



## FUNDAMENTOS E ARQUITETURA DE COMPUTADORES

Professor Me. Andre Abdala Noel



Acesse o seu livro também disponível na versão digital.

Quando identificar o ícone QR-CODE, utilize o aplicativo **Unicesumar Experience** para ter acesso aos conteúdos online. O download do aplicativo está disponível nas plataformas:



Google Play



App Store

**UNICESUMAR**

Av. Guedner, 1610 - Jardim Aclimação  
Cep 87050-900 - MARINGÁ - PARANÁ  
unicesumar.edu.br  
44 3027.6360

**UNICESUMAR EDUCAÇÃO A DISTÂNCIA**

NEAD - Núcleo de Educação a Distância  
Bloco 4 - MARINGÁ - PARANÁ  
unicesumar.edu.br  
0800 600 6360

as imagens utilizadas neste  
livro foram obtidas a partir  
do site SHUTTERSTOCK.COM

**FICHA CATALOGRÁFICA**

C397 **CENTRO UNIVERSITÁRIO DE MARINGÁ.** Núcleo de Educação a Distância; **NOEL**, Andre Abdala.

**Fundamentos e Arquitetura de Computadores.** Andre Abdala Noel.  
Maringá-Pr.: UniCesumar, 2019. Reimpresso em 2022.  
184 p.  
"Graduação - EaD".

1. Fundamentos. 2. Arquitetura de computadores. 3. Tecnologia.  
4. EaD. I. Título.

ISBN 978-85-459-1676-5

CDD - 22 ed. 005  
CIP - NBR 12899 - AACR/2

Ficha catalográfica elaborada pelo bibliotecário  
João Vivaldo de Souza - CRB-8 - 6828

Impresso por:

**Reitor**

Wilson de Matos Silva

**Vice-Reitor**

Wilson de Matos Silva Filho

**Pró-Reitor Executivo de EAD**

William Victor Kendrick de Matos Silva

**Pró-Reitor de Ensino de EAD**

Janes Fidélis Tomelin

**Presidente da Mantenedora**

Cláudio Ferdinandi

**NEAD - Núcleo de Educação a Distância****Diretoria Executiva**

Chrystiano Mincoff

James Prestes

Tiago Stachon

**Diretoria de Graduação**

Kátia Coelho

**Diretoria de Pós-graduação**

Bruno do Val Jorge

**Diretoria de Permanência**

Leonardo Spaine

**Diretoria de Design Educacional**

Débora Leite

**Head de Curadoria e Inovação**

Tania Cristiane Yoshie Fukushima

**Gerência de Processos Acadêmicos**

Taessa Penha Shiraishi Vieira

**Gerência de Curadoria**

Carolina Abdalla Normann de Freitas

**Gerência de Contratos e Operações**

Jislaine Cristina da Silva

**Gerência de Produção de Conteúdo**

Diogo Ribeiro Garcia

**Gerência de Projetos Especiais**

Daniel Fuverki Hey

**Supervisora de Projetos Especiais**

Yasminn Talyta Tavares Zagonel

**Coordenador de Conteúdo**

Danillo Xavier Saes

**Designer Educacional**

Hellyery Agda Gonçalves da Silva

**Projeto Gráfico**

Jaime de Marchi Junior

José Jhonne Coelho

**Arte Capa**

Arthur Cantareli Silva

**Ilustração Capa**

Bruno Pardinho

**Editoração**

Victor Augusto Thomazini

**Qualidade Textual**

Jaqueline Mayumi Ikeda Loureiro

**Ilustração**

Bruno Pardinho



Professor  
Wilson de Matos Silva  
Reitor

Em um mundo global e dinâmico, nós trabalhamos com princípios éticos e profissionalismo, não só para oferecer uma educação de qualidade, mas, acima de tudo, para gerar uma conversão integral das pessoas ao conhecimento. Baseamo-nos em 4 pilares: intelectual, profissional, emocional e espiritual.

Iniciamos a Unicesumar em 1990, com dois cursos de graduação e 180 alunos. Hoje, temos mais de 100 mil estudantes espalhados em todo o Brasil: nos quatro campi presenciais (Maringá, Curitiba, Ponta Grossa e Londrina) e em mais de 300 polos EAD no país, com dezenas de cursos de graduação e pós-graduação. Produzimos e revisamos 500 livros e distribuímos mais de 500 mil exemplares por ano. Somos reconhecidos pelo MEC como uma instituição de excelência, com IGC 4 em 7 anos consecutivos. Estamos entre os 10 maiores grupos educacionais do Brasil.

A rapidez do mundo moderno exige dos educadores soluções inteligentes para as necessidades de todos. Para continuar relevante, a instituição de educação precisa ter pelo menos três virtudes: inovação, coragem e compromisso com a qualidade. Por isso, desenvolvemos, para os cursos de Engenharia, metodologias ativas, as quais visam reunir o melhor do ensino presencial e a distância.

Tudo isso para honrarmos a nossa missão que é promover a educação de qualidade nas diferentes áreas do conhecimento, formando profissionais cidadãos que contribuam para o desenvolvimento de uma sociedade justa e solidária.

Vamos juntos!





## Janes Fidélis Tomelin

Pró-Reitor de Ensino de EaD

## Kátia Solange Coelho

Diretoria de Graduação e Pós

## Débora do Nascimento Leite

Diretoria de Design Educacional

## Leonardo Spaine

Diretoria de Permanência

Seja bem-vindo(a), caro(a) acadêmico(a)! Você está iniciando um processo de transformação, pois quando investimos em nossa formação, seja ela pessoal ou profissional, nos transformamos e, consequentemente, transformamos também a sociedade na qual estamos inseridos. De que forma o fazemos? Criando oportunidades e/ou estabelecendo mudanças capazes de alcançar um nível de desenvolvimento compatível com os desafios que surgem no mundo contemporâneo.

O Centro Universitário Cesumar mediante o Núcleo de Educação a Distância, o(a) acompanhará durante todo este processo, pois conforme Freire (1996): “Os homens se educam juntos, na transformação do mundo”.

Os materiais produzidos oferecem linguagem dialógica e encontram-se integrados à proposta pedagógica, contribuindo no processo educacional, complementando sua formação profissional, desenvolvendo competências e habilidades, e aplicando conceitos teóricos em situação de realidade, de maneira a inseri-lo no mercado de trabalho. Ou seja, estes materiais têm como principal objetivo “provocar uma aproximação entre você e o conteúdo”, desta forma possibilita o desenvolvimento da autonomia em busca dos conhecimentos necessários para a sua formação pessoal e profissional.

Portanto, nossa distância nesse processo de crescimento e construção do conhecimento deve ser apenas geográfica. Utilize os diversos recursos pedagógicos que o Centro Universitário Cesumar lhe possibilita. Ou seja, acesse regularmente o Studeo, que é o seu Ambiente Virtual de Aprendizagem, interaja nos fóruns e enquetes, assista às aulas ao vivo e participe das discussões. Além disso, lembre-se que existe uma equipe de professores e tutores que se encontra disponível para sanar suas dúvidas e auxiliá-lo(a) em seu processo de aprendizagem, possibilitando-lhe trilhar com tranquilidade e segurança sua trajetória acadêmica.

**Professor Me. Andre Abdala Noel**

Professor e programador, mestre em Ciência da Computação pela Universidade Estadual de Maringá, com ênfase em sistemas de computação, e bacharel em Ciência da Computação pela Universidade Estadual de Maringá. Possui boa experiência em programação, aplicando também na docência superior, desde 2008. Autor do site Vida de Programador, se mantém bem ativo na comunidade de desenvolvedores.

Acesse: <<http://lattes.cnpq.br/9035823171388697>>.

# FUNDAMENTOS E ARQUITETURA DE COMPUTADORES

## SEJA BEM-VINDO(A)!

Olá, nobre aprendiz da bela arte da computação e tecnologia, escrevo este livro, tendo em mente que vai ser lido por aprendizes recém-chegados à nossa área, com um conhecimento básico que qualquer pessoa tem sobre computadores, ou por aprendizes que já estão há mais tempo na área, há talvez 5, 10, 20 ou 30 anos. Essa é uma das belezas de nossa área: somos todos aprendizes! Sempre!

Estou eu aqui com vocês, como um aprendiz. O quanto não aprendo a cada dia? Muitos de meus alunos me ajudam e me incentivam a aprender cada vez mais, me ensinam conhecimentos específicos que eles possuem, me forçam a estudar um pouco mais de áreas que não domino. Então, estamos todos em constante aprendizado. Uma pessoa que trabalha em T.I., que não está em permanente estado de aprendiz, torna-se ultrapassada em algum tempo.

Escrever esse livro foi um misto de prazer e desafio. Foi muito bom poder juntar em unidades o conhecimento que já venho trabalhando com os alunos há um tempo, mas também um desafio de buscar transmitir informações acuradas. Não é como uma aula, em que na próxima semana posso me desculpar e corrigir um lapso. Também, foi o primeiro livro que escrevi que tem mais textos do que imagens! (Tenho dois livros de tirinhas publicados.)

Um outro desafio foi saber a hora de parar. A nossa área é, ao mesmo tempo, muito grande e muito apaixonante. Como esse livro é para uma disciplina introdutória, eu não posso me aprofundar em todos os assuntos estudados aqui, nem mesmo seria justo fazer isso com vocês. Para quem está chegando seria um peso muito grande e desnecessário neste momento.

A minha intenção com este livro foi a de cobrir todos os assuntos essenciais, tentando deixar para vocês um direcionamento de como seguir adiante, tentando dar a base que você precisa saber para depois caminhar os próximos passos por conta própria quando precisar se aprofundar em algum ponto.

Neste livro, iremos tratar alguns assuntos diferentes. Primeiro, lançaremos as bases, os fundamentos da computação. Veremos, inicialmente, alguns conceitos gerais, até para situar o conteúdo que você vai acompanhar. Em seguida, veremos um pouco da história da computação, na qual acabei buscando mais informações, mas quanto mais eu buscava, mais aparecia, então, tem muito mais história se alguém quiser ir mais fundo. Finalizaremos a primeira unidade vendo como são formados os sistemas de computação.

