 \times 8^3 + 0 \times 8^2 + 1 \times 8^1 + 4 \times 8^0 = 3584 + 0 + 8 + 4 = 3596$

Veja o exemplo que ilustra essa conversão na Figura 4.

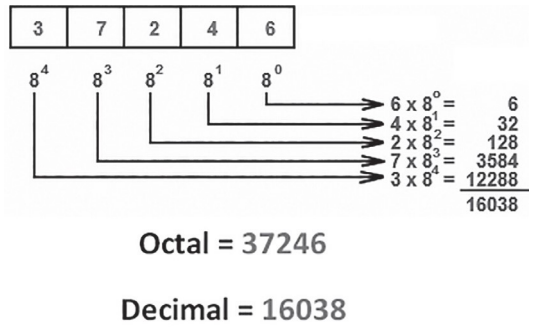



**Figura 4.** Conversão de base octal para decimal.
*Fonte:* Math Only Math (2018, documento on-line).

## A importância das bases numéricas na Tecnologia da Informação

Todo computador é composto por muitos componentes eletrônicos. É por isso que é necessário um conhecimento básico de eletrônica para entender como e por que números binários são usados em computadores.

Um computador é construído com muitas conexões e componentes, que são usados para transferir e armazenar dados, além de se comunicar com outros componentes. A maior parte desse armazenamento, transferência e comunicação acontece com a eletrônica digital. A eletrônica digital usa o sistema binário (ON/OFF) — um sinal com uma série de pulsos ON/OFF é igual a um número binário.

Na eletrônica, um nível de tensão ou fluxo de corrente é uma maneira de representar um valor. Por exemplo, 5 V (volts) ou 0,5 A (amperes). Os fabricantes de dispositivos eletrônicos poderiam, é claro, atribuir qualquer significado que desejassem a diferentes valores de tensão. Se você precisa de 10 valores, pode dividir o intervalo de 0 V – 4 V (incluindo 0, que cria 5 "etapas") por 10 — você acabaria com 0,5 V por etapa. Logo, podemos verificar que podemos ter 2 estados, um ligado e outro desligado (1/0).

Numerais hexadecimais são amplamente utilizados por programadores e programadores de sistemas de computador, pois fornecem uma representação mais humana dos valores codificados em binários. Cada dígito hexadecimal representa quatro dígitos binários, também conhecidos como *nibble*, que são meio *byte*. Por exemplo, um único *byte* pode ter valores variando de 0000 0000 a 1111 1111 na forma binária e pode ser representado de forma mais conveniente como 00 a FF em hexadecimal.

O octal tornou-se amplamente utilizado na computação quando sistemas como os *mainframes* PDP-8, ICL 1900 e IBM empregavam palavras de 12 *bits*, 24 *bits* ou 36 *bits*. Octal foi uma abreviação ideal de binário para essas máquinas, porque o tamanho da palavra é divisível por três (cada dígito octal representa três dígitos binários). Então, quatro, oito ou doze dígitos podem exibir de forma concisa uma palavra de máquina inteira. Ele também reduz custos ao permitir que tubos Nixie, monitores de sete segmentos e calculadoras sejam usados para os consoles do operador, já que os monitores binários eram muito complexos para serem usados, monitores decimais precisavam de *hardware* complexo para converter e monitores hexadecimais, necessários para exibir mais numerais (FERDJALLAH, 2011).

Todas as plataformas de computação modernas, no entanto, usam palavras de 16, 32 ou 64 *bits* divididas em *bytes* de oito *bits*. Nesses sistemas, seriam necessários três dígitos octal por *byte*, com o dígito octal mais significativo representando dois dígitos binários (mais um *bit* do próximo *byte* significativo, se houver). A representação octal de uma palavra de 16 *bits* requer 6 dígitos, mas o dígito octal mais significativo representa (bastante deselegantemente) apenas um *bit* (0 ou 1). Essa representação não oferece nenhuma maneira de ler facilmente o *byte* mais significativo, porque ele é espalhado por quatro dígitos octal. Portanto, hexadecimal é mais comumente usado em linguagens de programação hoje, já que dois dígitos hexadecimais especificam exatamente um *byte*. Algumas plataformas com um tamanho de palavra de duas palavras ainda têm *subwords* de instrução que são mais facilmente entendidas se exibidas em octal; isso inclui a família PDP-11 e Motorola 68000. A moderna arquitetura x86 ubíqua também pertence a essa categoria, mas o octal raramente é usado nessa plataforma, embora certas propriedades da codificação binária de opcodes se tornem mais prontamente aparentes quando exibidas em octal — por exemplo, o *byte* ModRM, que é dividido em campos de 2, 3 e 3 *bits* —, então o octal pode ser útil na descrição dessas codificações (FERDJALLAH, 2011).

Às vezes, o octal é usado na computação em vez de hexadecimal — talvez na maioria das vezes nos tempos modernos — em conjunto com as permissões de arquivo em sistemas Unix (veja chmod). Tem a vantagem de não exigir

símbolos extras como dígitos (o sistema hexadecimal é base-16 e, portanto, precisa de seis símbolos adicionais além de 0 a 9) e é usado para displays digitais.

Nas linguagens de programação, os literais octal são tipicamente identificados com uma variedade de prefixos, incluindo o dígito 0, as letras o ou q, a combinação de dígitos e letras 0o, ou os símbolos & ou $. Na convenção da Motorola, os números octal são prefixados com @, enquanto uma pequena letra o é adicionada como um *postfix* após a convenção da Intel. No DR-DOS e no Multiuser DOS, várias variáveis de ambiente, como $ CLS, $ ON, $ OFF, $ HEADER ou $ FOOTER, suportam uma notação de número \ nnn octal, e o DEBUG DR-DOS utiliza \ to como prefixo de números octal também.

