



INovação  
E TECNOLOGIA

unidade

1

# Fundamentos COMPUTACIONAIS

---



# Introdução ao Mundo Digital

Prezado(a) estudante

Estamos começando uma unidade desta disciplina. Os textos que a compõem foram organizados com cuidado e atenção, para que você tenha contato com um conteúdo completo e atualizado tanto quanto possível. Leia com dedicação, realize as atividades e tire suas dúvidas com os tutores. Desta forma, você com certeza alcançará os objetivos propostos para essa disciplina.

## OBJETIVO GERAL



Compreender o mecanismo de funcionamento digital de dispositivos computacional.

## OBJETIVOS ESPECÍFICOS



- Reconhecer o processamento de dados e sistemas de computação.
- Identificar a evolução dos computadores (histórico).
- Descrever a diferença entre hardware e software.
- Reconhecer os sistemas de numeração
- Identificar os sistemas numéricos.
- Desenvolver os cálculos para conversão dos tipos de sistemas (decimal, binário, octal, hexadecimal).
- Conceituar sistemas digitais.
- Listar as principais vantagens do sistema digital.
- Identificar a aritmética binária.
- Conceituar as portas lógicas básicas.
- Listar as principais características das portas lógicas.
- Enumerar os tipos de portas lógicas e seus circuitos.



## Parte 1

# Conceitos Básicos de Informática

O conteúdo deste livro é  
disponibilizado por SAGAH.



# Conceitos básicos de informática

## Objetivos de aprendizagem

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

- Reconhecer processamento de dados e sistemas de computação.
- Identificar a evolução dos computadores.
- Descrever a diferença entre hardware e software.

## Introdução

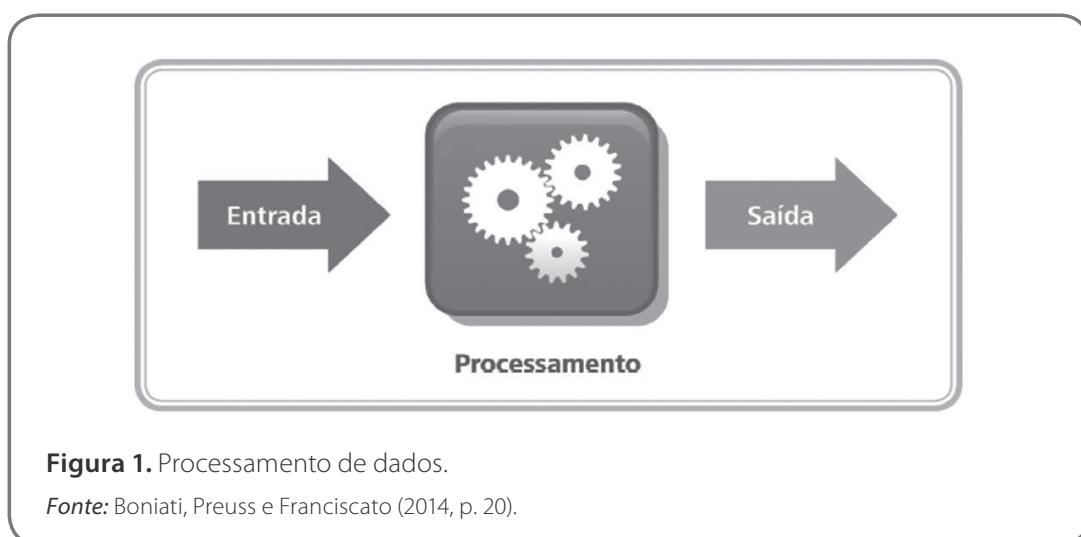
Em um mundo cada vez mais digital, faz-se necessário conhecer os termos básicos relacionados à informática. Não importa a sua profissão ou área de atuação, a tecnologia está em toda parte: no celular, na televisão, no ar-condicionado, no automóvel e até mesmo em sua torradeira. Apesar de o termo informática, historicamente, estar ligado a informação, e não a computação, neste texto esses termos muitas vezes serão usados como sinônimos.

Neste capítulo, você vai estudar os principais conceitos relacionados a sistemas computacionais e ao computador, como ele surgiu, as principais etapas de sua evolução e o seu funcionamento básico. Ainda, você aprenderá a categorizar as principais partes de um computador e as diferenças entre elas.

## Processamento de dados e sistemas de computação

Um computador é uma máquina feita para processar entradas e exibir saídas. Ainda que o conceito pareça simples, ele de início pode trazer alguma confusão. Então, vamos tentar explicá-lo por meio de um exemplo: você trabalha em um escritório especializado em finanças, que recebe diariamente centenas de documentos de empresas que contrataram os seus serviços (notas fiscais, recibos,

relatório de vendas, relatório de investimentos, etc.). O seu computador recebe como **entrada** essas informações, as quais estão desorganizadas e sem uma estrutura padronizada; ele soma, multiplica, agrupa e transforma (**processa**) esses dados aparentemente desorganizados em informações que serão úteis para as empresas, como projeções financeiras e estimativas de custos futuros. Essas informações serão colocadas em um relatório e exibidas para o cliente (**saída do processamento**) (Figura 1).



**Figura 1.** Processamento de dados.

Fonte: Boniati, Preuss e Franciscato (2014, p. 20).

Antes de falarmos sobre sistemas computacionais, vamos entender o que é um computador. O óbvio seria dizer que é uma máquina (e não estaríamos errados), mas o computador é muito mais do que isso. Ele é uma combinação de hardware, software e inteligência humana. **Hardware** é a parte física do computador: a caixa, as placas internas, os circuitos, a impressora, o modem, o roteador sem fio, o monitor, o mouse e outros. O **software** são os programas de computador, como o sistema operacional (Windows, Linux, etc.), planilhas e editores de texto (como o Microsoft Word) e muitos outros. Entre os profissionais da área de informática, existe uma expressão bem-humorada que diz: hardware é tudo aquilo que você chuta, e software é tudo aquilo que você xinga. Seguindo essa lógica, fica mais fácil entender a que cada termo se refere — mas evite de sair por aí chutando o seu computador!

Resumindo, o computador é uma máquina que resolve problemas por meio da execução de instruções que são passadas a ele. Essas instruções são chamadas de **programas de computador**. O programa é um conjunto de instruções lógicas e finitas (também chamado de **algoritmo**), que executam uma tarefa específica.

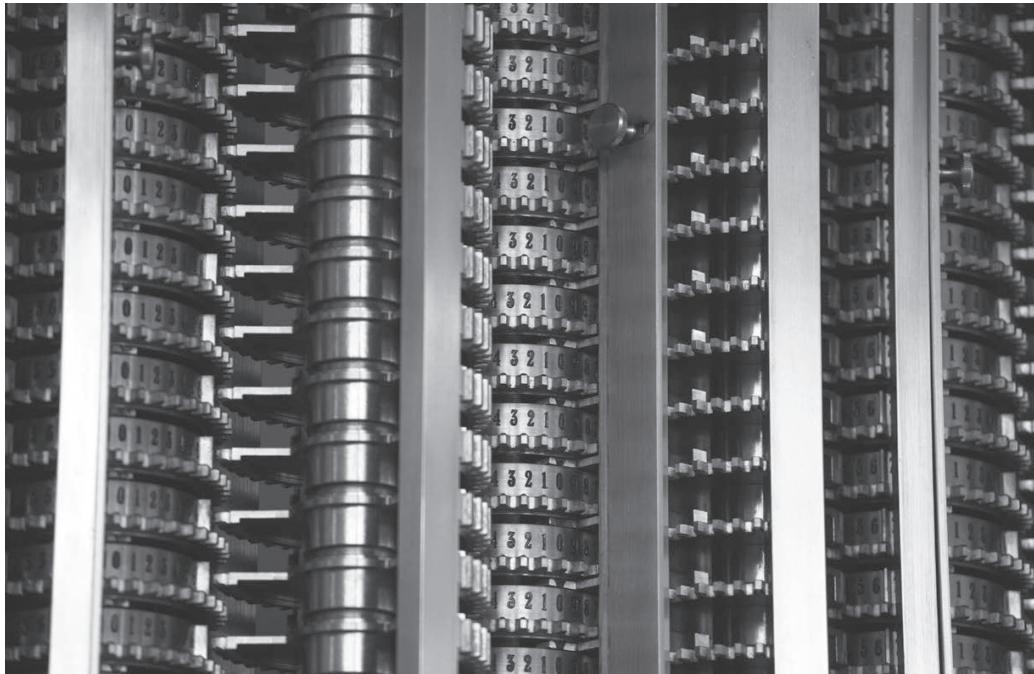
Mas onde está a **inteligência humana**? O hardware do computador é quem faz todo o trabalho “sujo”, mas alguém precisa dizer a ele o que e como fazer. É aí que entra o software. E quem fez o software? Quem escreveu os algoritmos? Existe uma profissão chamada de programador de computadores, e é esse **programador** que é responsável por escrever os programas. Então, o software só pode ser tão inteligente quanto o programador que o fez. Em outras palavras, o computador só faz o que você manda que ele faça.

Um sistema computacional é o agrupamento de tudo isso: componentes de hardware, softwares e pessoas que, em conjunto, são capazes de resolver problemas específicos. Por exemplo, um sistema para caixas eletrônicos possui hardware (o próprio caixa), software (o programa que identifica as suas requisições e as processa) e as pessoas que alimentaram o sistema com as informações necessárias para que ele pudesse funcionar da forma correta, como regras de negócio (uma pessoa não pode retirar dinheiro, se não possuir limite para isso) ou manuais para quem vai utilizar o software ou realizar manutenções no hardware.

Outros exemplos de sistemas computacionais são sistemas para controle de elevadores, sistemas integrados para lojas de varejo, controle acadêmico, automação de bibliotecas e muito mais. Existem sistemas computacionais com os quais o usuário não interage (e às vezes nem sabe que existem), como o sistema que controla o seu ar-condicionado ou partes do seu carro. Você até pode saber que ali existe um sistema computacional, mas nunca interagiu diretamente com ele. Há vários outros com os quais você interage, como o seu editor de textos preferido, o seu caixa automático ou a ferramenta que você usa para navegar na internet.

## Breve histórico da evolução dos computadores

A história dos computadores provavelmente começa com a tentativa do inglês Charles Babbage de construir dois computadores: o chamado dispositivo diferencial e o dispositivo analítico (Figura 2), no começo do século XIX. Embora eles nunca tenham sido construídos, esse fato representou um marco científico importante na época (WEBER, 2012). É claro que, desde Babbage, muita coisa aconteceu, e os computadores evoluíram em ritmo meteórico, transformando o que pensávamos ser ficção científica em realidade trivial. Neste tópico, veremos a evolução dos computadores e o seu contexto. Muitos autores dividem a história do computador em gerações, e essa também será a nossa abordagem.



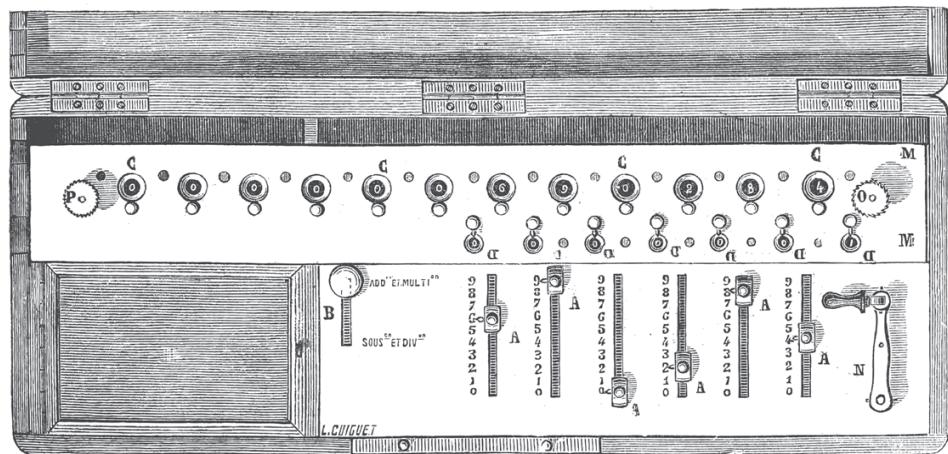
**Figura 2.** Máquina diferencial de Babbage.

Fonte: Purplexsu/Shutterstock.com.

## Geração zero (1642–1945)

As máquinas de Babbage pertencem à primeira das gerações, conhecida como **geração zero**. Além delas, temos ainda a máquina de somar e subtrair, de Blaise Pascal (1623–1662), construída com o intuito de ajudar o seu pai a calcular impostos. É claro que poderíamos falar também dos cartões perfurados, da calculadora de Leibniz (inspirada na calculadora de Pascal), do Arithmomètre de Thomas (calculadora um pouco mais sofisticada, que realizava cálculos mais complexos) e de outros equipamentos similares (Figura 3). Contudo, essa geração é caracterizada por máquinas mecânicas e, já no final dessa época, eletromecânicas.