Em seguida, na segunda unidade, daremos início a um conhecimento mais prático. A ideia é apresentar a você como os dados são representados em um sistema computacional, desde a menor unidade, o bit. Veremos como os bits se juntam para formar as informações e quais são as unidades utilizadas. Depois, passaremos a estudar os sistemas de numeração, que são muito importantes na hora de converter os bits para algo que nos seja mais familiar, como um número decimal, um texto ou até mesmo um vídeo na tela. Veremos como realizar as conversões numéricas de uma base para a outra e qual é a importância de saber outras bases.

# APRESENTAÇÃO

Na terceira unidade, continuaremos o nosso estudo mais prático, aplicando o que aprendemos sobre bits e sobre binários, para entender o funcionamento dos circuitos digitais e da lógica booleana. Veremos, então, o que vem a ser a lógica digital, que trabalha apenas com 0 e 1. Veremos como podemos combinar os valores, por meio de operadores, similar ao que fazemos na aritmética. Como essas operações ou expressões se tornam os circuitos digitais propriamente ditos. Ao final da unidade, veremos algumas regras da álgebra booleana, para aprender a simplificar expressões e, com isso, simplificar os circuitos digitais.

Depois disso, na quarta unidade, veremos um pouco de nosso computador por dentro, focando no “cérebro” da máquina, analisando como funciona o processamento e a memória. As tecnologias de processadores e memórias evoluem num ritmo rápido, por isso não nos deteremos em modelos específicos, mas veremos como se dá o funcionamento interno do processador, como ele se divide por dentro, como executa as instruções da máquina e como se comunica com o resto do computador. Também, veremos como funciona a memória, quais são os tipos de memória existentes e como a memória se comporta de acordo com o seu tipo.

Por fim, na última unidade, iremos mais próximos aos usuários, analisando como funciona a parte de entrada e saída de um computador, para entender como o usuário passa dados ao computador e como recebe dados de volta. Veremos como isso chega ao processador e/ou à memória para que a informação seja tratada. Depois, veremos, por alto, o que é um sistema operacional e para que ele serve, vendo também algumas particularidades sobre alguns sistemas operacionais. Então, passaremos aos programas aplicativos, os softwares desenvolvidos para serem utilizados por usuários finais, como podem ser feitos, como são executados e uma pequena discussão sobre as diferentes licenças de software aplicáveis a qualquer programa desenvolvido.

Espero, com este livro, poder colaborar para que você tenha uma formação com mais qualidade, que você possa entender os conceitos de uma forma direta e que o que você ainda vai estudar pela frente possa fazer sentido, devido às parcelas de conhecimento que vão se ligar ao seu cérebro e te acompanhar pela vida. Tenha uma ótima leitura e tenha também muito sucesso no que você fizer com esse conhecimento! Ah, e não deixe de passar aquele café gostoso para lhe fazer companhia durante a leitura!

# SUMÁRIO

## UNIDADE I

### **FUNDAMENTOS DA COMPUTAÇÃO**

- 
- 15 Introdução
  - 16 Conceitos Gerais
  - 18 História da Computação
  - 36 Sistemas de Computação
  - 39 Considerações Finais
  - 49 Referências
  - 50 Gabarito

## UNIDADE II

### **REPRESENTAÇÃO DE DADOS**

- 
- 53 Introdução
  - 54 Unidades de Informação
  - 59 Notação Posicional
  - 64 Conversões Entre Bases Numéricas
  - 72 Considerações Finais
  - 78 Referências
  - 79 Gabarito



# SUMÁRIO

## UNIDADE III

### **LÓGICA DIGITAL E CIRCUITOS**

- 
- 83 Introdução
  - 84 Conceitos de Lógica Digital
  - 85 Operadores Lógicos e Portas Lógicas
  - 92 Expressões Lógicas e Circuitos Digitais
  - 98 Noções de Álgebra Booleana
  - 101 Considerações Finais
  - 108 Referências
  - 109 Gabarito

## UNIDADE IV

### **PROCESSADOR E MEMÓRIA**

- 
- 113 Introdução
  - 114 Organização do Processador
  - 117 Funcionamento do Processador
  - 121 Paralelismo
  - 127 Conceito de Memória
  - 129 Hierarquia de Memória
  - 136 Considerações Finais
  - 144 Referências
  - 145 Gabarito



# SUMÁRIO

 UNIDADE V

## INTERAÇÃO HOMEM-MÁQUINA

---

149 Introdução

---

150 Entrada e Saída (E/S)

---

158 Sistemas Operacionais

---

163 Aplicativos e Desenvolvimento

---

167 Licenças de Software

---

170 Considerações Finais

---

179 Referências

---

180 Gabarito

**181 CONCLUSÃO**

**182 ANOTAÇÕES**





# FUNDAMENTOS DA COMPUTAÇÃO

UNIDADE

I

## Objetivos de Aprendizagem

- Conhecer alguns dos principais conceitos da área de computação.
- Descobrir a origem dos equipamentos que usamos hoje e como foram evoluindo.
- Entender o que são os sistemas de computação e como eles funcionam em parceria com os equipamentos.

## Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Conceitos gerais
- História da computação
- Sistemas de computação



## INTRODUÇÃO

Olá jovem padawan, seja bem-vindo ao início do nosso estudo, o qual conceituaremos melhor aquilo que conhecemos por “computador” e algumas coisas que o cercam. Por ser uma disciplina introdutória, veremos um pouco sobre cada parte do computador, sem aprofundar muito em uma parte específica, mas a minha intenção com esse texto é dar o caminho das pedras para que você comprehenda as bases e como pode aprofundar no assunto por conta própria.

Em nossa área de T.I., tomamos um caminho sem volta de ter a obrigação de estarmos sempre atualizados no conteúdo. Portanto, as bases que vocês aprenderão aqui permanecem, mas as tecnologias podem mudar e sabemos que elas estão em constante evolução. Nesta primeira unidade, começaremos apresentando alguns conceitos importantes, como o que é um computador, em que consiste a informação e o processamento de dados, de forma a criar um ponto de partida para o nosso conhecimento que será adquirido.

Em seguida, passaremos a estudar a história dos computadores. E nesse ponto você se pergunta: “Mas para quê? Eu não quero trabalhar com computadores antigos, e sim com computadores modernos”. Bom, neste momento, nós entenderemos melhor porque o nosso computador moderno é feito dessa forma, o que as máquinas têm em comum independentemente de ser um computador pessoal, um smartphone ou um supercomputador, além de aprender com os gigantes do passado. E o mais legal de nossa área é que esse passado é recente, sendo que muitas das lendas, ainda, estão vivas e podemos até trocar ideias com alguns pelo Twitter.

Por fim, veremos como funcionam os sistemas de computação, muitas vezes, é o que mais nos chama a atenção para estudar essa área. Estudaremos, então, como é constituído um sistema de computador, como ele é executado e o que ele precisa para existir. Acredito que esse ponto jogue uma boa luz sobre os motivos de estudar todo o resto. Então, aperte os cintos, prepare um bom café e abra a mente, porque muito conhecimento vem por aí!



## CONCEITOS GERAIS

Olá aluno(a), tire uns breves segundos agora para se perguntar o porquê de estar fazendo este curso. Essa pergunta não é fácil para todo mundo, mas para alguns costumam surgir algumas ideias: “eu gosto de computadores”, “não vivo sem meus *apps* e quero aprender a fazer um”, “já sei programar um pouco”, “meus pais me obrigaram a isso...”. Bom, independentemente dos seus motivos pessoais, para iniciar nessa área, algumas perguntas precisam ser respondidas (mesmo que você nunca as tenham feito) e isso nós iniciaremos aqui.

Como a nossa área é uma área em constante evolução e a tecnologia muda de forma muito rápida, é difícil definir qual é a abrangência total dela. Não conseguimos demarcar com precisão os limites de onde ela começa e onde ela termina. Até a forma em que ela é chamada já mudou um tanto com o tempo, apesar de que os conceitos ainda permanecem válidos.

Podemos ouvir referências à área como “computação”, “informática”, “processamento de dados”, ou ainda como “tecnologia da informação” (T.I.), que é um dos termos mais utilizados no momento. Ainda, há quem defende que o termo já deve ser “tecnologia da informação e comunicações” (T.I.C.), que está correto, mas costuma-se usar T.I. por ser menor.

Aqui, já podemos ver como a nossa área está constituída ao redor de dois termos importantes: **informação e dados**. E cada vez mais, observamos no mundo que informações e dados valem muito mais do que os equipamentos de tecnologia.

Segundo Monteiro (2001), um **computador** é uma máquina capaz de manipular informações para um ou mais objetivos, ou ainda, é algo que pode ser chamado de “*equipamento de processamento eletrônico de dados*”. Ainda, indica que um **dado** “*pode ser definido como a matéria-prima originalmente obtida de uma ou mais fontes*”, enquanto a **informação** seria o resultado do processamento desses dados.

Sendo assim, os dados seriam resultados diretos de uma observação de algo no mundo real, mas que podemos manipular ou dar valor a eles tornando-os como uma informação.



SAIBA MAIS

Um termo que está em alta hoje em dia é o “Big Data”. Em linhas gerais, Big Data nada mais é do que a análise de uma quantidade de dados imensa de forma a extrair deles uma informação até então desconhecida, mas que tenha algum valor. É lógico que para trabalhar com tal volume de dados precisaremos de ferramentas e técnicas específicas, mas isso é um assunto para um outro livro.

Fonte: o autor.

De outra forma, um computador pode ser definido como sendo uma máquina que executa *instruções* para resolver determinados problemas e uma sequência de instruções é o que conhecemos por **programa** (TANENBAUM, 2013).

“*Tudo bem, eu sei o que é um computador, uso um desde que nasci*”, alguém pode estar pensando, “*Por que eu tenho que estudar arquitetura de computadores? Eu só quero programar.*”

Nas primeiras etapas da programação, você aprende que programar é praticamente dizer a um computador o que ele deve fazer, mas com uma diferença básica de quando a gente pede algo para uma pessoa: você precisa entender como o computador pensa.