Por exemplo, o literal 73 (base 8) pode ser representado por 073, o73, q73, 0o73, \ 73, 73 e 73, 73 ou 73o em varias linguagens de programação.

Linguagens de programação mais novas têm abandonado o prefixo 0, já que números decimais são frequentemente representados com zeros à esquerda. O prefixo q foi introduzido para evitar que o prefixo o seja confundido com um zero, enquanto o prefixo 0o foi introduzido para evitar iniciar um literal numérico com um caractere alfabético (como o ou q), uma vez que isso poderia confundir o literal com um nome variável. O prefixo 0o também segue o modelo definido pelo prefixo 0x usado para literais hexadecimais na linguagem C; é suportado por Haskell, OCaml, Perl 6, Python a partir da versão 3.0, Ruby, Tcl a partir da versão 9, e destina-se a ser suportado pelo ECMAScript 6 (o prefixo 0 foi desencorajado no ECMAScript 3 e descartado no ECMAScript 5) (FERDJALLAH, 2011).

Números octais que são usados em algumas linguagens de programação (C, Perl, PostScript…) para representações textuais e gráficas de cadeias de *bytes* quando alguns valores de *byte* (não representados em uma página de códigos, não gráficos, tendo significado especial no contexto atual ou não desejados) tem que ser para escapar como \ nnn. A representação octal pode ser particularmente útil com *bytes* não ASCII de UTF-8, que codifica grupos de 6 *bits*, e onde qualquer *byte* inicial tem valor octal \ 3nn e qualquer *byte* de continuação tem valor octal \ 2nn.

## Convertendo bases numéricas

Embora você já tenha visto como fazer a conversão de um número binário, octal ou hexadecimal para decimal, é importante praticar e verificar como realizar as conversões no caminho inverso.

## Conversões: decimal para octal

Lembre-se que "decimal" é chamado de base 10 porque cada dígito representa uma potência de 10. Nós chamamos os primeiros três dígitos de 1s, 10s, 100s, mas também podemos escrever isso como o $10^0$ lugar, o $10^1$ lugar e o $10^2$ lugar. O octal ou sistema de numeração de base 8 usa potências de 8 em vez de potências de 10. Escreva algumas dessas potências de 8 em uma linha horizontal, da maior para a menor. Note que esses números são todos escritos em decimal (base 10) (THURSTON, 2007).

Para converter um decimal para octal, você deve apenas dividir por 8 o número e pegar o resto da divisão de baixo para cima. Veja, na Figura 5, uma ilustração de como fazer essa conversão.

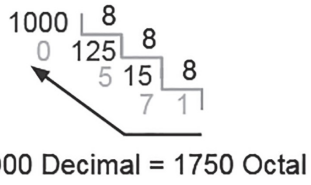


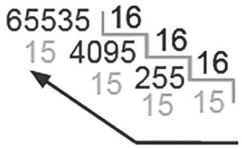

**Figura 5.** Conversão de base decimal para octal.
*Fonte:* Math Only Math (2018, documento on-line).

## Conversões: decimal para hexadecimal

A conversão de um número decimal para hexadecimal é muito simples. Você vai precisar seguir as seguintes etapas para realizar essa conversão:

1. Divida o número por 16.
2. Obtenha o quociente inteiro para a próxima iteração.
3. Obtenha o restante para o dígito hexadecimal.
4. Repita as etapas até que o quociente seja igual a 0.

O processo é muito similar ao da conversão realizada do decimal para o octal, mas, aqui, fazemos a divisão pelo 16. Veja, na Figura 6, um exemplo de conversão do número 65535 para a base hexadecimal, resultando no número $FFFF_{16}$.




65535 Decimal = FFFF Hexadecimal

**Figura 6.** Conversão base decimal para hexadecimal.
***Fonte:*** Math Only Math (2018, documento on-line).

## Exemplo

No método de expansão, a conversão de números binários em seus equivalentes decimais é mostrada com a ajuda dos exemplos. Observe o exemplo de como converter o número decimal 256 em seu equivalente binário.

| 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Como o dado número 256 aparece na primeira linha, colocamos 1 no *slot* abaixo de 256 e preenchemos todos os outros *slots* à direita desse *slot* com zeros.

Assim, $256_{10} = 100000000_2$

## Referências

FERDJALLAH, M. *Introduction to digital systems:* Modeling, Synthesis, and Simulation Using VHDL. Hoboken: Wiley-Blackwell, 2011.

JAVAREVISITED. *How to convert binary number to decimal in java – algorithm*. 28 jan. 2017. Disponível em: <https://javarevisited.blogspot.com/2015/01/how-to-convert-binary-number-to-decimal.html>. Acesso em: 31 out. 2018.

MATH ONLY MATH. *Conversion of numbers*. 2018. Disponível em: <https://www.math-only-math.com/conversion-of-numbers.html>. Acesso em: 31 out. 2018.

THURSTON, H. A. *The number system*. New York: Dover, 2007. (Dover Books on Mathematics).

## Leituras recomendadas

ARPACI-DUSSEAU, R. H.; ARPACI-DUSSEAU, A. C. *Operating systems:* Three Easy Pieces. Madison: Arpaci-Dusseau Books, 2014.

GERALDI, L. M. A.; GALASSI, C. R.; FORMICE, C. R. *Elucidando os sistemas operacionais*. Taquaritinga: Dos autores, 2013.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Sistemas operacionais com Java*. 8. ed. Rio de Janeiro: Campus, 2016.

TANENBAUM, A. S.; BOS, H. *Sistemas operacionais modernos*. 4. ed. São Paulo: Pearson, 2016.