A Revolução Industrial foi a principal fonte de demanda para que a tecnologia pudesse finalmente decolar. Um fato interessante é que, nessa época, temos a primeira programadora de computadores: Ada, a Condessa de Lovelace. Ela atuou com Babbage e sugeriu a ele o que é considerado hoje como o primeiro programa de computadores: um plano para que a sua máquina diferencial realizasse cálculos.



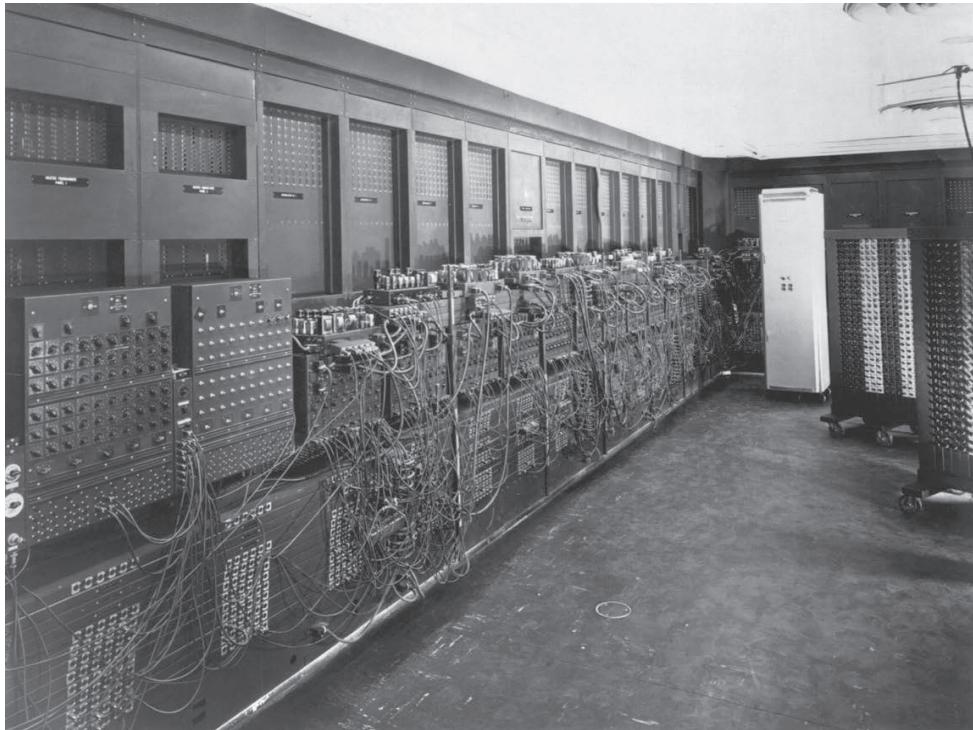
**Figura 3.** Arithmomètre de Thomas.

Fonte: Morphart Creation/Shutterstock.com.

## 1ª Geração (1945–1953)

Infelizmente, as guerras sempre são precursoras de avanços tecnológicos, e foi nesse contexto que novas tecnologia surgiram. Os componentes mecânicos ou eletromecânicos foram substituídos por válvulas, as quais foram inicialmente desenvolvidas para a indústria de rádio. Elas eram muito mais rápidas, mas não muito confiáveis (NULL; LOBUR, 2011). O problema com as válvulas estava em sua função de controlar o fluxo da corrente, amplificando a tensão de entrada, o que provocava a sua queima como um evento bastante frequente. Além disso, elas ocupavam muito espaço, o seu processamento era lento e o consumo de energia era gigantesco.

Os primeiros computadores que utilizaram essa tecnologia foram o ENIAC (Figura 4), feito na Universidade da Pennsylvania (EUA); o IBM 603, o 701 e o SSEC, da Universidade de Cambridge; e o UNIVAC I. O ENIAC levou três anos para ser construído, funcionava com 19.000 válvulas, consumia 200 quilowatts de energia, pesava 30 toneladas, media 5,5 metros de altura e 25 metros de comprimento, e ocupava uma sala de 150 m<sup>2</sup>. Para você ter uma ideia dos problemas causados pelo uso de válvulas, havia, naquela época, uma pessoa cuja única função era trocar as válvulas queimadas, visto que isso acontecia a todo momento, trazendo falta de confiabilidade a todo o sistema (NULL; LOBUR, 2011).



**Figura 4.** ENIAC.

*Fonte:* Everett Historical/Shutterstock.com.

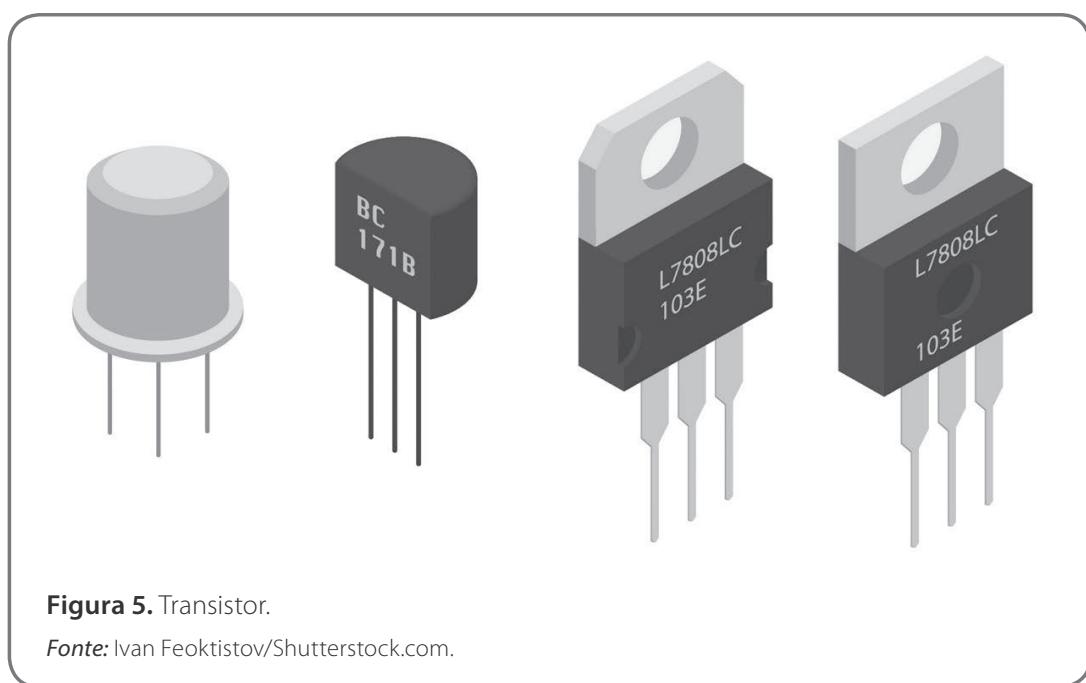
Além disso, a programação no ENIAC era extremamente cansativa e complexa: era feita por meio de 6.000 chaves manuais, e toda entrada de dados era realizada com cartões perfurados. Eram necessárias três equipes para toda a operação: uma para programar os cartões, outra para trocar os cartões à medida que eram lidos pela máquina, e uma terceira equipe, que traduzia os cartões de saída para o padrão decimal.

Essa também foi a época de John von Neumann (1903–1957), matemático brilhante, nascido em Budapeste (Hungria), que contribuiu durante a sua vida em diversas áreas de conhecimento: economia, teoria dos jogos, mecânica quântica e, é claro, computação. O seu modelo de computador foi e é o alicerce dos computadores modernos. Von Neumann participou também da construção do ENIAC (NULL; LOBUR, 2011).

## 2ª Geração (1954–1965)

O fato de as válvulas consumirem enormes quantidades de energia e serem pouco eficientes e confiáveis levou a comunidade científica e as indústrias a pesquisarem novas tecnologias. Além disso, as pesquisas em todos os campos

de conhecimento, desde o setor militar até a área de saúde, começaram a se tornar mais complexas. Esse cenário favoreceu o aparecimento do transistor (Figura 5). No computador, o transistor atua como um interruptor eletrônico.



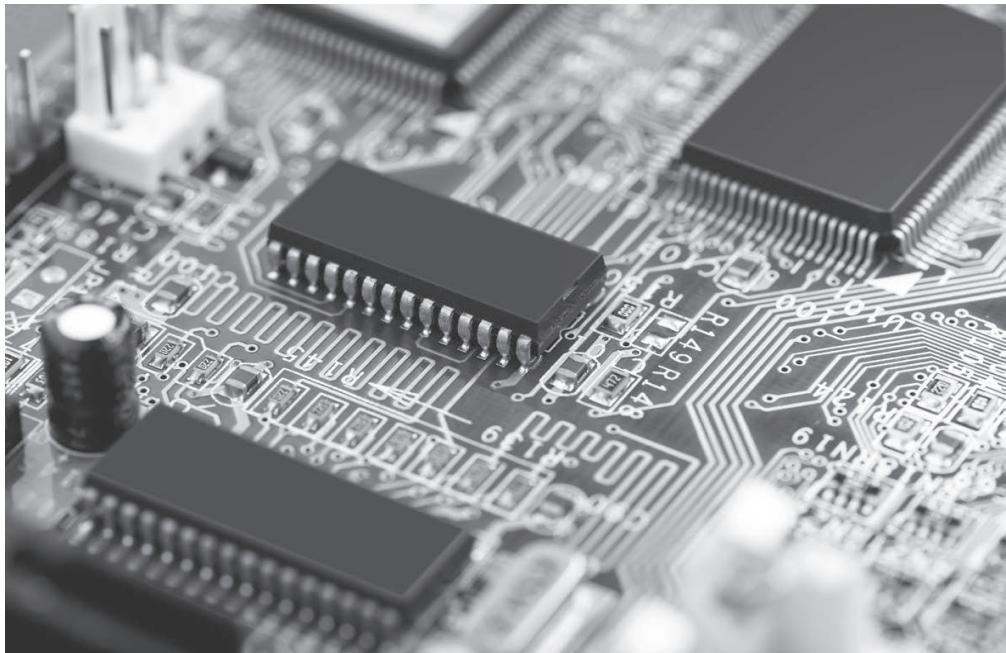
**Figura 5.** Transistor.

Fonte: Ivan Feoktistov/Shutterstock.com.

Descoberto em 1947 por cientistas da Bell Telephone, o transistor era mais barato, menor e mais confiável, possibilitando a redução de tamanho dos computadores (TANENBAUM, 2007). Em 1960, surgiu então o IBM 1401, um computador menor, mais rápido e mais eficiente. Nesse contexto, surgiram também rádios e televisores menores. Todavia, o transistor ainda não era pequeno o suficiente, uma vez que precisava ser conectado a fios e a outros componentes. Foi então que se iniciou a terceira geração, com o **circuito integrado** (NULL; LOBUR, 2011).

### 3<sup>a</sup> Geração (1965–1980)

O circuito integrado (Figura 6), chamado carinhosamente de **chip**, é um componente que encapsula diversos transistores dentro dele. Isso trouxe várias vantagens em relação ao modelo anterior: por não possuir partes móveis, ele é mais confiável; contribui para a miniaturização dos componentes; é mais rápido e a um custo de fabricação muito menor. O surgimento dos chips possibilitou que mais pessoas pudessem ter acesso ao computador (NULL; LOBUR, 2011).



**Figura 6.** Circuitos integrados.

Fonte: Karynav/Shutterstock.com.

O IBM 360 é considerado um dos precursores dessa geração. Ele podia realizar 2 milhões de adições e 500 mil multiplicações por segundo, um feito que alguns anos antes só poderia ser considerado como ficção científica. Outras duas características importantes do IBM 360 eram a sua habilidade de emular outros computadores e a multiprogramação. Nesse caso, multiprogramação se refere ao fato de o IBM 360 ser capaz de armazenar em sua memória diferentes programas: enquanto esperava uma tarefa ser realizada, podia fazer outra (TANENBAUM, 2007).

#### 4<sup>a</sup> Geração (1980–?)

Você provavelmente notou a interrogação acima e se perguntou o que ela quer dizer, não é? A questão é que a maioria dos autores concordam que ainda não sabemos quando essa geração termina, nem se já terminou — você verá adiante que temos a 5<sup>a</sup> geração, mas falaremos disso mais tarde.

Essa geração é caracterizada, principalmente, pelo aperfeiçoamento de tecnologias existentes: o que era menor ficou ainda menor, o que era rápido ficou muito mais rápido. Nasce assim a era dos circuitos integrados. Mas o que são circuitos integrados? Trata-se de uma lâmina de silício (material

semicondutor) na qual são gravados diversos componentes, como transistores, capacitores e resistores. A partir dessa geração, ocorre uma corrida para tentar colocar o maior número possível de componentes em um único circuito e deixar esse circuito cada vez menor. A Tabela 1 mostra, de acordo com a percepção comum (esses números podem variar um pouco, dependendo do autor), quantos transistores podem ser colocados em um único circuito e suas respectivas denominações (TANENBAUM, 2007).