O computador responde as instruções não ambíguas pré-definidas, que são facilitadas pelas linguagens de programação. Por que isso? Por que ele precisa de instruções? Como ele executa essas instruções? O que é que está deixando o meu programa tão lento?

Entendendo como é a arquitetura do seu computador você pode criar programas melhores, mais bem estruturados, e identificar de forma mais rápida onde você pode conseguir um ganho de desempenho. Entender como a informação é armazenada na memória e como o processador trabalha, essas informações podem fazer muita diferença em seu trabalho como T.I., seja qual for a especialização.

Portanto, para começar a entender melhor, podemos entender a **arquitetura de computadores** como os “atributos de um sistema visíveis a um programador ou, em outras palavras, aqueles atributos que possuem um impacto direto sobre a execução lógica de um programa” (STALLINGS, 2010, p. 6), enquanto o termo **organização de computadores** se refere a como os componentes do computador são combinados e organizados, de acordo com as especificações da arquitetura (STALLINGS, 2010).

Em outras palavras, podemos ter computadores que seguem uma determinada arquitetura, mas que possuem organizações diferentes. Isso acontece quando encontramos, por exemplo, computadores de um mesmo fabricante, com as mesmas características, mas com variações entre os modelos que podem implicar em diferentes desempenhos e preços, comumente, são chamados de “família”.

## HISTÓRIA DA COMPUTAÇÃO

Estamos em uma área do conhecimento que possui uma história recente e, inclusive, estamos vivendo a escrita de partes importantes da história neste momento.

Você já parou para pensar que muitas das lendas da nossa área ainda estão vivas e acessíveis? Diferente, por exemplo, de Aristóteles ou Sócrates para a filosofia e matemática. Hoje, você pode ir a congressos com criadores de tecnologias que foram grandes marcos na história da computação, você pode assisti-los pelo YouTube, conversar com eles pelo Twitter etc., mas a história da computação é um pouco mais antiga do que parece. Há quanto tempo você acha que existem os computadores? Quando você acha que foi escrito o primeiro programa de computador?

Para facilitar o estudo da história da computação, é comum encontrar essa história dividida em gerações, que veremos a seguir. Tenha em mente que a divisão das gerações não tem um momento exato, a tecnologia foi evoluindo e, olhando para trás, alguns marcos foram estabelecidos para dizer que uma nova geração se iniciou.

E você percebe, na literatura, que não existe um consenso de quais seriam as datas ou mesmo o número de gerações pelas quais passamos.

## GERAÇÃO ZERO - PRÉ-HISTÓRIA DA COMPUTAÇÃO

A computação tem origem em estudos muito antigos, aproveitando diferentes áreas do conhecimento, como a matemática, a filosofia e a física. Costuma-se colocar como o primeiro mecanismo precedente ao computador o **ábaco**, que tem a sua origem na Mesopotâmia, há cerca de 5000 anos atrás (ARAGÃO, 2009).

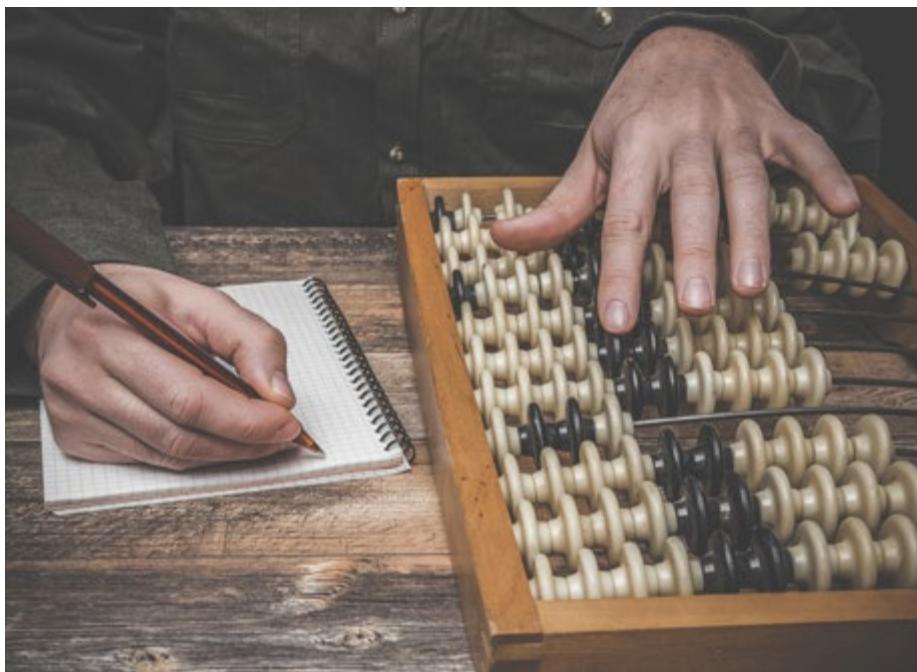


Figura 1 - Ábaco

O ábaco ganhou diversas variações em povos diferentes, mas tinha o mesmo objetivo que era o de auxiliar cálculos matemáticos, para não depender apenas da mente.

O desenvolvimento de equipamentos ou máquinas analógicas, no entanto, teve um processo muito lento, sendo que a primeira máquina de calcular mecânica veio a surgir apenas em 1642, criada por **Blaise Pascal**, que tinha 17 anos na época e queria ajudar seu pai no trabalho com as finanças e cálculo de impostos (ARAGÃO, 2009). A sua máquina realizava apenas as operações de soma e subtração, utilizando rodas e engrenagens dentadas (MONTEIRO, 2001).

A máquina de Pascal possuía já a habilidade de calcular o “vai um”. Os números eram representados por engrenagens que iam de 0 a 9, assim que a engrenagem passasse de 9 de volta para zero, ela movia a próxima, aumentando em 1. O mesmo sistema que você já deve ter visto em odômetros no painel de carros.

Depois de Pascal, um outro matemático chamado **Leibniz**, em 1671, construiu uma máquina capaz de efetuar multiplicações, usando um “espião dentado” (ARAGÃO, 2009). A máquina de Leibniz era, praticamente, a máquina de Pascal acrescida de dois conjuntos de rodas que permitiam multiplicação e divisão por operações sucessivas e ambas as máquinas eram manuais (MONTEIRO, 2001).

Algumas outras máquinas para cálculos foram desenvolvidas, mas vale ressaltar aqui o trabalho de **Joseph Jacquard**, que, em 1801, produziu uma máquina de tecelagem controlada por cartões perfurados. O conceito de cartões perfurados persistiu até mais da metade do século XX e, ainda, é utilizada uma variação em folhas de gabaritos de provas e concursos, que são marcadas as informações para serem lidas de forma automatizada.

Chegamos a um momento importante da história, onde aparece o considerado pai da computação, o britânico **Charles Babbage** (1791-1871). Ele desenvolveu duas máquinas mecânicas para cálculos: a *máquina de diferenças* e a *máquina analítica* (MONTEIRO, 2001).

A máquina de diferenças era uma máquina que realizava cálculos sucessivos e elaborados para a marinha real inglesa. Essa máquina trabalhava com o sistema decimal, era acionada por um motor movido a vapor.

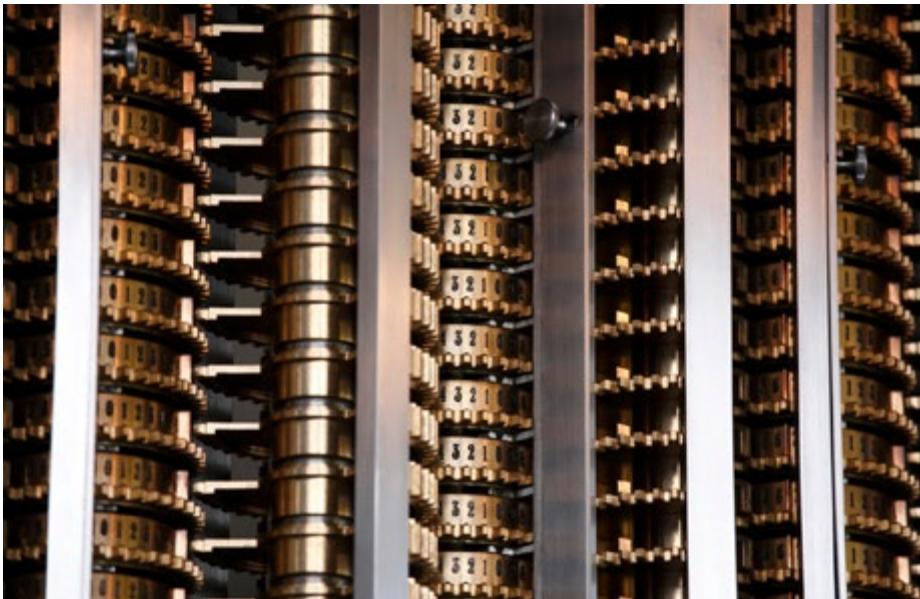


Figura 2 - Reprodução da máquina analítica de Babbage

Em 1833, Babbage apresentou, então, a máquina analítica (ou engenho analítico), que era uma máquina de cálculos controlada por cartões perfurados e que permitia que diferentes operações fossem realizadas de acordo com a programação do cartão perfurado (ARAGÃO, 2009), sendo assim, a primeira máquina programável da história.

Babbage trabalhava junto com **Ada Lovelace** (ou Augusta Ada Byron, filha do poeta Lord Byron e condessa de Lovelace), que foi quem criou os exemplos de funcionamento da máquina analítica e que é a primeira pessoa a escrever um programa de computador na história (SCHWARTZ, 2006). Ada morreu cedo, aos 36 anos, devido a problemas de saúde. Babbage e Ada não chegaram a completar o seu trabalho, em parte por falta de financiamento e por estarem muito à frente de seu tempo.

A máquina analítica permitia armazenar 1000 números de 20 algarismos e possuía os mesmos mecanismos de um computador atual: memória, processador, entrada e saída (MONTEIRO, 2001). Em 1854, o matemático **George Boole** publicou sua obra *The Laws of Thought* (As Leis do Pensamento), a qual ele descreveu o que hoje conhecemos como **álgebra booleana**, que consiste na manipulação de objetos que podem receber apenas valores verdadeiros ou falsos (0 ou 1), que é a base para os circuitos digitais (NULL, 2011).

Depois, houve um grande avanço, em 1889, com **Herman Hollerith**, que desenvolveu o cartão perfurado para armazenamento de dados e uma máquina tabuladora mecânica que contava e ordenava as informações contidas nos cartões. Seu invento teve muito sucesso após ser usado no censo dos Estados Unidos, o que levou à criação da *Tabulating Machine Company*, em 1896. Em 1914, essa empresa se uniu a mais outras duas e teve, em 1924, o seu nome alterado para *International Business Machines*, que conhecemos muito bem como **IBM** (MONTEIRO, 2001).

Já no começo do século XX, o avanço foi ainda mais rápido. Em 1935, o alemão **Konrad Zuse** criou a primeira máquina de calcular eletrônica e, no ano seguinte, a máquina Z1, que utilizava um teclado como entrada e lâmpadas como saída, já utilizando os conceitos binários de “aceso” ou “apagado” (MONTEIRO, 2011).

Em 1941, ele criou o Z3, uma máquina aperfeiçoadas, utilizando relés eletromecânicos e controlada por programa, o que pode ser considerado o “primeiro computador efetivamente operacional” (MONTEIRO, 2011). Depois, ainda criou o Z4, que foi utilizado pelos alemães para fins militares na Segunda Guerra Mundial, mas boa parte do seu trabalho e suas máquinas foram destruídas no meio da guerra.



SAIBA MAIS

O primeiro “bug” (inseto, em inglês) de computador ocorreu em 1945 numa máquina Harvard Mark II da IBM (sigla para International Business Machines) e foi provocado por um inseto de verdade. Uma mariposa entrou pela janela e entrou na máquina Mark II, na universidade de Harvard e travou todo o sistema. O inseto foi descoberto por Grace Hopper, que não conseguia descobrir por que o computador estava com uma pane, até que ele descobriu a mariposa nos contatos de um relé. Grace teve que tirar o inseto com uma pinça e disse que “estava tirando o bug da máquina” (debugging). Depois, colocou a mariposa em seu caderno de anotações com uma fita adesiva (foto) e escreveu tecnicamente em seu Relatório de Manutenção “primeiro caso de bug realmente encontrado”.

Fonte: Dias (2017, on-line)<sup>1</sup>.

No lado dos Estados Unidos, **Howard Aiken** inventou, em 1944, um computador eletromecânico chamado de “Mark I”, que utilizava os princípios da máquina analítica de Babbage, usando ainda o sistema decimal (MONTEIRO, 2011).

## PRIMEIRA GERAÇÃO: COMPUTADORES A VÁLVULAS (1946-1954)

A primeira geração de computadores ficou marcada pelo surgimento dos computadores a válvulas, como o ENIAC, que foi lançado em 1946 e era composto de quase 18 mil válvulas, 1.500 relés, pesava 30 toneladas e consumia 140 kw de energia (TANENBAUM, 2013). Para entender a primeira geração, precisamos entender que ela começou efetivamente alguns anos antes de seu marco inicial.



Figura 3 - Exemplos de válvulas

Em 1939, **John Vincent Atanasoff** projetou o primeiro computador eletrônico, ao invés de mecânico. Ao tentar construir uma máquina baseada nos trabalhos de Pascal e Babbage, ele refletiu sobre os erros dos projetos anteriores e decidiu utilizar eletricidade no lugar de processos mecânicos, adotando o sistema binário ao invés do decimal, e capacitores para funcionarem como a memória para guardarem a energia (NULL, 2011).

O trabalho de Atanasoff chamou a atenção de **John Mauchly**, que junto com **John P. Eckert** projetaram o ENIAC. A intenção do ENIAC era ser utilizado para elaboração de tabelas de alcance e trajetória para armas balísticas, mas o seu desenvolvimento durou de 1943 a 1946, sendo finalizado após o fim da Guerra. Ainda assim, funcionou para outros fins até 1955, servindo de base para diversos outros trabalhos (MONTEIRO, 2011).

O trabalho de programação do ENIAC ficou por conta de seis mulheres, matemáticas, que se tornaram as primeiras programadoras profissionais da história: Frances Bilas, Jean Jennings, Ruth Licherman, Kathleen McNulty, Betty Snyder e Marlyn Wescott. Tem mais informações sobre elas no texto de “leitura complementar” ao final da unidade.

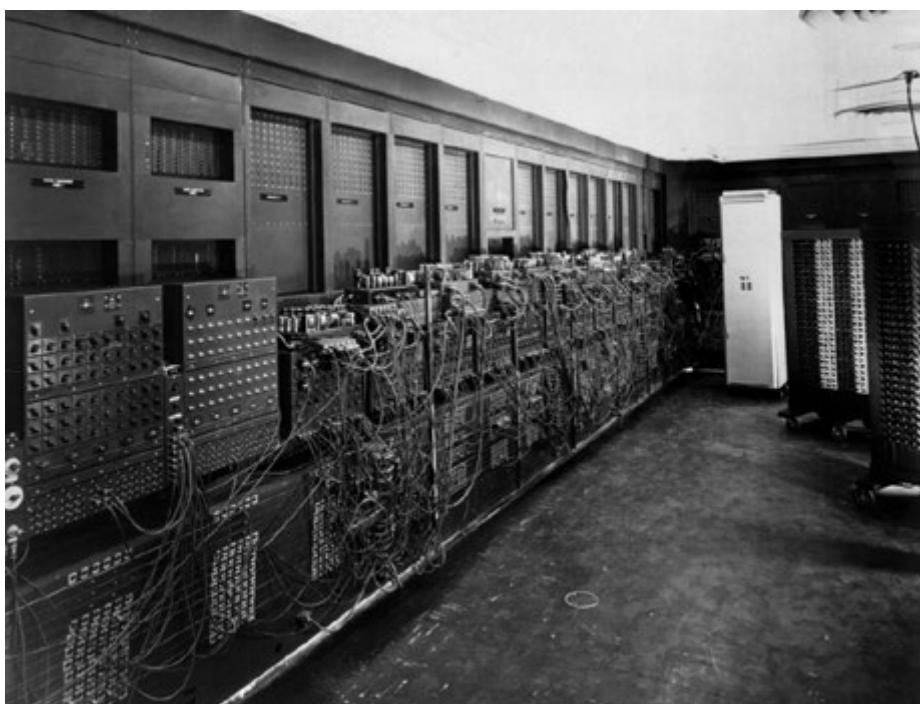


Figura 4 - ENIAC: quase 18 mil válvulas e 30 toneladas

Após o ENIAC, Mauchly e Eckert começaram a desenvolver o EDVAC, mas eles deixaram a Universidade da Pensilvânia para criar a própria empresa, onde passaram a desenvolver o UNIVAC.

SAIBA MAIS



Alan Turing teve um papel excepcional para a computação em diversas áreas e seus trabalhos serviram de base para muitos cientistas. Em 1943, ele e sua equipe colocaram em funcionamento um computador eletrônico ultrassecreto, a bomba eletromecânica (ou 'bombe'), COLOSSUS, que funcionava com o único objetivo de quebrar a criptografia da máquina alemã ENIGMA. Esse projeto ficou em segredo por 30 anos, o que acabou não servindo de base para nenhum outro projeto. Recentemente, essa história foi descrita com detalhes no filme "O Jogo da Imitação".

Fonte: o autor.

No mesmo momento, **John von Neumann**, que também colaborou no ENIAC, começou a desenvolver sua própria versão do EDVAC, a máquina IAS (que tinha o mesmo nome do instituto onde ele foi trabalhar) (TANENBAUM, 2013).

Von Neumann era um gênio, da mesma estirpe de Leonardo da Vinci. Falava muitos idiomas, era especialista em ciências físicas e matemática e guardava na memória tudo o que já tinha ouvido, visto ou lido. Conseguiu citar sem consulta, palavra por palavra, o texto de livros que tinha lido anos antes. Na época em que se interessou por computadores, já era o mais eminente matemático do mundo (TANENBAUM, 2013, p. 14).

Neste momento, von Neumann deu uma das maiores contribuições ao desenvolvimento de praticamente todos os computadores implementados da segunda geração até hoje: **o modelo de arquitetura de von Neumann**.

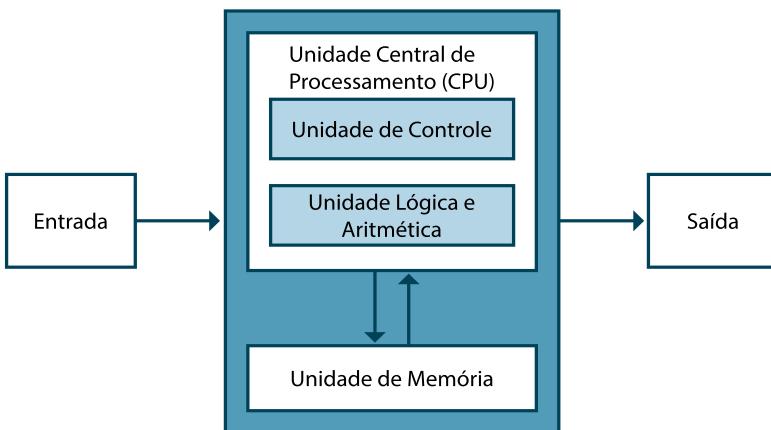


Figura 5 - Modelo de arquitetura de von Neumann

Fonte: o autor.

Ao projetar a máquina IAS, von Neumann percebeu que um programa poderia ser representado de forma digital, na memória. Passou a usar a aritmética binária, ao invés da decimal usada no ENIAC. No fim, a máquina ficou com 5 partes: a Unidade Lógica e Aritmética, a Unidade de Controle, a memória e os equipamentos de entrada e saída (TANENBAUM, 2013).