**Tabela 1.** Número de transistores por tipo de circuito.

| Abreviação | Denominação                   | Interpretação comum  |
|------------|-------------------------------|----------------------|
| SSI        | Small Scale Integration       | Até 10               |
| MSI        | Medium Scale Integration      | 11–100               |
| LSI        | Large Scale Integration       | 101–9.999            |
| VLSI       | Very Large Scale Integration  | 10.000–100.000       |
| ULSI       | Ultra Large Scale Integration | 100.001–1.000.000    |
| SLSI       | Super Large Scale Integration | 1.000.001–10.000.000 |

*Fonte:* Tanenbaum (2007).

## 5ª Geração (2018?–??)

É provável que a quinta geração seja marcada pela conectividade entre computadores e entre pessoas. Nessa geração, ouvimos termos como big data, internet das coisas, cidades inteligentes, compartilhamento e armazenamento em nuvem. Todos eles têm algo em comum: conectividade e informação. Essa era é marcada por um dilema físico, uma vez que está cada vez mais difícil tornar os componentes do computador menores e mais rápidos. Então, a solução viável é colocar mais processadores no computador, de forma que ele possa realizar tarefas em paralelo real. Nesse cenário, podemos colocar 10, 20, 1.000, 10.000 processadores em um computador. Outra forma muito utilizada de ganhar mais processamento é agrupando computadores — em um mesmo local ou não — e fazendo com que eles trabalhem juntos, dividindo assim o custo de processamento.

Para que possamos entender essa nova geração, precisamos compreender melhor alguns dos conceitos citados. O termo **big data** se refere ao tsunami de informações em que vivemos. As informações vêm de todos os lugares: Facebook, Instagram, base de dados corporativas, bases de dados abertas na web, etc. O desafio do conceito de big data é conseguir agrupar todas essas informações, de diferentes fontes e com diferentes formatos, extrair delas informações úteis para a sua empresa e mostrar esses resultados de forma lógica e simples. De posse dessas informações, o empresário pode tomar as melhores decisões possíveis para a sua empresa.

**Cidades inteligentes** são, possivelmente, um dos grandes desafios do século XXI. O termo se refere tanto à conectividade, como ao uso inteligente de informações. Para melhor entender esse conceito, acompanhe um exemplo. Imagine que você está viajando com a sua família rumo ao litoral, para aquela praia que você planejou visitar durante meses. Você está viajando em seu carro, e a velocidade é de 80 km; de repente, você passa por uma placa de trânsito indicando que a velocidade máxima é de 60 km. Nesse momento, a placa “conversa” com o seu carro, que automaticamente reduz a velocidade para 60 km. O carro só poderá mudar a velocidade dele para mais de 60 km quando houver uma placa que sinalize tal condição, não importa o que você faça. Isso é cidade inteligente: todos os dispositivos eletrônicos podem conversar entre si e trocar informações. Seu relógio poderá falar com a geladeira, o computador com ar-condicionado, e assim por diante.

Ninguém tem o poder de prever o futuro, mas a cada geração as mudanças são mais rápidas e mais impressionantes. Entretanto, há algo em que podemos acreditar: o que hoje achamos ser ficção científica, amanhã poderá se tornar uma realidade talvez até mesmo trivial.



## Link

Você pode saber um pouco mais sobre a história dos computadores assistindo ao vídeo “Evolução da Informática: dos primeiros computadores à internet”. Acesse o link a seguir.

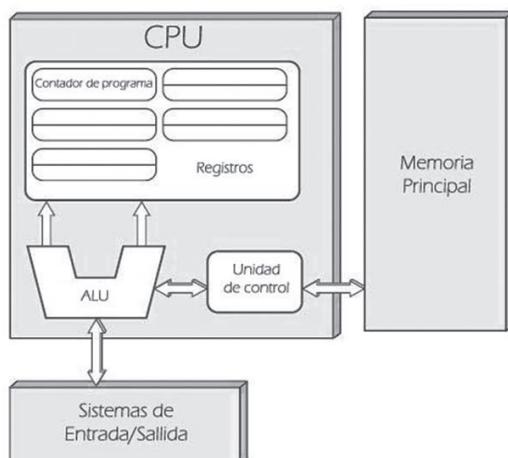
<https://goo.gl/WP8AEb>

## Diferenças entre hardware e software

Nesta seção, você aprenderá um pouco melhor como diferenciar hardware de software, por meio de um olhar um pouco mais aprofundado dos tipos de hardware e de software disponíveis.

### Hardware

Você já viu que o hardware é a parte física do computador, mas vamos examinar isso mais de perto. Entre as grandes contribuições de John von Neumann para a computação, está a ideia de armazenamento de informações. Ele desenvolveu uma nova arquitetura para computadores, baseada em uma unidade de processamento (CPU), um sistema de memória principal e um sistema de entrada e saída (Figura 7).



**Figura 7.** Arquitetura de Von Neumann.

Fonte: Google imagens (com direitos de reutilização).

A arquitetura de Von Neumann define a CPU como unidade de processamentos das instruções, a memória principal (chamada também de memória RAM ou memória volátil) e os dispositivos de entrada (teclado) e saída (impressora). Na CPU, temos ainda registradores, os quais armazenam pequenos volumes de informação. Alguns desses registradores possuem tarefas específicas, como o contador de programa (PC), o qual aponta para a próxima instrução que será decodificada pela CPU.

A CPU é formada por duas partes: a unidade de lógica e aritmética (ULA) e a unidade de controle (UC). A ULA é um dispositivo que realiza operações aritméticas e controla o fluxo de dados, enquanto a UC tem como função acessar, decodificar e executar instruções de um programa que está sendo armazenado em memória.



### Saiba mais

Uma curiosidade interessante é que a maioria das pessoas pensam que o computador executa diversas tarefas em paralelo, ou seja, executa o Word, o Excel e, ao mesmo tempo, navega na internet. Porém, não é bem assim: o computador possui uma peça chamada de processador, o qual é responsável pelo processamento das informações e pela execução dos aplicativos. Se o computador possui apenas um processador, ele só pode processar uma instrução de cada vez; se houver dois processadores, duas instruções de cada vez, e assim por diante. O que realmente acontece é que o processador é muito rápido — um processador de um computador pessoal processa, em média, 100 milhões de instruções por segundo. Assim, você tem a ilusão de que ele realiza várias tarefas ao mesmo tempo. Em comparação com um cérebro humano, porém, que processa cerca de 10 quadrilhões de instruções por segundo, o computador não chega nem perto dessa capacidade (MAIO, 2005).

## Software

Já estabelecemos que softwares são programas de computador, mas vamos conhecer brevemente como os softwares são feitos, por meio de um exemplo. Digamos que você é dono de uma empresa que fabrica programas de computadores, e um cliente, dono de uma empresa de contabilidade, gostaria de contratá-lo para fazer um sistema de controle administrativo e fiscal de condomínios.

O primeiro passo é entender o domínio da aplicação (contabilidade e condomínios), e então relacionar em um documento tudo o que o sistema deve fazer. Após essa etapa, você deverá modelar como as partes do sistema vão interagir entre si, e como o usuário vai interagir com o sistema. A seguir, você começa a escrever o programa, escolhendo uma linguagem de programação. **Linguagem de programação** é uma linguagem próxima à linguagem humana, com a qual você descreverá como o sistema deve se comportar.

Entretanto, o computador não entende essa linguagem, então ela deve ser compilada. O processo de **compilação**, em termos gerais, consiste em transformar uma linguagem em outra — no nosso caso, em linguagem binária (0 e 1), uma vez que essa é a linguagem que o computador comprehende. O computador executa um conjunto de instruções simples, como adição e subtração. Assim, os programas são convertidos nessas instruções antes de serem executados. Esse processo de fabricação de um software está bem resumido, e há diversas etapas não descritas aqui, mas é suficiente para o nosso escopo.

Existem inúmeros tipos de software, para as mais variadas situações. **Softwares de aplicativo, ou simplesmente aplicativos**, são aqueles utilizados por usuários para realizar trabalhos rotineiros. Exemplos de aplicativos são editores de texto, calculadoras, aplicativos para baixar músicas ou filmes, aplicativos para contabilidade e recursos humanos, aplicativos de apoio a decisões gerenciais.

Você com certeza já ouviu muito sobre Windows e Linux, que são exemplos de **sistemas operacionais**. Eles têm a função de gerenciar os recursos do seu computador (memória, periféricos, programas, etc.) e fazer a mediação entre os aplicativos e o hardware do computador.

Além disso, há também **softwares embarcados**, isto é, programas embutidos cuja presença não é percebida pelo usuário. Seu carro provavelmente possui diversos desses sistemas, seu ar-condicionado, sua geladeira, os aviões, os celulares e smartphones também. Enfim, tudo aquilo que possui componentes eletrônicos possivelmente contém sistemas computacionais embarcados.

Outro tipo de software que vem ganhando espaço são os **jogos educativos** e os games de computador. Os softwares educativos vêm crescendo em importância nas salas de aula, possibilitando ao aluno formas lúdicas de aprendizagem, além de contribuir com um processo de aquisição de conhecimentos mais ativo por parte do aluno.



## Referências

- BONIATI, B. B.; PREUSS, E.; FRANCISCATO, R. *Introdução a informática*. 2014. Disponível em: <[http://estudio01.proj.ufsm.br/cadernos/cafw/tecnico\\_agroindustria/introducao\\_informatica.pdf](http://estudio01.proj.ufsm.br/cadernos/cafw/tecnico_agroindustria/introducao_informatica.pdf)>. Acesso em: 2 abr. 2018.
- MAIO, W. de. *O raciocínio lógico matemático*. Fortaleza: Arte & Ciência, 2005.
- NULL, L.; LOBUR, J. *Princípios básicos de arquitetura e organização de computadores*. 2. ed. Porto Alegre: Bookman, 2011.
- TANENBAUM, S. A. *Organização estruturada de computadores*. 5. ed. São Paulo: Pearson, 2007.
- WEBER, F. R. *Fundamentos de arquitetura de computadores*. 4. ed. Porto Alegre: Bookman, 2012.

## Leituras recomendadas

- FONSECA FILHO, C. *História da computação: o caminho do pensamento e da tecnologia*. Porto Alegre: EdiPucrs, 2007. Disponível em: <<http://www.pucrs.br/edipucrs/online/historiadacomputacao.pdf>>. Acesso em: 2 abr. 2018.
- LISBOA JUNIOR, A. de. Evolução da Informática: dos primeiros computadores à internet. *Youtube*, 15 abr. 2012. Disponível em: <[https://www.youtube.com/watch?v=Sx1Z\\_MGwDS8&t=28s](https://www.youtube.com/watch?v=Sx1Z_MGwDS8&t=28s)>. Acesso em: 2 abr. 2018.
- NOBREGA FILHO, R. de G. *A organização de um computador*. [200-?]. Disponível em: <<http://www.di.ufpb.br/raimundo/ArqDI/Arq2.htm>>. Acesso em: 2 abr. 2018.
- PROJETO MAC MULTIMIDIA. *História do computador*. [200-?]. Disponível em: <<https://www.ime.usp.br/~macmulti/historico/>>. Acesso em: 2 abr. 2018.
- TAVARES, T.; COUVRE, M. *Unidade lógica e aritmética*. 2015. Disponível em: <<http://www.dca.fee.unicamp.br/~tavares/courses/2015s2/ea773-3.pdf>>. Acesso em: 2 abr. 2018.

# ANOTAÇÕES



## Parte 2

# Sistema de Numeração

O conteúdo deste livro é  
disponibilizado por SAGAH.



# Sistemas de numeração

## Objetivos de aprendizagem

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

- Conceituar os sistemas de numeração.
- Identificar os sistemas numéricos.
- Desenvolver os cálculos para conversão dos tipos de sistemas (decimal, binário, octal, hexadecimal).

## Introdução

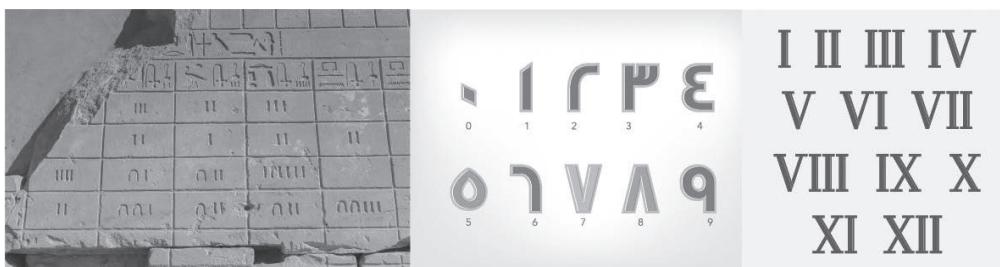
A necessidade de criação de um sistema numérico veio com a necessidade de contar — seja contar gado, plantas, porções de trigo ou qualquer outra coisa. Talvez o fato de termos cinco dedos em cada mão fez com que o nosso sistema numérico viesse a ser naturalmente um sistema baseado em dez números, ou seja, o sistema decimal (0–9). O número é um conceito fundamental em matemática, o qual foi construído numa longa história. Existem evidências arqueológicas de que o homem, já há 50 mil anos, era capaz de contar. Como veremos neste texto, o número também é um conceito importante na computação, e o principal sistema utilizado por essa área é o binário.

Neste capítulo, você vai estudar os conceitos básicos relacionados aos sistemas de numeração, como eles surgiram, os principais tipos de sistemas de numeração e como fazer a conversão entre eles.

## Sistemas de numeração

A necessidade de contar é tão antiga quanto as primeiras civilizações. Os sumérios (localizados onde hoje é o Iraque), os egípcios, os maias, os gregos, os romanos (Figura 1): todos estabeleceram sistemas numéricos com a finalidade de controlar bens, pagamentos, impostos, etc. A palavra **cálculo** (*calculus* em latim) significa pedrinha. Na Antiguidade, pastores associavam as ovelhas dos seus rebanhos a pedras que guardavam em sacolas — cada

ovelha correspondia a uma pedrinha. No início e no final do dia, verificavam se o número de pedrinhas correspondia ao número de ovelhas. Se sobrasse pedra, faltava ovelha.



**Figura 1.** Números na tábua do templo de Karnak (Egito), números arábicos e números romanos.

*Fonte:* Anton\_Ivanov, ihsan kamal, VikiVector/Shutterstock.com

Os egípcios desenvolveram um sistema de numeração aditivo e não posicional (que pode ser escrito da direita para a esquerda ou vice-versa), o qual tinha sete símbolos e era de base 10. Já o sistema babilônico utilizava a base 60. O nosso sistema numérico é baseado no sistema indo-árabe; trata-se de um sistema posicional, no qual existe um símbolo para o valor nulo (zero), e cada algarismo utilizado é uma unidade maior que o seu predecessor. Esse sistema foi adotado na Europa, no século XVI (NULL; LOBUR, 2010). Null e Lobur (2010) explicam a ideia geral por trás desse tipo de sistema:

A ideia geral por trás de sistemas de numeração posicionais é que um valor numérico é representado por potências crescentes de uma raiz (ou base). Isto é frequentemente referido como sistema de numeração ponderado porque cada posição é ponderada por uma potência de uma base.

A criação do zero é considerada um marco na matemática e pode ser comparada à invenção da roda. Ele não foi criado para contagem e foi o último número natural a surgir. Sua necessidade veio da concepção posicional da numeração, solucionando o problema de mecanização do cálculo. Isso possibilitou a criação de máquinas de calcular e foi a base para o desenvolvimento do computador atual.



### Saiba mais

#### Você sabe o que é guematria?

Essa ciência existe apenas no judaísmo e na língua hebraica. A “guematria” é a ciência judaica da codificação bíblica — é um método hermenêutico de análise das palavras bíblicas em hebraico, atribuindo-lhes um valor numérico definido a cada letra. Conhecida ainda pelo nome de “numerologia judaica”, está presente na Toráh (Pentateuco) há mais de 3.300 anos. Pelo valor numérico de cada letra, para os místicos cabalistas judeus, a Toráh tem, para além do seu sentido literal, um sentido místico escondido nos números de cada palavra, como um código, fazendo diferentes conexões e extraindo da palavra divina uma revelação — ou um sentido mais aprofundado, para os espiritualistas (COISAS JUDAICAS, 2015).

## Identificando os sistemas de numeração

### O sistema decimal

O sistema decimal é um sistema posicional de base 10. Os dez algarismos indo-árabicos (0 1 2 3 4 5 6 7 8 9) servem para contar unidades, dezenas, centenas, etc., da direita para a esquerda. Contrariamente a outros tipos de numeração, como a romana ou a antiga egípcia, o algarismo árabe tem um valor diferente, de acordo com a sua posição no número. Assim, em 111, o primeiro algarismo significa 100; o segundo algarismo, 10; e o terceiro, 1.

Logo, o número 1.237, na base 10, pode ser representado por:

$$1.237 = 1 \times 1.000 + 2 \times 100 + 3 \times 10 + 7 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 7 \times 10^0$$

Nesse sistema, o símbolo 0 (zero), quando posicionado à esquerda do número escrito, não altera o seu valor. Dessa forma, 1, 01, 001 ou 0001 representam a mesma coisa. Quando o símbolo zero é colocado à direita, devemos multiplicar a grandeza pela base, que nesse caso é 10.

### Os sistemas binário, octal e hexadecimal

Em computação, o sistema de números mais importante é de base 2 (0 ou 1). Esse sistema foi adotado por causa da natureza do computador: todas as informações armazenadas ou processadas nele usam apenas **duas grandezas**,

representadas pelos algarismos 0 e 1 (desligado ou ligado). A criação desse sistema binário é atribuída a Leibniz (matemático alemão do século XVII). Essa representação binária facilita a representação interna do computador, que é obtida por meio de diferentes níveis de tensão.

Se temos apenas dois números, o elemento mínimo de informação nos computadores foi gentilmente nomeado de **bit** (*binary digit*), ou dígito binário, que pode ser 1 ou 0. Cada conjunto de oito bits é chamado de um **byte** (*binary term*). Às vezes, para facilitar a visualização e manipulação de dados, são utilizadas as bases 8 (octal) e 16 (hexadecimal), mas o computador só opera na base 2 (Figura 2).

O sistema octal, ou sistema de base 8, possui oito algarismos (0, 1, 2, 3, 4, 5, 6 e 7), e é utilizado por ter uma relação direta com o sistema binário. O sistema octal foi muito utilizado na computação como uma alternativa mais compacta do sistema de base 2, na programação em linguagem de máquina.

O hexadecimal é um sistema de numeração muito utilizado na programação de microprocessadores, especialmente em equipamentos de estudo e sistemas de desenvolvimento. Utiliza os símbolos **0,1,2,3,4,5,6,7,8,9** do sistema decimal, e as letras **A, B, C, D, E, F**. As equivalências funcionam da seguinte maneira: A = 10, B = 11, C = 12, D = 13, E = 14 e F = 15 (WEBER, 2012).

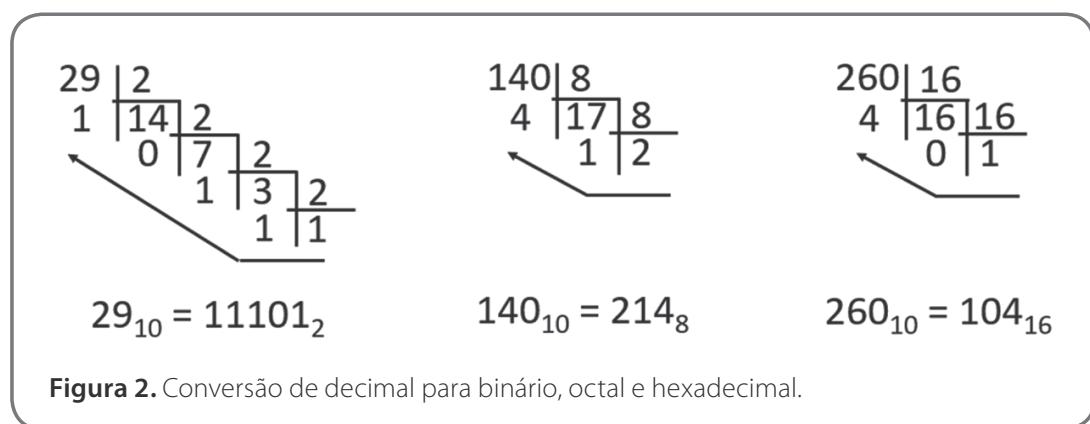
Tabela 1. Números em binário, decimal, octal e hexadecimais.

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

## Conversão entre bases

### Convertendo decimal para binário, octal e hexadecimal

Para converter o número binário ao seu número decimal correspondente, são realizadas divisões sucessivas do número decimal por 2. Em seguida, o resto da divisão de cada operação é agrupado de forma invertida. Na verdade, se você quiser converter um número decimal para qualquer base, o procedimento é o mesmo. Assim, caso você queira saber os números em binário, octal e hexadecimal dos números decimais 29, 140 e 260, respectivamente, basta fazer como na Figura 2 (divisões sucessivas pela base).



### Convertendo binário, octal e hexadecimal para decimal

Para converter de binário, octal ou hexadecimal para o sistema de numeração decimal, utilizamos o somatório da base elevada de zero, até o número de dígitos menos um que queiramos converter. No exemplo da Figura 3, vemos o número binário 1001011<sub>2</sub>, o qual possui sete dígitos; logo, multiplicamos 2 (base) pelo dígito correspondente e somamos os resultados. O mesmo procedimento é feito para números em octal ou hexadecimal.

|   |   |   |   |   |   |       |
|---|---|---|---|---|---|-------|
| 1 | 0 | 0 | 1 | 0 | 1 | $1_2$ |
|---|---|---|---|---|---|-------|

$$(2^6 \times 1) + (2^5 \times 0) + (2^4 \times 0) + (2^3 \times 1) + (2^2 \times 0) + (2^1 \times 1) + (2^0 \times 1) = 75_{10}$$

|   |   |   |       |
|---|---|---|-------|
| 1 | 0 | 5 | $6_8$ |
|---|---|---|-------|

$$(8^3 \times 1) + (8^2 \times 0) + (8^1 \times 5) + (8^0 \times 6) = 558_{10}$$

|   |   |   |          |
|---|---|---|----------|
| 1 | 0 | A | $6_{16}$ |
|---|---|---|----------|

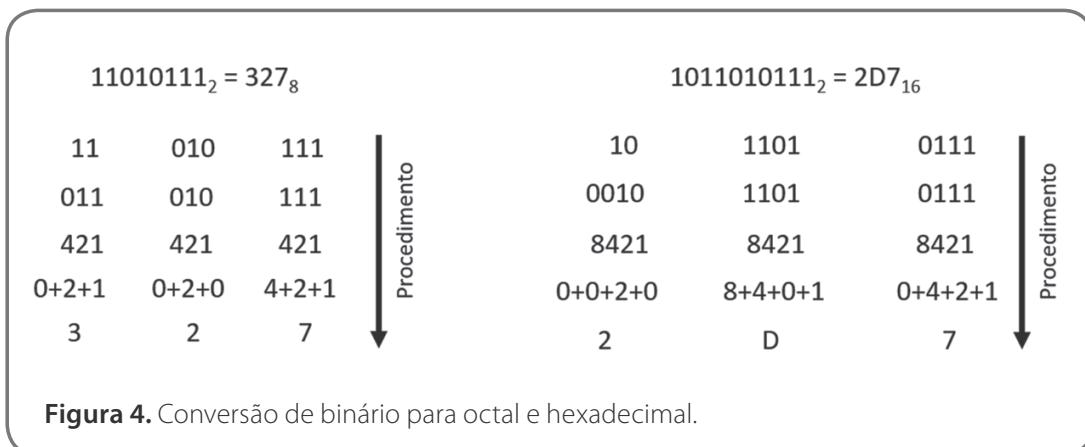
$$(16^3 \times 1) + (16^2 \times 0) + (16^1 \times 10) + (16^0 \times 6) = 4262_{10}$$

**Figura 3.** Conversão de binário, octal e hexadecimal para decimal.

## Convertendo binário para octal e hexadecimal

Para converter de binário para as outras bases que possuem um relacionamento direto com a base 2, como é o caso do sistema numérico de base 8 ( $2^3$ ) e o sistema numérico de base 16 ( $2^4$ ), o procedimento é o mesmo (Figura 4). Digamos que você tenha o número binário  $11010111_2$  e queira convertê-lo para octal e hexadecimal. Os passos para realizar essa conversão são os seguintes:

1. Inicialmente, divide-se o número binário no número de bits correspondentes à base na qual se quer converter (p. ex., base 8 são 3 bits, ou seja,  $2^3 = 8$ ). Esse procedimento é feito da direita para a esquerda.
2. Caso a quantidade dos últimos números binários da esquerda não tiver o número correspondente à base na qual se quer converter, preenche-se com zeros.
3. Depois de divididos os números em grupos de três (no caso da base 8) ou quatro (no caso da base 16), associa-se a cada número binário o seu valor em decimal. No exemplo da Figura 5, no número 011 010 111, o 011 corresponde a  $2^2 + 2^1 + 2^0 = 4 + 2 + 1$ .
4. Por último, efetua-se a soma dos elementos e tem-se o valor na base 8 ou 16. Perceba que se um elemento decimal corresponde a um número 0 do binário, ele não é somado.



## Convertendo octal para hexadecimal e vice-versa

Para converter números da base octal para hexadecimal e vice-versa, você precisa fazer um procedimento de duas etapas: converter o número para a base dois (2) e depois converter para a base que deseja. Não existe método direto para realizar essa conversão.



### Referências

COISAS JUDAICAS. *Os números no judaísmo*. 2015. Disponível em: <<https://www.coisasjudaicas.com/2015/07/os-numeros-no-judaismo.html>>. Acesso em: 7 abr. 2018.