SAIBA MAIS

Como um aporte legal, Eckert e Mauchly solicitaram uma patente alegando que haviam inventado o computador digital. Em retrospecto, possuir essa patente não seria nada mau. Após anos de litígio, o tribunal decidiu que a patente de Eckert-Mauchly era inválida e que John Atanasoff tinha inventado o computador digital, embora nunca o tivesse patenteado, colocando efetivamente a invenção em domínio público.

Fonte: Tanenbaum (2013).

A IBM, que até então estava mais voltada a desenvolver máquinas de cartões perfurados, entrou no ramo de computadores apenas em 1953, quando lançou o IBM-701, que era mais direcionado a aplicações científicas, e em 1955, o IBM-702, com recursos de hardware voltados a aplicações comerciais (STALLINGS, 2010).

Em 1956, a IBM lançou o IBM-704, com 4K palavras de memória, e em 1958, lançou o IBM-709, que era mais aperfeiçoadas. Neste momento, a IBM se destacou em um mercado que era dominado pela UNIVAC desde 1950 (MONTEIRO, 2002).

## SEGUNDA GERAÇÃO: TRANSISTORES (1955-1964)

Qual você acha que foi a maior invenção do século XX? Os computadores? A exploração espacial? A Internet? Muitos consideram os transistores como a maior invenção (ou pelo menos entre as maiores) do século XX, pois eles possibilitaram todos os equipamentos eletrônicos modernos.

O **transistor** foi inventado em 1947, no Bell Laboratories, com o objetivo de substituir as válvulas. Os transistores tinham as vantagens de serem menores, geravam menos calor e eram mais baratos (STALLINGS, 2010). Além de baratear as máquinas, possibilitaram a redução do seu tamanho.



Figura 6 - Alguns tipos de transistores

Apenas na década de 1950 que surgiram os primeiros computadores transistorizados. O primeiro desses computadores foi o **TX-0**, construído no MIT, em 1955. Dois anos mais tarde, em 1957, Kenneth Olsen, que era um dos engenheiros do MIT, fundou a Digital Equipment Corporation - **DEC** (TANENBAUM, 2013), que veio a ser, mais tarde, a segunda maior fabricante de computadores do mundo, atrás só da IBM (MONTEIRO, 2002).

A IBM entrou na era dos computadores transistorizados mudando a sua linha de 700 para 7000. O primeiro desses computadores foi o IBM-7090 (1959), que era a versão transistorizada do IBM-709, mais tarde, lançou o IBM-7094 (1962), com ciclos de 2 microssegundos e 32.768 palavras de 36 bits de memória (TANENBAUM, 2013). Enquanto o 7094 fazia sucesso na área científica, a IBM também lançou o IBM-1401, que foi um sucesso na área comercial (MONTEIRO, 2002).

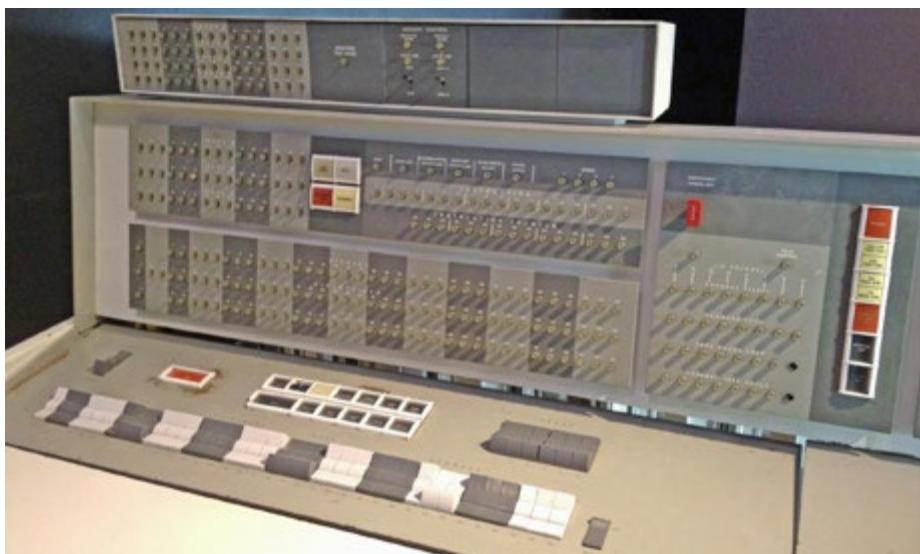


Figura 7 - IBM 7094 (Foto por Arnold Reinhold - CC BY-SA 3.0)

Fonte: Wikimedia Commons (2012, on-line)<sup>2</sup>.

A DEC colocou seu primeiro computador, o **PDP-1**, no mercado apenas em 1961, trabalhando com 4k palavras de 18 bits, podendo executar 200 mil instruções por segundo. Tinha a metade do desempenho do IBM-7090, mas custava 120 mil dólares, enquanto a máquina da IBM custava milhões (TANENBAUM, 2013).

Após o PDP-1, a DEC lançou uma série de máquinas até a mais famosa PDP-11, que teve grande aceitação do mercado. A DEC foi líder do mercado de minicomputadores por um bom tempo, primeiro com a linha PDP, depois com a linha VAX (MONTEIRO, 2002).

## TERCEIRA GERAÇÃO: CIRCUITOS INTEGRADOS (1964-1977)

À medida em que os computadores foram ficando mais complexos, começou a se tornar um problema a acomodação de seus componentes. Esse problema começou a ser resolvido em 1958, quando **Jack Kilby**, da Texas Instruments Co., colocou dois circuitos em uma única peça de germânio. Aproveitando a ideia, **Robert Noyce**, da Fairchild Semicondutor Inc., integrou múltiplos componentes em um substrato de silício (MONTEIRO, 2002).



Figura 8 - Circuitos integrados

Esse foi o início dos **circuitos integrados**, também chamados de **microchips**, que consistiam em placas que continham inicialmente dezenas, logo centenas de circuitos em um único componente, e milhares de circuitos, no que foi chamado de **LSI - Large Scale Integration** (integração em grande escala). Esse foi o início da **microeletrônica**.

Nesta época, a IBM tinha os seus principais computadores, o 7094 e o 1401, que eram totalmente incompatíveis. Uma utilizava aritmética binária com palavra de 36 bits, enquanto a outra utilizava aritmética decimal com palavras de tamanho variável (TANENBAUM, 2013).

Então, em 1964, a IBM lançou a família **System/360**, que consistia em computadores que utilizavam circuitos integrados, porém lançados em 5 diferentes modelos ao mesmo tempo (30, 40, 50, 65 e 75), com diferentes configurações, mas seguindo a mesma arquitetura (MONTEIRO, 2002).

Com a família 360, também, foi introduzida a técnica de multiprogramação, a qual vários programas poderiam compartilhar a memória e o uso da UCP, de forma aparentemente simultânea. Além de trabalhar com palavras de 32 bits, 16 M bytes de memória, memória principal orientada a bytes (como continua comum até hoje) e um sistema operacional para gerenciar os recursos de hardware, o OS/360 (MONTEIRO, 2002).

Enquanto isso, a DEC lançou o PDP-11, que seria o concorrente mais próximo da linha 360 da IBM, também, trabalhando com memória orientada a bytes e registradores orientados a palavras (TANENBAUM, 2013).

## QUARTA GERAÇÃO: O PC E OS CIRCUITOS VLSI (1977-1991)

Os circuitos integrados foram evoluindo, até que na década de 1980 já contavam com dezenas de milhares de circuitos por componente, os circuitos integrados **VLSI** - *Very Large Scale Integration* (integração em escala muito grande).

Com essa integração em pequenos componentes, foi possível também a diminuição do tamanho das máquinas e os componentes dos, até então, minicomputadores, também, foram tendo uma grande redução de preços (STALLINGS, 2010). Mais uma grande revolução começou em 1971, quando uma empresa criada para produzir componentes eletrônicos produziu a primeira UCP (Unidade Central de Processamento) em uma só pastilha de circuito integrado. Essa empresa se chamava **Intel Corporation** e o primeiro processador em circuito integrado foi o **Intel 4004** (MONTEIRO, 2002).

Enquanto o 4004 possuía palavra de 4 bits e cerca de 2300 transistores na pastilha, logo foi lançado o 8008, com palavra de 8 bits e cerca de 3500 transistores, mas o primeiro microprocessador para uso geral foi lançado em 1973, o **Intel 8080**, com cerca de 5000 transistores, palavras de 8 bits, endereços de memória de 16 bits, permitindo o uso de 64Kbytes de memória, e um conjunto de 78 instruções (MONTEIRO, 2002).

Começou, então, a era dos PCs (*Personal Computer* - Computador Pessoal), inicialmente com o **Altair 8800**, da empresa MITS, lançado em 1975, com um processador Intel 8080. Ele possuía um interpretador da linguagem BASIC que foi criado por **Bill Gates** e **Paul Allen**, o que também foi o pontapé inicial de criação da **MicroSoft** (originalmente Micro-Soft) (MONTEIRO, 2002).

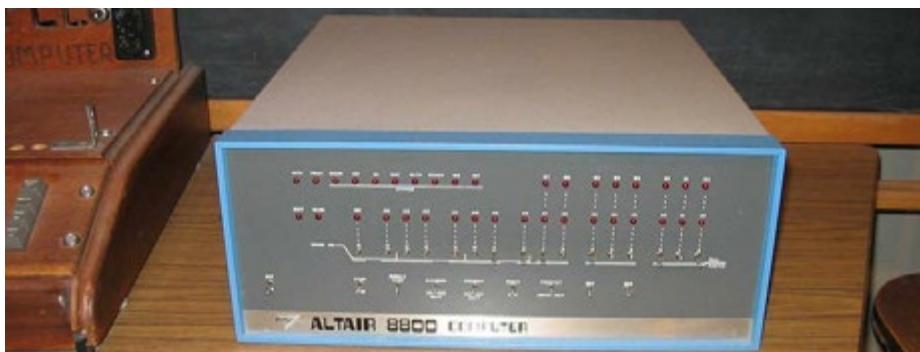


Figura 9 - Altair 8800

Fonte: Wikimedia Commons (2005, on-line)<sup>3</sup>

Em 1976, surgiu um importante concorrente no mercado. Uma empresa, “fundada em uma garagem” (a história foi desmentida recentemente), chamada **Apple**, entrou no mercado com seu primeiro PC, o **Apple I**.