NULL, L.; LOBUR, J. *Princípios básicos de arquitetura e organização de computadores*. 2. ed. Porto Alegre: Bookman, 2010.

WEBER, F. R. *Fundamento de arquitetura de computadores*. 4. ed. Porto Alegre: Bookman, 2012.

### Leituras recomendadas

BARROS JUNIOR, D.; BEZERRA, E. A. *Sistemas numéricos*. 2004. Disponível em: <<https://www.inf.pucrs.br/flash/orgarq/aulas/u1.pdf>>. Acesso em: 7 abr. 2018.

CURSO EM VÍDEO. Notação posicional - bases numéricas #01. *Youtube*, 9 jan. 2017. Disponível em: [https://www.youtube.com/watch?v=J5q7s7l2Eul&list=PLHz\\_AreHm4dl-meSpWzJGWOmFnVF5k\\_lYi](https://www.youtube.com/watch?v=J5q7s7l2Eul&list=PLHz_AreHm4dl-meSpWzJGWOmFnVF5k_lYi). Acesso em: 7 abr. 2018.

JUNIOR, A. Sistemas de Numeração e Conversões de Bases. *Youtube*, 26 mar. 2015. Disponível em: <https://www.youtube.com/watch?v=DJYlndxhcKc&t=223s>. Acesso em: 7 abr. 2018.

# ANOTAÇÕES



## Parte 3

### Sistemas Digitais

O conteúdo deste livro é  
disponibilizado por SAGAH.



# Sistemas digitais

## Objetivos de aprendizagem

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

- Conceituar sistemas digitais.
- Listar as principais vantagens do sistema digital.
- Identificar a aritmética binária.

## Introdução

É difícil, atualmente, pensar em um mundo que não seja digital. Os avanços tecnológicos são cada vez mais constantes e num ritmo cada vez mais acelerado. Relógios, aparelhos domésticos, carros, drones, computadores pessoais, celulares: tudo é baseado na tecnologia que chamamos de **digital**. Existem diversos tipos de sistemas digitais, com aplicações e formas diferentes. Entre as vantagens desses sistemas, estão o custo e o tamanho dos aparelhos que podem ser construídos com essa tecnologia.

Neste capítulo, você vai estudar os conceitos básicos relacionados aos sistemas digitais, suas principais vantagens em relação a outros tipos de sistemas e as operações básicas com o sistema binário, o qual é considerado a base da matemática dos sistemas digitais.

## Sistemas digitais

Um sistema qualquer pode ser definido como um conjunto de elementos que são interligados para compor algo que realize uma funcionalidade específica. Por exemplo, um aparelho de televisão é composto de vários componentes, como tela, autofalantes, circuitos internos, saídas para antena, USB, etc. Todos esses componentes são interconectados por cabos e circuitos elétricos.

Vahid (2008) define sinal digital, sistema digital e circuito digital como:

Um **sinal digital** é aquele que pode assumir um de um conjunto finito de valores possíveis, a qualquer instante, sendo também conhecido como sinal discreto. Em comparação, um sinal analógico pode ter um valor de um conjunto infinito de valores possíveis, sendo também conhecido como sinal contínuo. Um sinal é apenas um fenômeno físico que tem um único valor em cada instante de tempo. Um **sistema digital** é aquele que recebe entradas digitais e gera saídas digitais. Um **circuito digital** é uma conexão de componentes digitais que juntos constituem um sistema digital.

Um sistema também tem uma função bem definida, a qual pode ser identificada a partir das funcionalidades de seus componentes. Por exemplo, a função do aparelho de televisão é receber informação e transmiti-la de modo visual e auditivo — algo que nenhum dos componentes do sistema pode realizar por si só (FLOYD, 2007).

Nesse sentido, pode-se identificar dois aspectos fundamentais em qualquer sistema: sua estrutura e seu comportamento. A **estrutura** reflete os componentes e como eles estão interconectados, enquanto o **comportamento** reflete a funcionalidade do sistema.

Um sistema digital é uma combinação de dispositivos projetados para manipular informação lógica ou quantidades físicas que são representadas no formato digital, ou seja, as quantidades só podem assumir valores discretos. Exemplos de sistemas digitais são computadores digitais, calculadoras, televisores, celulares e muitos outros (Figura 1).



**Figura 1.** Exemplos de sistemas digitais.

*Fonte:* Ruslan Ivantsov/Shutterstock.com.; Namig/Shutterstock.com.; Artos/Shutterstock.com.

É praticamente impossível falar sobre sistemas digitais e entendê-los sem mencionar o seu passado — os sistemas analógicos. Um sistema analógico é composto por dispositivos que manipulam quantidades físicas representadas na forma analógica. Em sistemas analógicos, as quantidades físicas podem variar ao longo de uma faixa contínua de valores. Exemplos de sistemas analógicos

são a amplitude do sinal de saída de um alto-falante, equipamentos de gravação/reprodução de fita magnética, reguladores de luminosidade (ou *dimmers*) (FLOYD, 2007). Exemplos de sistemas analógicos são mostrados na Figura 2.



**Figura 2.** Exemplos de sistemas analógicos.

Fonte: Vladeep/Shutterstock.com.; Radomir/Shutterstock.com.

Sistemas digitais modernos abrangem uma vasta gama de graus de complexidade. Os componentes disponíveis para a construção desses sistemas vão desde chaves do tipo liga-desliga, até computadores completos. O número de componentes em um sistema digital pode variar de um até milhares. Obviamente, quanto mais componentes são necessários à implementação de um sistema digital, mais complexo ele é e, consequentemente, mais difícil é de entender o seu funcionamento e de projetá-lo. Daí a importância do uso de níveis de abstração durante o processo de projeto de sistemas digitais (GÜNTZEL; NASCIMENTO, 2001).

O problema é que o mundo é analógico: o som e a luz, por exemplo, são analógicos. Então por que transformá-los em sinais digitais? Esse processo de transformação de analógico para digital consiste em discretizar o sinal analógico, convertendo-o para uma representação digital que possa ser manipulada e armazenada, como um código binário.



### Link

Acesse o link a seguir para saber mais sobre as diferenças entre sinais analógicos e digitais.

<https://goo.gl/b2UW8P>

## As vantagens e desvantagens dos sistemas digitais

### Vantagens

- Em sistemas digitais, existe uma maior imunidade à distorção e à interferência (o sinal digital só tem dois estados). Com circuitos analógicos, até uma pequena perturbação pode tornar o sinal distorcido de forma inaceitável.
- Em relação ao analógico, o sistema digital tem uma maior capacidade de compactação de dados. Como um sinal digital não passa de uma sequência de números, estes podem ser compactados para reduzir drasticamente o tamanho do arquivo.
- Os sistemas digitais são mais fáceis de projetar, em função de os circuitos empregados nos sistemas digitais serem circuitos de chaveamento. Neles, os valores exatos da tensão ou da corrente dos sinais manipulados não são tão importantes, bastando resguardar a faixa de operação (ALTO ou BAIXO) desses sinais.
- O armazenamento da informação é fácil. Circuitos especiais de chaveamento podem reter a informação pelo tempo que for necessário.
- Precisão e exatidão são maiores. Os sistemas digitais podem trabalhar com qualquer quantidade necessária de dígitos de precisão, com a simples adição de mais circuitos de chaveamento. Nos sistemas analógicos, a precisão em geral é limitada a três ou quatro dígitos, porque os valores de tensão e corrente dependem diretamente dos componentes empregados.
- As operações podem ser programadas. É relativamente fácil e conveniente desenvolver sistemas digitais cuja operação possa ser controlada por um conjunto de instruções previamente armazenadas (programa). Os sistemas analógicos também podem ser programados, mas a variedade e a complexidade das operações envolvidas são bastante limitadas (LIMA, 2011).

### Desvantagens

A grande maioria das variáveis (quantidades) físicas são, em sua natureza, analógicas, e geralmente elas são as entradas e saídas que devem ser monitoradas, operadas e controladas por um sistema. Como exemplos, temos a temperatura, a pressão, a posição, a velocidade, o nível de um líquido, a

vazão. Via de regra, expressamos essas variáveis digitalmente — como quando dizemos que a temperatura é de 64°. Na realidade, porém, estamos fazendo uma aproximação digital de uma quantidade analógica.

Para tirarmos proveito das técnicas digitais, quando lidamos com entradas e saídas analógicas, precisamos executar três etapas:

1. Converter o “mundo real” das entradas analógicas para a forma digital.
2. Processar (ou operar) a informação digital.
3. Converter as saídas digitais de volta para o mundo real, em sua forma analógica.

A necessidade das conversões AD/DA da informação pode ser considerada uma desvantagem, porque introduz complexidade e maior custo aos sistemas. Outro fator muito importante é o tempo extra gasto na conversão (LIMA, 2011).

## Aritmética binária

A álgebra booleana, junto com a aritmética binária, são a matemática dos sistemas digitais. Um conhecimento básico dessas ferramentas é indispensável para o estudo e a análise de circuitos lógicos. Os termos **variável**, **complemento** e **literal** são usados em álgebra booleana. Uma **variável** é um símbolo (geralmente uma letra maiúscula em itálico) usado para representar uma grandeza lógica; qualquer variável simples pode ter um valor 1 ou 0. O **complemento** é o inverso de uma variável e é indicado por uma barra sobre a variável (por exemplo, o complemento da variável A é  $\bar{A}$ ). Se  $A = 1$ , então  $\bar{A} = 0$ ; se  $A = 0$ , então  $\bar{A} = 1$ . O complemento de uma variável A é lido como “A negado” ou “A barrado”. Às vezes é usado outro símbolo, em vez de uma barra, para indicar o complemento de uma variável (por exemplo,  $B'$  indica o complemento de B). Desse ponto em diante, diremos que uma **literal** é a variável ou o complemento de uma variável (FLOYD, 2007).

## Soma de números binários

A soma de números binários segue a mesma lógica de soma em qualquer base — incluindo a base 10 que estamos habituados a usar. A Figura 3 mostra a soma de números binários: em A, vemos os dois números binários, os quais correspondem aos números decimais 41 e 44 em B, a soma procede normalmente, ou seja,  $0 + 1$  ou  $1 + 0$  será 1. Em decimal, quando somamos 5 + 8,

por exemplo, o resultado é 3 e vai 1. Em C, podemos verificar que  $1 + 1 = 0$  e vai 1; em D, o processo é o mesmo que em C.

|  |   |
|--|---|
| $1\ 0\ 1\ 0\ 0\ 1_2 = 41_{10}$<br>$1\ 0\ 1\ 1\ 0\ 0_2 = 44_{10}$                                   | $+ \quad 1\ 0\ 1\ 0\ 1\ 0\ 1_2 = 85_{10}$   |
| <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">A</span>                  | <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">B</span>                     |
| $\begin{array}{r} 1\ 0\ 1\ 0\ 0\ 1 \\ + \ 1\ 0\ 1\ 1\ 0\ 0 \\ \hline 1\ 0\ 1\ 1\ 0\ 0 \end{array}$ | $\begin{array}{r} 1\ 0\ 1\ 0\ 0\ 1 \\ + \ 1\ 0\ 1\ 1\ 0\ 0 \\ \hline 1\ 0\ 1\ 1\ 0\ 1 \end{array}$    |
|  | <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">C</span>                     |
|  | $\begin{array}{r} 1\ 0\ 1\ 0\ 0\ 1 \\ + \ 1\ 0\ 1\ 1\ 0\ 0 \\ \hline 0\ 1\ 0\ 1 \end{array}$          |
|  | <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">D</span>                     |
|  | $\begin{array}{r} 1\ 0\ 1\ 0\ 0\ 1 \\ + \ 1\ 0\ 1\ 1\ 0\ 0 \\ \hline 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{array}$ |

**Figura 3.** Soma de números binários.

## Subtração de números binários

Na subtração, assim como na adição, a lógica não muda. Quando subtraímos 82 de 91, ou seja,  $91 - 82$ , fazemos  $1 - 2$  primeiro; antes, porém, acrescentamos 10 ao 1 (então temos  $11 - 2 = 9$ ) e somamos 1 ao próximo número. Agora,  $9 - (8 + 1) = 0$ ; nesse caso, a resposta é 9. Em binário, como podemos verificar em B (Figura 4),  $0 - 1 = 1$  e vai 1.