**Steve Jobs** e **Steve Wozniak**, os fundadores da Apple, apresentaram, em 1977, o **Apple II**. Segundo o próprio Wozniak, o Apple II foi a primeira versão vendida já montada, enquanto a versão anterior era vendida em partes, para o próprio usuário montar. Para muitos, o Apple II foi um PC digno de marcar o início dessa geração, enquanto algumas literaturas marcam a quarta geração, iniciando junto com a criação do microprocessador.

Também, nessa época, a **Motorola** estava trabalhando com processadores. Em 1974, lançou o MC6800 com 8 bits, e em 1979, lançou o MC68000, já contando com palavras de 32 bits (a Intel começou a trabalhar com 32 bits apenas em 1985) (MONTEIRO, 2002).

Surgiram, ainda, PCs da **Comodore**, como o **PET**, em 1977, e o **Amiga**, em 1985. Outra concorrente da época era a **Atari**, que, aqui no Brasil, ficou famosa pelo videogame, mas produzia também PCs na década de 1980.

Em 1981, a IBM entrou no mercado de PCs com o **IBM-PC**, com o processador 8088 da Intel. O PC da IBM usava o sistema operacional **PC-DOS**. Por causa de problemas no licenciamento, a IBM precisou buscar uma alternativa e foi atrás da MicroSoft, que comprou uma versão do DOS e licenciou para a IBM com o nome **86-DOS**. A IBM o rebatizou como PC-DOS, como já costumava usar o nome, enquanto a Microsoft o atualizou e relançou como **MS-DOS**.



SAIBA MAIS

Todo ano a Revista Time elege a “pessoa do ano”, a qual elege um homem, mulher, grupo ou ideia, que “para o bem ou o mal” mais influenciou os eventos naquele ano. Em 1982, a “pessoa” do ano não foi uma pessoa, mas ficou como a “máquina do ano”, que era o computador. Esse evento destacava a diferença que os computadores pessoais estavam fazendo na sociedade e como eles já estavam se tornando presentes no cotidiano.

Fonte: o autor.



Figura 10 - IBM 5160, uma versão do IBM PC com disco rígido embutido, lançado em 1983

A parceria IBM-Intel-MS perdurou por um bom tempo, enquanto a Motorola produzia os processadores para os computadores da Apple, Comodore e Atari. A IBM, porém decidiu manter o seu projeto de computador aberto, o que possibilitou o surgimento de inúmeros “**IBM clones**” (que eram de diversos fabricantes, mas compatíveis e utilizavam os mesmos sistemas), o que barateou, popularizou e estabeleceu firme no mercado a linha IBM PC (TANENMAUM, 2013).

Os processadores Intel foram evoluindo com o tempo. Em 1982, a Intel lançou o 80286 (que ficou conhecido como “286”), ainda com 16 bits. Em 1985, o 80386 (popular “386”), com 32 bits de palavra, e em 1989, o 80486 (conhecido como “486”), também 32 bits (MONTEIRO, 2002).

Em 1984, a Apple lançou a linha **Macintosh**, que revolucionou em termos de interface gráfica (a Apple já tinha lançado o Lisa antes com interface gráfica) com o conjunto que era chamado de *WIMP - Windows, icons, menus and pointer*. Este computador introduziu o conceito de janelas e mouse, que, na verdade, havia sido inventado em 1973 pela equipe de engenharia da Xerox, mas os diretores não acharam que era algo a se investir. O projeto foi vendido depois para a Apple.



Figura 11 - Apple Macintosh SE, 1987

Pouco depois, aproveitando os conceitos da Apple, a Microsoft lançou o **Windows**, que passou a ser padrão nos PCs e, nas versões iniciais, rodava sobre o MS-DOS. Vale mencionar que, em 1985, Steve Jobs foi demitido da Apple, empresa que ele mesmo fundou, e fundou a NeXT, uma outra companhia focada no desenvolvimento de plataformas de computador.

A linha PC cresceu muito e dominou essa fatia do mercado, fazendo antigas concorrentes como Comodore a Atari saírem de cena.

## QUINTA GERAÇÃO: MOBILIDADE E ULSI (1991)

Chegamos, então, à quinta geração e é muito difícil escrever sobre uma geração que ainda está aberta e em constante mudança. O desenvolvimento de equipamentos e tecnologias cresce de forma exponencial, tornando impossível mencionar tudo o que acontece.

Portanto, falaremos de pontos chaves da quinta geração, que conta com circuitos integrados **ULSI** - *Ultra Large Scale Integration*, que conseguem conter mais de um milhão de circuitos em um único chip (STALLINGS, 2010).

Neste momento, temos o **NeXTstation**, uma estação de trabalhos lançada pela NeXT, em 1990, já com uma aparência que estamos habituados para computadores desktop, com gabinete, monitor, teclado e mouse.

Em 1991, tivemos o lançamento ao público da **WWW - World Wide Web** que, para muitos, é a própria Internet, mas a Internet é uma rede antiga, que começou a ser constituída na década de 1960, inicialmente ligando bases militares e universidades. A WWW é constituída pela ideia de documentos interligados para construir o conhecimento, daí é que surgiram, então, os hipertextos, hiperlinks, navegadores, http etc.

Também, em 1991, foi criado o **Linux**, que é um sistema operacional em software livre, baseado no **Unix**, e que anos depois se tornou referência na linha de servidores e redes.

A partir da década de 1990, tivemos o foco voltado para mobilidade, com a adoção dos **laptops** (ou notebooks). Os laptops já vinham se desenvolvendo, junto com os PCs, na década de 1980, mas em 1990 passaram a se tornar mais próximos às pessoas. Em 1992, a IBM lançou o **ThinkPad**, que se popularizou bastante nas empresas e essa linha perdura até hoje, até 2005 pela própria IBM e depois pela **Lenovo**.

Ainda, houve muita evolução na área de telecomunicações e armazenamento. Passamos dos disquetes de 3½ polegadas, que armazenava 1,44Mb, para CD-ROMs, com 700Mb, depois, DVDs, Blu Rays etc. Os HDs (hard disks) também tiveram avanço muito grande em tamanho e em velocidade.

Na década de 1990, a Internet chegou para o uso doméstico, com a famosa (e saudosa) conexão discada, avançou para ADSL e Internet via Rádio, chegando às fibras ópticas. Sem mencionar as redes móveis, que são responsáveis por muito dos avanços.

Voltando aos computadores pessoais, a linha PC dominava a maior parte do mercado, graças aos IBM Clones e a parceria com a Microsoft. Em 1990, foi lançado o Windows 3.0. Em 1992, a versão 3.1, que ficou muito popular. Em 1995, o Windows 95, que modernizou muito a interface e já estava bem consolidado no mercado.

Nessa época, a Apple estava com pouco espaço e até com algumas ameaças de fechar. Em 1996, a Apple comprou a NeXT e, com isso, tinha de volta Steve Jobs em sua equipe, que foi muito responsável por avanços que vieram em seguida. Com o lançamento do **Mac OS X**, em 2001, a Apple começou a retornar à concorrência.

Um evento que vale mencionar foi o lançamento do **iPhone** pela Apple, em 2007, que mostrou como a evolução da tecnologia ia seguir dali por diante. Foi o começo da era dos *smartphones*, que logo veio seguido pelo sistema operacional da Google, o **Android**, que é baseado em Linux, em boa parte software livre, e que se tornou o principal concorrente ao **iOS**, sistema operacional do iPhone.



Figura 12 - iPhone, primeira geração

A partir dos *smartphones*, a ideia de mobilidade se expandiu muito mais. Vários negócios foram abertos na área de desenvolvimento de aplicativos móveis (bem conhecidos como *apps*), surgiram diversas *startups*, surgiram outros dispositivos móveis, como tablets, computação vestível (*wearables*), Internet das Coisas etc.

Não temos como saber ou dizer para onde a tecnologia está indo, mas tem muita coisa boa surgindo. Não sabemos como será que, no futuro, olharemos para trás para delimitar até quando consideraremos uma quinta geração, se já acabou, se não. Estão vindo, por aí, muitas tecnologias disruptivas, há um avanço muito grande da Inteligência Artificial e da robótica, há, ainda, os veículos autônomos, a comunicação mundial em tempo real, já trabalhamos com realidade aumentada e realidade virtual, ou seja, nós já vivemos, em boa parte, a realidade futurística sonhada nos anos 1980.

Agora, o que nos reserva o futuro? Isso você que está lendo esse livro poderá escrever na história.

## SISTEMAS DE COMPUTAÇÃO

Quando estudamos a história da computação, acabamos focando bastante no desenvolvimento de *hardware* e não tem como ser diferente, mas ao realizar uma graduação na área de desenvolvimento de sistemas, estamos bastante interessados em aprender sobre o *software*. A primeira noção que precisamos ter, neste ponto, é a de que um não vive sem o outro.

### HARDWARE E SOFTWARE

Primeiro, vamos fazer uma diferenciação simples entre hardware e software. Pegue uma máquina qualquer (um computador, um smartphone etc.). Desligue ela. Pronto, o que você ainda consegue ver e mexer é o hardware, o que sumiu é o software.

Desde sempre, os computadores possuem o hardware, que é todo o maquinário, e o software, que envolve a lógica de funcionamento, os programas em si, mas até a década de 1980, do século XX, o software não era algo que tinha valor, era apenas algo necessário para os computadores funcionarem.



Figura 13 - Desenvolvimento de software

Muitos atribuem ao Bill Gates e, consequentemente, à Microsoft, a mudança de paradigmas e a noção de valor que foi atribuída ao software. Gates criou a Microsoft com o objetivo exclusivo de trabalhar com desenvolvimento de software e começou a vender as licenças de software.

A partir daí, o software passou a ser visto de uma outra forma. As máquinas agora divulgavam o software que era utilizado e o desenvolvimento do próprio hardware, também, passou a se preocupar em suportar os softwares específicos.