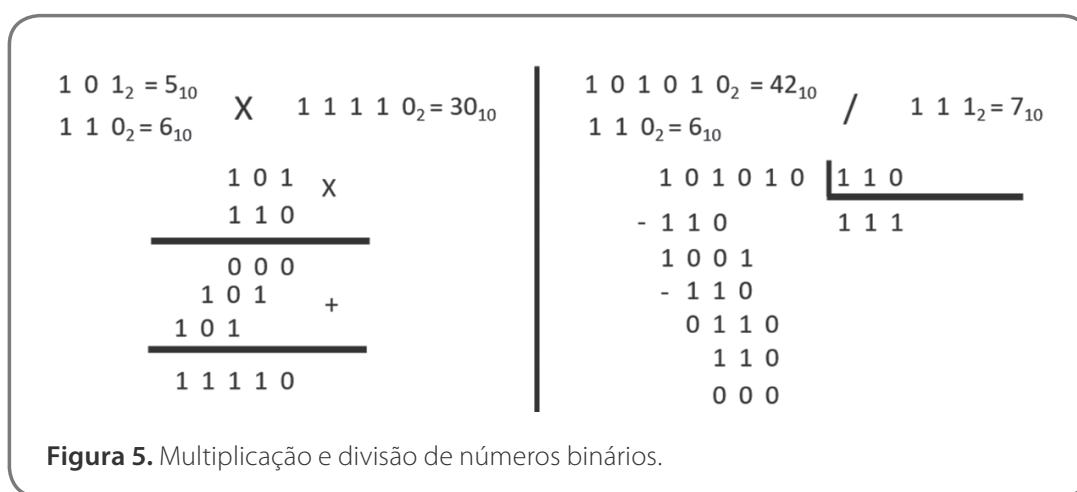
|   |   |
|---|---|
| $1\ 1\ 1\ 1\ 0\ 1_2 = 61_{10}$<br>$1\ 0\ 0\ 0\ 1\ 0_2 = 34_{10}$                                | $- \quad 1\ 1\ 0\ 1\ 1_2 = 27_{10}$   |
| <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">A</span>               | <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">B</span>   |
| $\begin{array}{r} 1\ 1\ 1\ 1\ 0\ 1 \\ - \ 1\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 0 \end{array}$ | $\begin{array}{r} 1\ 1\ 1\ 1\ 0\ 1 \\ - \ 1\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 0 \\ \quad   \\ \hline 0\ 1\ 0 \end{array}$          |
|   | <span style="border: 1px solid black; border-radius: 50%; padding: 2px;">C</span>   |
|   | $\begin{array}{r} 1\ 1\ 1\ 1\ 0\ 1 \\ - \ 1\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 0 \\ \quad   \\ \hline 0\ 1\ 1\ 0\ 1\ 1 \end{array}$ |

**Figura 4.** Subtração de números binários.

## Multiplicação e divisão de números binários

A multiplicação com números binários é realizada da mesma maneira que com números decimais. Ela envolve a formação de produtos parciais e deslocamento

de cada produto parcial sucessivo uma posição à esquerda, seguidos da soma de todos os produtos parciais. O exemplo da Figura 5 ilustra o procedimento; as multiplicações decimais equivalentes são mostradas para referência (FLOYD, 2007). A divisão binária usa o mesmo método de deslocamentos e subtrações utilizado no sistema decimal.

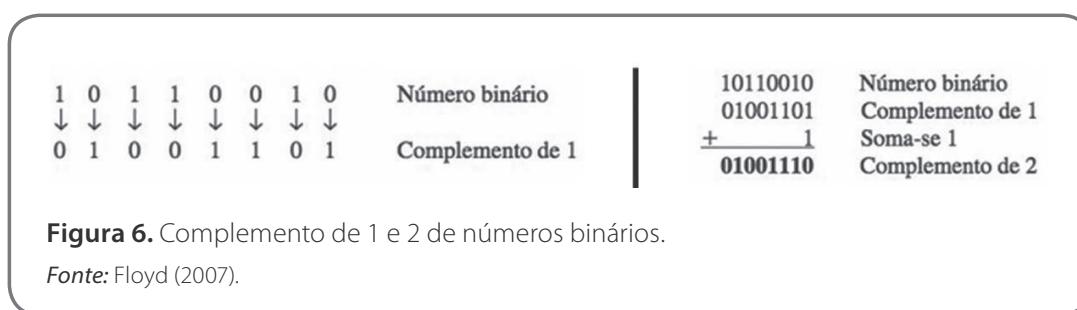


**Figura 5.** Multiplicação e divisão de números binários.

## Complementos de 1 e 2 (sistema binário)

O complemento de 1 e o complemento de 2 de um número binário são importantes, porque eles permitem a representação de números negativos. O método da aritmética do complemento de 2 geralmente é usado em computadores na operação com números negativos (FLOYD, 2007).

O complemento de 1 de um número binário é obtido simplesmente invertendo os bits do número, e o complemento de 2 é esse resultado + 1 (Figura 6).



**Figura 6.** Complemento de 1 e 2 de números binários.

Fonte: Floyd (2007).



## Fique atento

Álgebra binária é muito importante para todos aqueles que querem estudar qualquer conteúdo relacionado a sistemas e circuitos digitais.



## Referências

FLOYD, T. L. *Sistemas digitais: fundamentos e aplicações*. 9. ed. Porto Alegre: Bookman,

2007.

GÜNTZEL, J. L.; NASCIMENTO, F. A. *Introdução aos sistemas digitais*. Florianópolis: UFSC, 2001. Disponível em: <<https://www.inf.ufsc.br/~j.guntzel/isd/isd1.pdf>>. Acesso em: 11 abr. 2018.

LIMA, J. A. G. *Sistemas digitais*. 2012. Disponível em: <<http://www.di.ufpb.br/jose/#disciplinas1>>. Acesso em: 9 abr. 2018.

VAHID, F. *Sistemas digitais: projeto, otimização e HDLs*. Porto Alegre: Bookman, 2008.

## Leituras recomendadas

BARROS JUNIOR, D.; BEZERRA, E. A. *Sistemas numéricos*. 2004. Disponível em: <<https://www.inf.pucrs.br/flash/orgarq/aulas/u1.pdf>>. Acesso em: 9 abr. 2018.

FREITAS JUNIOR, L. C. Números Binários: adição, subtração, complemento de 1 e de 2. *Youtube*, 3 mar. 2015. Disponível em: <<https://www.youtube.com/watch?v=7igvEoqSby8>>. Acesso em: 9 abr. 2018.

JUNIOR, A. Sistemas de Numeração e Conversões de Bases. *Youtube*, 26 mar. 2015. Disponível em: <<https://www.youtube.com/watch?v=DJYIndxhcKc&t=223s>>. Acesso em: 9 abr. 2018.

# ANOTAÇÕES



## Parte 4

# Portas Lógicas e Circuitos Digitais

O conteúdo deste livro é  
disponibilizado por SAGAH.



# Portas lógicas e circuitos digitais

## Objetivos de aprendizagem

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

- Conceituar as portas lógicas básicas.
- Listar as principais características das portas lógicas.
- Enumerar os tipos de portas lógicas e seus circuitos.

## Introdução

Portas lógicas são a base construtiva de qualquer sistema digital. Elas são usadas para criar circuitos digitais e até mesmo circuitos integrados complexos. Por exemplo, circuitos integrados complexos podem ser circuitos digitais completos prontos para serem usados — processadores e microcontroladores são os melhores exemplos, mas internamente esses circuitos integrados foram projetados usando várias portas lógicas.

Neste capítulo, você vai estudar os conceitos básicos relacionados às portas lógicas, suas principais características, seus tipos e sua relação com circuitos digitais.

## Portas lógicas

Portas lógicas são a base para compreender os circuitos digitais. Inicialmente, você vai entender as portas lógicas básicas; depois, por meio da álgebra de Boole, vai compreender as principais operações que podem ser realizadas por elas. O termo **porta** é usado para descrever um circuito que realiza uma operação lógica básica.

Os símbolos lógicos usados para representar as portas lógicas estão de acordo com o padrão 91-1984, da ANSI/IEEE. Esse padrão foi adotado pela indústria privada e militar para uso em documentações internas, bem como na literatura publicada (FLOYD, 2007).

Tanto a lógica programável quanto a lógica de funções fixas são discutidas neste capítulo. Em função de os circuitos integrados (CIs) serem usados em todas as aplicações, as funções lógicas de um dispositivo geralmente são mais importantes para o técnico ou tecnólogo, do que os detalhes da operação do circuito em nível de componentes, dentro do encapsulamento do CI. Portanto, a abordagem detalhada de dispositivos em nível de componente pode ser tratada como um tópico opcional (FLOYD, 2007).

## Porta AND

A porta AND é uma porta que implementa o “E” lógico, ou seja, ele só é verdadeiro (1) quando as duas entradas são verdadeiras (1 e 1). Todas as outras opções têm como saída **falso** (0), como você pode ver na Figura 1. Independentemente de quantas entradas a porta tem (duas, três, quatro, n), a saída só será 1 se todas as entradas forem 1.

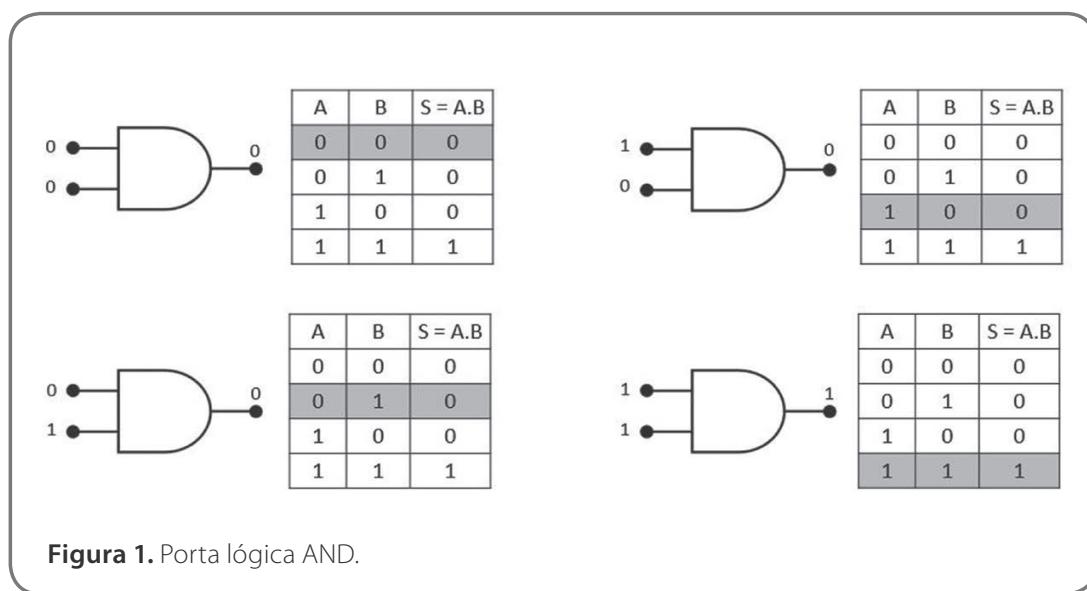
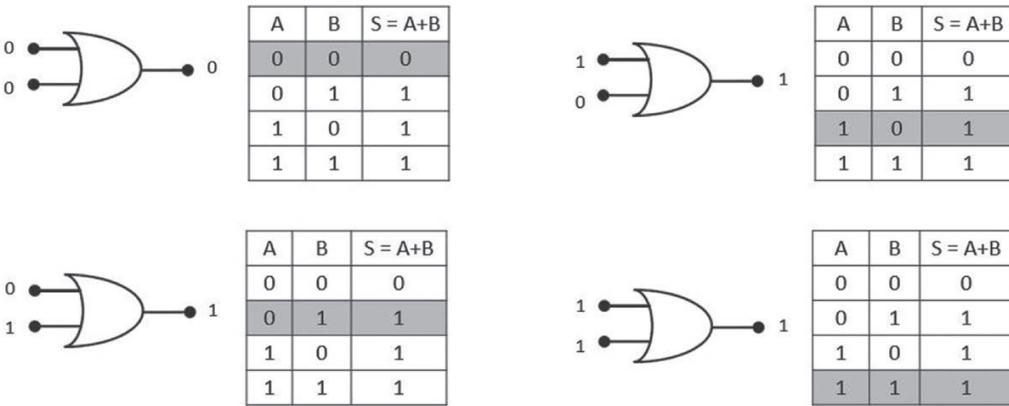


Figura 1. Porta lógica AND.

## Porta OR

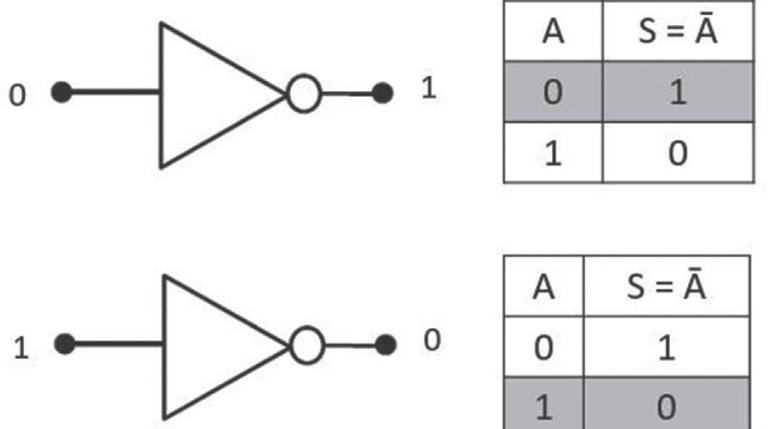
A porta OR implementa o “OU” lógico. Nesse caso, a saída será falsa (0) somente se todas as entradas forem falsas. Se pelo menos uma entrada for verdadeira (1), então a saída será verdadeira (Figura 2).



**Figura 2.** Porta lógica OR.

## Porta NOT

A porta NOT (NÃO) — ou inversor — implementa uma negação lógica: se a entrada é 1, a saída será 0; se a entrada for 0, a saída será 1. Essa função pode ser utilizada em conjunto com outras portas e, assim, serve para inverter o sinal de saída ou de uma entrada específica (daí o nome de inversor). A Figura 3 mostra a tabela e o desenho correspondentes à função NOT.



**Figura 3.** Porta lógica NOT ou inversor.

## Porta NAND e porta NOR

As portas AND e OR podem ser combinadas com o inversor lógico NOT, produzindo as portas lógicas NAND e NOR. O sinal de saída é, como o nome sugere, inverso ao sinal da porta sem o NOT. Essas portas podem ser vistas na Figura 4.

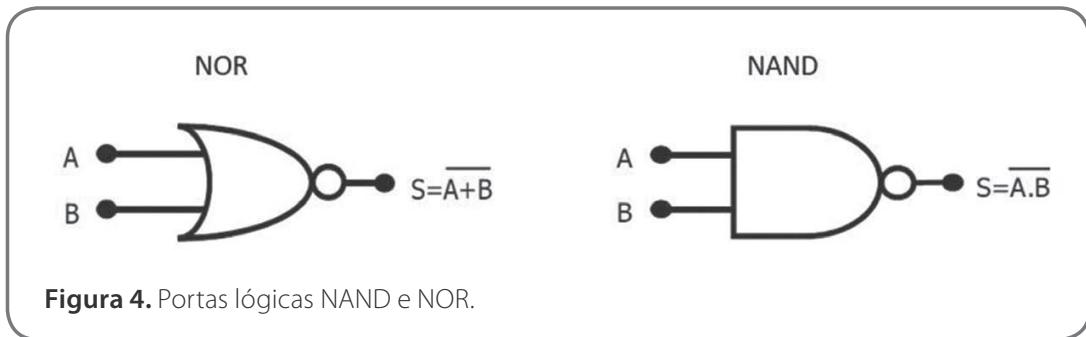


Figura 4. Portas lógicas NAND e NOR.

## Porta XOR

A porta XOR (ou OU EXCLUSIVO) fornece saída 1 quando as entradas forem diferentes entre si, e 0 em caso contrário. Veja a figura correspondente na Figura 5.

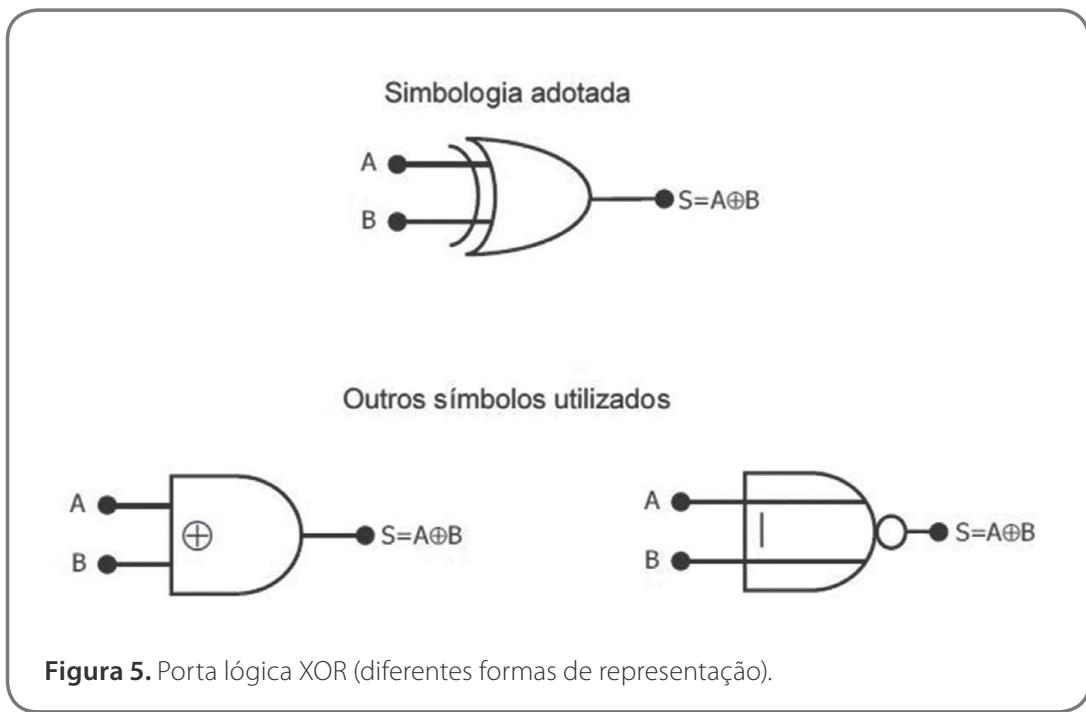
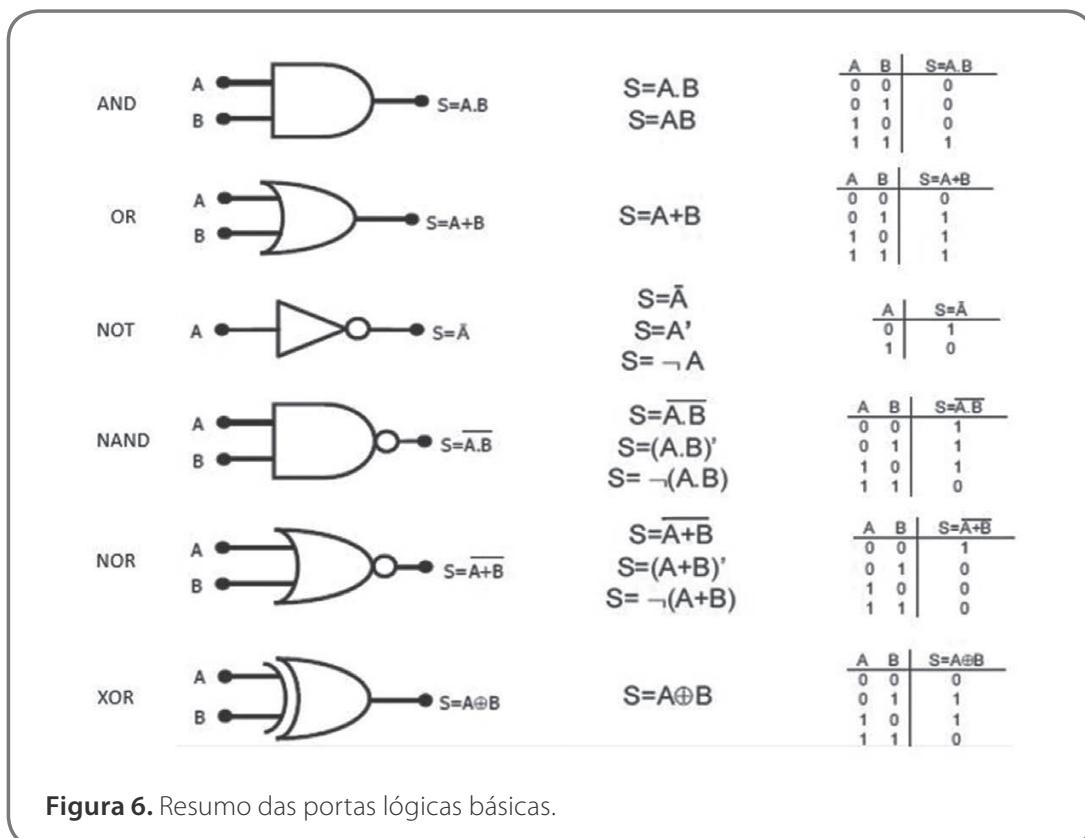


Figura 5. Porta lógica XOR (diferentes formas de representação).

A Figura 6 mostra um resumo das portas lógicas vistas até agora.



**Figura 6.** Resumo das portas lógicas básicas.

## Circuitos integrados

Um circuito integrado, também chamados CI ou **chip**, é um pedaço quadrado de silício, de aproximadamente 5 x 5 mm, contendo um conjunto de portas lógicas e encapsulado em um invólucro retangular de plástico ou cerâmica, de 5 a 15 mm de largura e 20 a 50 mm de comprimento (FELGUEIRAS, [200-?]).

Os CIs podem ser classificados, quanto à quantidade de portas lógicas, da seguinte forma:

- **Circuito SSI (Small Integration Scale):** de 1 a 10 portas lógicas (Figura 7).
- **Circuito MSI (Medium Integration Scale):** de 10 a 100 portas lógicas.
- **Circuito LSI (Large Integration Scale):** de 100 a 100.000 portas lógicas.
- **Circuito VLSI (Very Large Integration Scale):** > 100.000 portas lógicas.

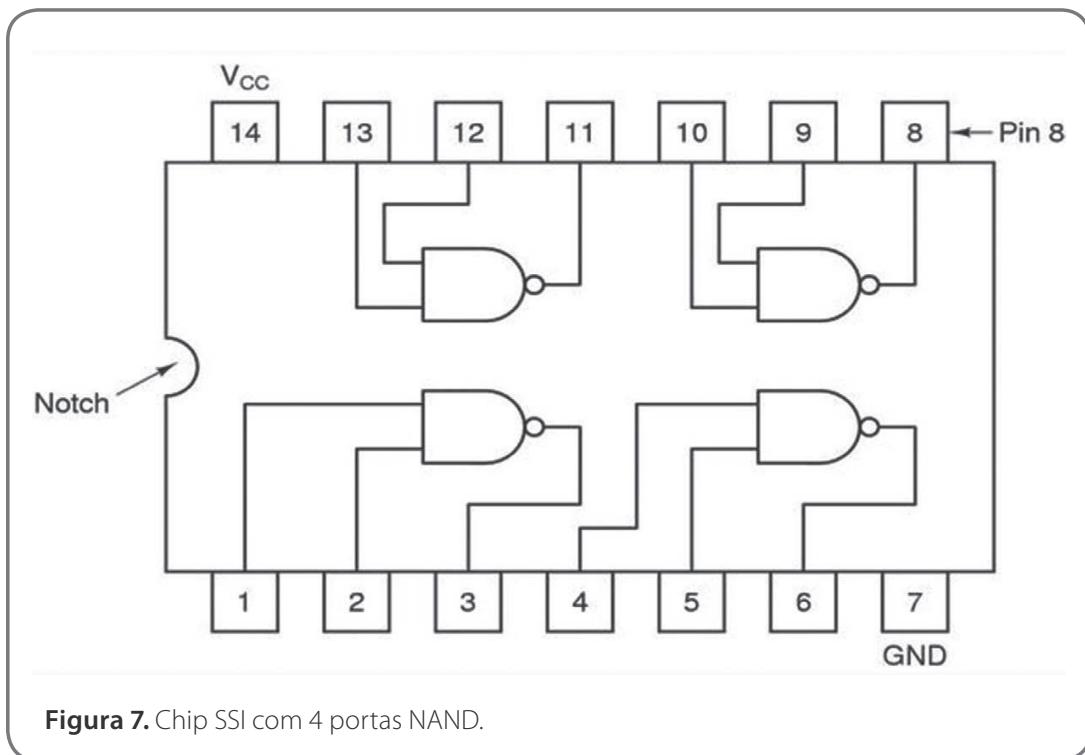


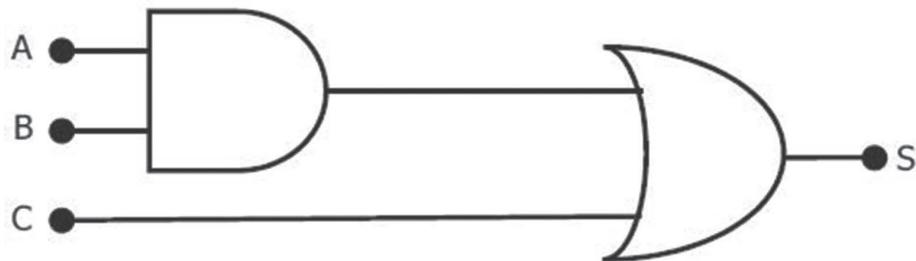
Figura 7. Chip SSI com 4 portas NAND.

Os circuitos lógicos dos sistemas digitais podem ser de dois tipos: circuitos combinacionais ou circuitos sequenciais. Um **circuito combinacional** é constituído por um conjunto de portas lógicas, as quais determinam os valores das saídas diretamente a partir dos valores atuais das entradas. Pode-se dizer que um circuito combinacional realiza uma operação de processamento de informação, a qual pode ser especificada por meio de um conjunto de equações booleanas. Cada combinação de valores de entrada pode ser vista como uma informação diferente, e cada conjunto de valores de saída representa o resultado da operação (GUNTZEL, [200-?]).