A evolução do hardware e do software dependem uma da outra, por isso, precisamos entender um pouco de cada área para conseguir mais sucesso em nosso desenvolvimento.

## SISTEMAS

A palavra sistema é utilizada em muitos escopos diferentes. O que seria exatamente um sistema? Um sistema pode ser definido como um “conjunto de partes coordenadas que concorrem para a realização de um determinado objetivo” (MONTEIRO, 2002, p. 3). Com isso em mente, podemos até compreender melhor a importância de conhecer um pouco de cada parte do todo.

Em computação, um sistema, muitas vezes, é resumido em um **programa** de computador. Como um programa executa instruções para resolver um determinado problema e ele precisa de todo o computador para isso, se enquadra bem em nossa definição de sistema.

Na última unidade, conversaremos melhor sobre a parte de software, mas vamos pensar um pouco em como funciona um programa. Um programa é um conjunto de instruções as quais precisam ser passadas para o computador. Esse conjunto de instruções precisa seguir uma sintaxe que o computador entenda, aí é que entram as **linguagens de programação**.

As linguagens de programação podem ter diferentes **níveis de abstração**, sendo assim, elas podem estar mais próximas às linguagens de máquina (instruções em binário que os processadores entendem) ou mais próximas à linguagem natural (linguagem que falamos).

Assim, todas as instruções escritas em alguma linguagem de programação precisam de uma espécie de tradução (explicaremos melhor depois) para que sejam entendidas pela máquina e esse processo todo não precisa ser passado para o usuário, o usuário tem apenas contato com o sistema pronto.

Nas próximas unidades, veremos melhor como a informação é armazenada dentro das máquinas, como são formados os circuitos digitais que utilizamos e veremos a estrutura interna dos computadores, para termos um domínio melhor dessas máquinas fabulosas que usamos.

Aproveite, que agora a parte mais divertida irá começar!

## CONSIDERAÇÕES FINAIS

Valente aprendiz das artes *computadorísticas* (tudo bem, essa palavra não existe), espero que tenha se inspirado com o conteúdo desta unidade. O principal objetivo, aqui, foi o de apresentar a você as máquinas como algo que não é místico e desconhecido, mas como algo que tem a sua lógica, a sua organização, começou pequeno e foi crescendo de uma forma que agora a gente pode entender melhor para aproveitar ao máximo o seu desempenho.

Olhando para trás, vemos que o *smartphone* que está no bolso tem um poder computacional maior do que os computadores que possibilitaram a ida do homem à Lua. Já pensou nas inúmeras possibilidades que você tem de aproveitar essa capacidade e toda essa evolução?

Isaac Newton disse certa vez: “*Se vi mais longe foi por estar sobre os ombros de gigantes*”. Em nossa caminhada em T.I., só podemos fazer o que fazemos porque diversos gigantes abriram caminho. Então, o seu trabalho, hoje, tem por trás Charles Babbage, Alan Turing, Ada Lovelace, Margaret Hamilton, John von Neumann e diversos outros.

Espero que toda essa história nos permita a inspiração, de não apenas criar trechos de códigos que serão compilados com sucesso, gerando zeros e uns que se perderão na história, mas que a gente possa também criar a história e deixar a nossa marca, criando soluções para os mais diferentes problemas e, quem sabe, ser uma dessas lendas contemporâneas estudadas por nossos filhos.

A seguir, nós veremos como os dados se transformam em bits e os bits se transformam de volta em dados. Procure sempre entender melhor como a sua ferramenta de trabalho funciona, assim, você vai conseguir ser um profissional muito mais completo ao dominar o conhecimento.

Como disse uma vez Abraham Lincoln: “*Se tivesse seis horas para derrubar uma árvore, eu passaria as primeiras quatro horas afiando o machado*”.

## ATIVIDADES



1. "A tarefa de entrar e alterar programas para o ENIAC era extremamente enfadonha. O processo de programação poderia ser facilitado se o programa pudesse ser representado em uma forma adequada para armazenamento na memória junto com os dados. Então, um computador poderia obter suas instruções lendo-as da memória, e um programa poderia ser criado ou alterado definindo-se os valores de uma parte da memória. Essa ideia, conhecida como conceito de programa armazenado, normalmente é atribuída aos projetistas do ENIAC, principalmente o matemático John von Neumann, que foi consultor no projeto ENIAC. Alan Turing desenvolveu a ideia praticamente ao mesmo tempo. A primeira publicação da ideia foi em uma proposta de 1945 de von Neumann para um novo computador, o EDVAC (Electronic Discrete Variable Computer)".

STALLINGS, William. **Arquitetura e organização de computadores**. 8. ed. São Paulo: Pearson Prentice Hall, 2010. p. 13.

Sobre o modelo de arquitetura criado por von Neumann, assinale a alternativa correta.

- a) Ainda que desenvolvido na primeira metade do século XX, os computadores de hoje ainda seguem a arquitetura de Von Neumann.
- b) O padrão estabelecido por Von Neumann constituía em uma máquina teórica, com memória e processamento infinito, para fins de validação teórica.
- c) A arquitetura de Von Neumann foi utilizada apenas na primeira geração, sendo trocada pela arquitetura de Turing com a troca das válvulas por transistores.
- d) O padrão de Von Neumann incluía uma Unidade Aritmética e Lógica, que deixou de ser necessária quando os processadores englobaram essa funcionalidade.
- e) A arquitetura de Von Neumann foi muito importante nas primeiras gerações da computação, sendo substituída pela arquitetura dos PCs, que surgiram na década de 1980.

## ATIVIDADES



2. O desenvolvimento de circuitos integrados impulsionou muito o desenvolvimento de hardware, barateou custos e possibilitou a criação de equipamentos menores, o que foi um marco separador entre as gerações da computação. Considerando o texto supracitado e a história das gerações da computação, avalie as afirmações a seguir.

- I. Por se basearem em grafeno ao invés de silício, os circuitos integrados superaram os transistores em velocidade e desempenho.
- II. Os circuitos integrados permitiram a utilização de milhares ou milhões de transistores dentro de um único chip, possibilitando a criação de hardwares menores.
- III. Os chips de silício passaram a ser considerados um avanço em relação aos transistores por possuir confiabilidade, tamanho reduzido e custo baixo.
- IV. Os circuitos integrados passaram a ser utilizados apenas em dispositivos pequenos, como dispositivos móveis e equipamentos embarcados, enquanto os computadores continuam utilizando transistores.

Assinale a alternativa que apresenta as afirmativas corretas.

- a) Somente as afirmativas I e II estão corretas.
- b) Somente as afirmativas II e III estão corretas.
- c) Somente as afirmativas III e IV estão corretas.
- d) Somente as afirmativas I, II e III estão corretas.
- e) Somente as afirmativas II, III e IV estão corretas.

## ATIVIDADES



3. “O transistor foi inventado na Bell Laboratórios em 1947 e, por volta da década de 1950, deu início a uma revolução eletrônica. Porém, não foi antes da década de 1950 que os computadores transistorizados foram disponibilizados comercialmente. A IBM novamente foi a primeira empresa a oferecer a nova tecnologia. A NCR e, com mais sucesso, a RCA foram as pioneiras com algumas máquinas pequenas a transistor. A IBM veio pouco depois com a série 7000”.

STALLINGS, William. **Arquitetura e organização de computadores**. 8. ed. São Paulo: Pearson Prentice Hall, 2010, p. 19.

Considerando o texto anterior, leia as afirmativas e assinale Verdadeiro (V) ou Falso (F):

- ( ) Os transistores possuem a vantagem de serem menores, possibilitando criar circuitos equivalentes aos de válvulas em espaço bem menor.
- ( ) Apesar de todas as vantagens, a única desvantagem dos transistores era a de ser mais caro do que as válvulas, o que acaba sendo comum em novas tecnologias.
- ( ) Outra vantagem é que os transistores geram menos calor do que as válvulas e não precisam de pré-aquecimento.

A sequência correta para a resposta da questão é:

- a) V, V e F.
- b) F, F e V.
- c) V, F e V.
- d) F, F e F.
- e) V, V e V.

## ATIVIDADES



4. “À primeira vista, a impressão é de que um chip menor é mais fraco ou tem menor poder de processamento comparado a um chip maior, mas na evolução da computação foi diferente, cada vez mais eles foram ficando menores, com cada vez mais transistores e com um forte aumento de desempenho, conforme o passar das gerações. A tecnologia que possibilitou o avanço dessa época foi a VLSI (Very Large Scale Integration), traduzindo seria Integração em escala muito grande, que é a possibilidade de trabalhar com milhões de transistores em apenas um único chip”.

FEITOSA, Y. R. G. **Tecnologias Emergentes em TI**. Maringá: UniCesumar, 2017.

O texto retrata uma realidade da quarta geração de computadores, a qual os PCs foram os grandes precursores da mudança.

O lançamento dos PCs e início da quarta geração datam de qual década?

- a) Década de 1960.
- b) Década de 1970.
- c) Década de 1980.
- d) Década de 1990.
- e) Anos 2000.

5. Na primeira geração, os computadores continham milhares de \_\_\_\_\_, que foram substituídos, na segunda geração, por \_\_\_\_\_, e na terceira geração por \_\_\_\_\_, que conseguiram reduzir muito os custos e o tamanho dos dispositivos.

Considerando o texto acima, assinale a alternativa que preenche corretamente as lacunas.

- a) Válvulas, transistores e circuitos integrados.
- b) Transistores, válvulas e circuitos integrados.
- c) Válvulas, circuitos integrados e transistores.
- d) Circuitos integrados, válvulas e transistores.
- e) Transistores, circuitos integrados e válvulas.



## **As programadoras do ENIAC apagadas da história da computação**

No meio dos anos 80, uma estudante de ciência da computação de Harvard estava se sentindo isolada. À medida que o curso avançava, havia cada vez menos colegas mulheres, um sinal preocupante para seu futuro profissional. Desmotivada, Kathy Kleiman decidiu buscar modelos inspiradores, mulheres que tivessem tido papéis importantes na evolução da computação.