Um **circuito sequencial**, por sua vez, emprega elementos de armazenamento denominados *latches* e *flip flops*, além de portas lógicas. Os valores das saídas do circuito dependem dos valores das entradas e dos estados dos *latches* ou *flip flops* utilizados. Como os estados dos *latches* e *flip flops* é função dos valores anteriores das entradas, diz-se que as saídas de um circuito sequencial dependem dos valores das entradas e do histórico do próprio circuito. Logo, o comportamento de um circuito sequencial é especificado pela sequência temporal das entradas e de seus estados internos (GUNTZEL, [200-?]). A seguir, você verá como obter as expressões booleanas geradas por um circuito lógico.

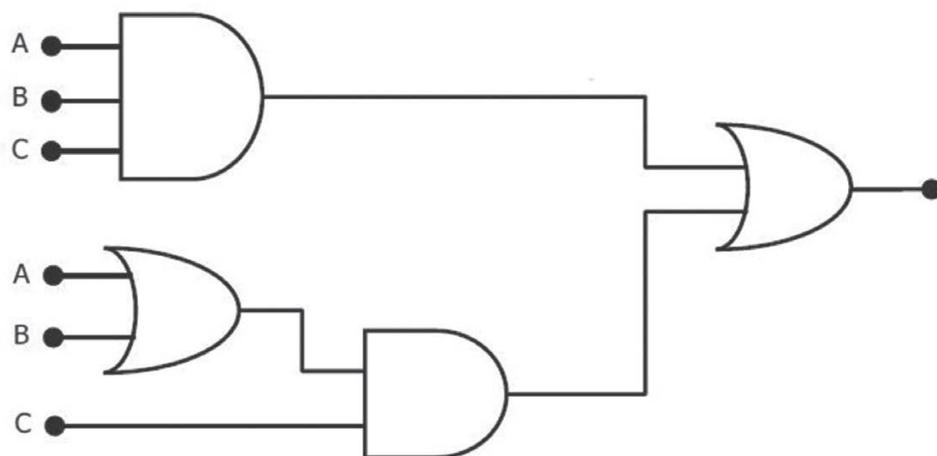
Dado o circuito apresentado na Figura 8, começamos por dividi-lo em portas lógicas básicas — nesse caso, temos uma porta AND (E1) e uma porta

OR (E2). A porta AND pode ser definida como  $E1 = A \cdot B$ . Já na porta OR, uma de suas entradas é a saída de E1; logo, ela pode ser definida como  $E2 = E1 + C$ . Trocando as variáveis, obtemos então a expressão final:  $F = (A \cdot B) + C$ .



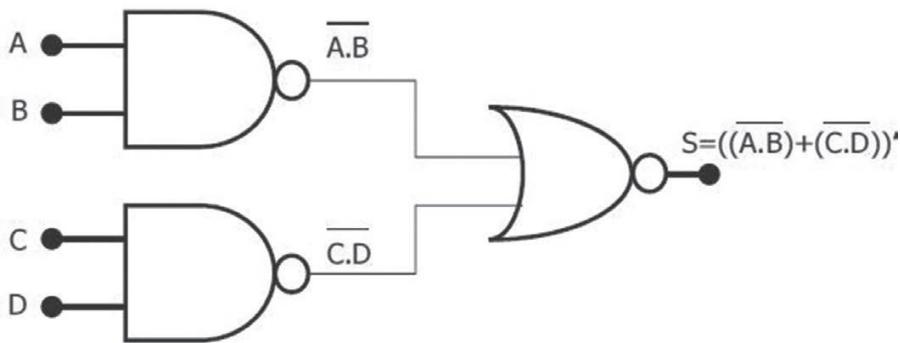
**Figura 8.** Exemplo de circuito  $(A \cdot B) + C$ .

No próximo exemplo, descrito na Figura 9, temos duas portas AND e duas portas OR. A primeira porta AND tem três entradas (A, B e C); logo, temos  $A \cdot B \cdot C$  (E1). A porta OR tem duas entradas (A e B) e pode ser representada por  $A + B$  (E2). A segunda porta AND tem E2 como entrada e a entrada C; logo, a representação é  $(A + B) \cdot C$  (E3). A última porta (OR) tem duas entradas (E1 e E3); portanto, a sua representação é  $E1 + E3$ , ou seja,  $S = (A \cdot B \cdot C) + (A + B) \cdot C$ .



**Figura 9.** Exemplo de circuito  $(A \cdot B \cdot C) + (A + B) \cdot C$ .

O inverso também é bastante útil, isto é, fazer o circuito a partir de dada representação. Por exemplo, digamos que temos a representação  $S = ((\cdot) + (\cdot))'$ . A forma mais fácil é dividir a expressão em blocos: ( $\cdot$ ) é o primeiro bloco; ( $\cdot$ ) o segundo bloco; S seria o bloco final. O primeiro e o segundo blocos são portas AND com o inversor, ou seja, portas NAND. O bloco final S é a porta NOR com duas entradas: primeiro e segundo blocos. A Figura 10 mostra o resultado.



**Figura 10.** Resultado da expressão  $S = ((\cdot) + (\cdot))'$

Uma forma de estudar uma função booleana consiste em utilizar a sua tabela verdade. Como visto anteriormente, há uma equivalência entre o circuito lógico e a sua expressão característica:

- podemos obter um circuito a partir de sua expressão;
- podemos obter expressões a partir dos circuitos.

Uma tabela verdade representa o comportamento tanto do circuito, como de sua expressão característica. Considere a expressão:  $S = A \cdot B \cdot C + A \cdot D + A \cdot B \cdot D$ ; como são quatro entradas, temos  $2^4 = 16$  possibilidades. Colocamos primeiro as possibilidades para cada variável, depois para cada expressão; no final, o resultado fica como mostra a Tabela 1.

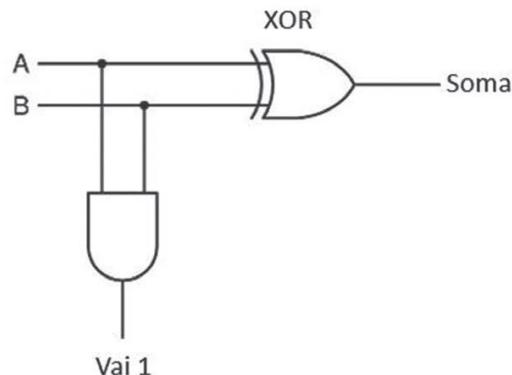
**Tabela 1.** Verdade da expressão  $S = A \cdot B \cdot C + A \cdot D + A \cdot B \cdot D$ .

| A | B | C | D | $A \cdot B \cdot C$ | $A \cdot D$ | $A \cdot B \cdot D$ | S        |
|---|---|---|---|---------------------|-------------|---------------------|----------|
| 0 | 0 | 0 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 0 | 0 | 1 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 0 | 1 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 0 | 1 | 1 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 1 | 0 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 1 | 0 | 1 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 1 | 1 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 0 | 1 | 1 | 1 | 0                   | 0           | 0                   | <b>0</b> |
| 1 | 0 | 0 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 1 | 0 | 0 | 1 | 0                   | 1           | 0                   | <b>1</b> |
| 1 | 0 | 1 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 1 | 0 | 1 | 1 | 0                   | 1           | 0                   | <b>1</b> |
| 1 | 1 | 0 | 0 | 0                   | 0           | 0                   | <b>0</b> |
| 1 | 1 | 0 | 1 | 0                   | 1           | 1                   | <b>1</b> |
| 1 | 1 | 1 | 0 | 1                   | 0           | 0                   | <b>1</b> |
| 1 | 1 | 1 | 1 | 1                   | 1           | 1                   | <b>1</b> |

Os blocos mais elementares da eletrônica são as portas lógicas, como vimos até agora. Vamos agora aplicar esse conhecimento para construir alguns blocos menos elementares — por exemplo, vamos fazer um circuito que implemente a soma de dois binários A e B. O problema dessa soma é quando A é 1 e B é 1: sua soma será 0 e vai 1. Como fazemos isso? Na Figura 11, temos a tabela verdade e o circuito, que chamamos **meio somador**.

Note que a saída do XOR corresponde à soma dos dois bits, enquanto a saída da porta AND corresponde ao transporte de bits, ou seja, “vai 1”.

| A | B | Soma | Vai 1 |
|---|---|------|-------|
| 0 | 0 | 0    | 0     |
| 0 | 1 | 1    | 0     |
| 1 | 0 | 1    | 0     |
| 1 | 1 | 0    | 1     |



**Figura 11.** Circuito meio somador.



## Link

Acesse o link a seguir para saber mais sobre somadores, codificadores e decodificadores.

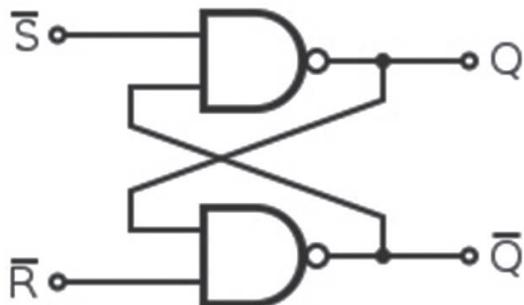


<https://goo.gl/usk8VE>

Em conjunto com o conceito de *flip flops*, precisamos ter em mente também o conceito de lógica sequencial. De maneira simples, porém clara, circuitos sequências são aqueles que têm as saídas dependentes das variáveis de entrada e/ou de seus estados anteriores, que permanecem armazenados e que operam sob o comando de uma sequência de pulsos (*clocks*). Voltando aos *flip flops*, temos em seu circuito suas variáveis de entrada, uma entrada para o *clock* e duas saídas, normalmente denominadas Q e Q' (MINIPA, 2009).

Quando falamos de *clocks* e circuitos sequências, precisamos entender apenas um conceito muito simples: as saídas se alteram de acordo com a entrada apenas quando damos um pulso no *clock*. Como você já deve ter notado, os *flip flops* são circuitos sequências lógicos desenvolvidos para inúmeras aplicações, como o controle de alguma produção industrial: temos várias entradas, que devem funcionar de acordo com determinada lógica, para que a produção possa ser otimizada e nunca parar (MINIPA, 2009).

O *flip flop* mais básico é o RS. Nele temos duas saídas Q e Q', e as suas variáveis de entrada são um *Set* e um *Reset* — o *Set* seleciona o nível lógico 1 na saída do circuito, e o *Reset* seleciona o nível lógico 0 (Figura 12).



**Figura 12.** Circuito *flip flop* RS.



### Link

Acesse o link a seguir para saber mais sobre *flip flops*.

<https://goo.gl/SdaRII>

O assunto é extenso, e aqui trouxemos o conhecimento básico com relação a portas lógicas e circuitos, mas você pode aprender muito mais com as referências trazidas neste texto, como o livro de Floyd (2007), o qual traz todo o material em detalhes e de forma bem didática.



## Referências

FELGUEIRAS, C. A. *Portas lógicas e álgebra de boole*. [200-?]. Disponível em: <[http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula\\_5bn1.html](http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula_5bn1.html)>. Acesso em: 9 abr. 2018.

FLOYD, T. L. *Sistemas digitais: fundamentos e aplicações*. 9. ed. Porto Alegre: Bookman, 2007.

GUNTZEL, L. J. *Circuitos combinacionais*. [200-?]. Disponível em: <<https://www.inf.ufsc.br/~j.guntzel/isd/isd3.pdf>>. Acesso em: 9 abr. 2018.

MINIPA. *Flip flops*. 2009. Disponível em: <<http://escolainustrial.com.br/escolainustrial.com.br/Apostilas/M-1113a-1100-Aluno-Por.pdf>>. Acesso em: 9 abr. 2018.

## Leituras recomendadas

BARANAUSKAS, J. A. *Funções lógicas e portas lógicas*. 2012. Disponível em: <<http://dcm.ffclrp.usp.br/~augusto/teaching/aba/AB-Funcoes-Logicas-Portas-Logicas.pdf>>. Acesso em: 9 abr. 2018.

DEAECTO, G. S. *Circuitos lógicos*. 2013. Disponível em: <[http://www.fem.unicamp.br/~grace/circuitos\\_combinacionais.pdf](http://www.fem.unicamp.br/~grace/circuitos_combinacionais.pdf)>. Acesso em: 9 abr. 2018.

ELETROÔNICA FÁCIL. Eletrônica digital 2 - funções e portas lógicas - porta lógica e - eletrônica fácil. *Youtube*, 4 ago. 2013. Disponível em: <<https://www.youtube.com/watch?v=dnW293BodTo>>. Acesso em: 9 abr. 2018.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



#### CONTRIBUA COM A QUALIDADE DO SEU CURSO

Se você encontrar algum problema nesse material, entre em contato pelo email [eadproducao@unilasalle.edu.br](mailto:eadproducao@unilasalle.edu.br). Descreva o que você encontrou e indique a página.

**Lembre-se:** a boa educação se faz com a contribuição de todos!



Av. Victor Barreto, 2288  
Canoas - RS  
CEP: 92010-000 | 0800 541 8500  
[ead@unilasalle.edu.br](mailto:ead@unilasalle.edu.br)