Em sua pesquisa, Kathy encontrou uma foto famosa do ENIAC, o primeiro computador totalmente eletrônico de uso geral. Publicada nos principais jornais dos Estados Unidos, na época de seu lançamento, em 1946, a imagem mostrava quatro homens e duas mulheres operando o computador, mas o que realmente chamou a atenção de Kathy foi um detalhe incômodo, apenas os homens da foto estavam identificados na legenda.

Kathy mostrou a foto a muitas pessoas, mas nem mesmo historiadores da computação sabiam quem eram aquelas mulheres. O mais provável, lhe disseram, era que as mulheres fossem "refrigerator ladies", ou seja, modelos que posavam junto de eletrodomésticos para torná-los mais atrativos, um truque de marketing bastante comum na época das fotos, mas esse palpite estava terrivelmente errado.

## **O nascimento do primeiro computador**

No final da Segunda Guerra Mundial, o exército (...) começou a recrutar pessoas formadas em matemática de todo o país para realizar as complexas equações diferenciais que traçavam a rota dos mísseis. E com a maioria dos homens lutando ou trabalhando em outras funções de guerra, essas pessoas eram todas mulheres. Seu título oficial era computer, "computadora".

Em alguns meses, havia uma centena de computadoras preenchendo tabelas balísticas manualmente, resolvendo folhas e mais folhas de equações contando apenas com a ajuda de calculadoras de mesa. Só tinha um problema: cada cálculo levava 30 horas.

(...)

O projeto foi tocado como segredo de guerra e liderado pelos engenheiros John Presper Eckert e John W. Mauchly. Na primavera de 1945, pouco tempo antes do ENIAC ficar pronto, seis computadoras foram escolhidas para o que era considerado um trabalho inferior comparado ao prodígio de engenharia que Eckert e Mauchly haviam criado: descobrir como a máquina funcionava e como ela deveria ser programada para executar os cálculos balísticos. As matemáticas selecionadas foram **Frances Bilas, Jean Jennings, Ruth Lichterman, Kathleen McNulty, Betty Snyder e Marlyn Wescoff**.

## **Programando um monstro**

O ENIAC era um monstro de 18.000 válvulas e um emaranhado de cabos e interruptores que ocupava um andar inteiro da Universidade da Pensilvânia. Tudo que as seis mulheres receberam para descobrir como programá-lo foram os diagramas lógicos dos 40 painéis que compunham o computador.





Não havia livros, sistema operacional, linguagem de programação ou ferramentas para ajudá-las. Programar o primeiro computador da história significava escrever toda a lógica no papel, reproduzi-la fisicamente com extrema precisão plugando e desplugando cabos e alternando interruptores e documentá-la nos mínimos detalhes para que pudesse ser reutilizada no futuro. Configurar um único programa levava várias semanas.

(...)

Depois de alguns meses, o trabalho foi recompensado: os cálculos balísticos que levavam 30 horas passaram a ser resolvidos em 15 segundos pelo ENIAC.

### O começo do esquecimento

(...)

O cálculo perfeito, a apresentação digna de filmes de ficção científica e o discurso futurista dos engenheiros garantiram o sucesso da festa de lançamento. Mauchly chegou a dizer à imprensa que, no futuro, máquinas como o ENIAC baixariam o preço do pão. Os jornalistas ficaram radiantes com a novidade, o ENIAC virou uma lenda e Eckert e Mauchly gravaram seus nomes permanentemente na história da computação.

Nenhuma das seis programadoras do ENIAC foi convidada para a festa.

### Legado

(...)

O ENIAC funcionou durante 10 anos e Frances, Jean, Ruth, Kathleen, Betty e Marlyn tornaram-se as primeiras programadoras profissionais, as primeiras professoras da programação moderna e as inventoras de ferramentas que abriram caminho para o software como conhecemos hoje.

**Frances Bilas** (depois Spence, 1922–2013) continuou programando equações com o ENIAC após a guerra e colaborou com os principais matemáticos do mundo.

**Jean Jennings** (depois Bartik, 1924–2011) trabalhou no time que transformou o ENIAC em uma máquina de programa armazenado, tornando mais rápido e fácil programar problemas mais complexos. Participou também dos projetos dos dois primeiros computadores comerciais, programando o BINAC e criando a lógica e um sistema de backup de memória eletrostática para o UNIVAC I. Mais tarde, criou relatórios para ajudar negócios a entenderem o potencial dos microcomputadores.

**Ruth Lichterman** (depois Teitelbaum, 1924–1986) foi realocada juntamente com o ENIAC para Aberdeen, Maryland, para ser a professora da próxima geração de programadores do projeto.

**Kathleen McNulty** (depois Mauchly Antonelli, 1921–2006) foi quem teve a ideia de criar subrotinas para conseguir calcular trajetórias que extrapolavam os limites computacionais do ENIAC. Essas subrotinas foram as precursoras das funções e tinham a nobre intenção de reaproveitar partes do programa que se repetiam. Se você pensou em DRY, acertou na mosca.





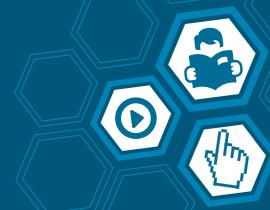
**Frances Elizabeth "Betty" Snyder** (depois Holberton, 1917–2001) trabalhou com Jean no UNIVAC I desenhando seu console de controle, teclado e teclado numérico. Também foi para esse projeto que ela escreveu o primeiro algoritmo de ordenação em 1952. Betty ainda escreveu padrões para o FORTRAN e participou de vários comitês nacionais e internacionais de computação como o que foi responsável pela criação do COBOL.

**Marlyn Wescoff** (depois Meltzer, 1922–2008) se desligou do projeto para se casar em 1947, antes do ENIAC ser deslocado para Aberdeen.

O trabalho delas alterou dramaticamente a computação nos anos 40 e 50, mas elas foram consideradas meras operadoras e não ganharam nenhum crédito até serem descobertas por uma estudante de Harvard 30 anos depois. (...)

Fonte: Bittencourt (2016, on-line)<sup>4</sup>.

# MATERIAL COMPLEMENTAR



LIVRO

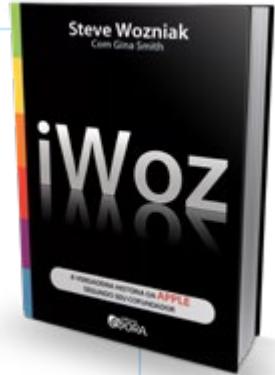
## iWoz - a Verdadeira História da Apple Segundo Seu Cofundador

Steve Wozniak e Gina Smith

**Editora:** Évora

**Sinopse:** antes dos BlackBerrys, PalmPilots e laptops que se encaixam em pastas, os computadores pareciam grandes e desajeitados. Eles tinham grandes telas, interruptores enigmáticos, caixas enormes e luzes estranhas. Mas, em 1975, um jovem assistente de engenharia chamado Steve Wozniak teve uma ideia: 'e se eu conseguir combinar circuitos de computador com um teclado de máquina de escrever e uma tela de vídeo?' O resultado foi verdadeiramente o primeiro computador pessoal, o Apple I. Amplamente acessíveis e de fácil compreensão, as invenções de Wozniak têm rapidamente transformado nosso mundo desde então, como o controle remoto universal.

A vida de Wozniak antes e depois da Apple é uma mistura de aventura com brilhantes descobertas, seja como engenheiro, promotor de concertos, professor, filantropo ou brincalhão irreprimível. Desde a invenção do primeiro computador pessoal até a ascensão da Apple como um gigante da indústria, iWoz apresenta uma história sem censura, divertida, que constrói um perfil em primeira mão do inventor humanista que iniciou a revolução do computador.



FILME

## Estrelas Além do Tempo

Ano: 2016

**Sinopse:** 1961. Em plena Guerra Fria, Estados Unidos e União Soviética disputam a supremacia na corrida espacial ao mesmo tempo em que a sociedade norte-americana lida com uma profunda cisão racial, entre brancos e negros. Tal situação é refletida também na NASA, onde um grupo de funcionárias negras é obrigada a trabalhar a parte. É lá que estão Katherine Johnson (Taraji P. Henson), Dorothy Vaughan (Octavia Spencer) e Mary Jackson (Janelle Monáe), grandes amigas que, além de provar sua competência dia após dia, precisam lidar com o preconceito arraigado para que consigam ascender na hierarquia da NASA.

**Comentário:** um ótimo filme para ver como foi a adoção da computação e da programação na década de 1960. Diversas pessoas eram contratadas apenas para fazerem cálculos e é impressionante como algumas se dedicaram a serem pioneiras em programação em uma época em que você tinha que aprender direto dos manuais.



# MATERIAL COMPLEMENTAR



FILME

## O jogo da imitação

Ano: 2014

**Sinopse:** Na década de 1930 a agência britânica MI6 foi criada e招招rou Alan Turing para auxiliar em trabalhos de codificação de mensagens nazistas durante a Segunda Guerra Mundial. O filme trata inicialmente o trabalho de desenvolvimento de um meio para decifrar estes códigos, mas depois trata questões da vida pessoal de Turing.



NA WEB

Websérie sobre as gerações da computação. Nós gravamos uma websérie, na Unicesumar, para falar sobre as diferentes gerações da computação em vídeos curtos.

<https://www.youtube.com/watch?v=LDz5lYoL6hU&list=PLhV6meBa8y2u5qKqR9sm53FxdyOFw06X7>



## REFERÊNCIAS

- ARAGÃO, M. J. **História da matemática**. Rio de Janeiro: Interciência, 2009. 212 p.
- MONTEIRO, M. A. **Introdução à organização de computadores**. 4. ed. Rio de Janeiro: LTC, 2001. 498 p.
- NULL, L.; LOBUR, J. **Princípios básicos de arquitetura e organização de computadores**. 2. ed. Porto Alegre: Bookman, 2011. 822 p.
- SCHWARTZ, J. et al. Mulheres na informática: quais foram as pioneiras?. **Cadernos Pagu**, Campinas, n. 27, p. 255-278, dez. 2006. Disponível em: <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0104-83332006000200010&lng=en&nrm=iso](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-83332006000200010&lng=en&nrm=iso)>. Acesso em: 21 ago. 2018.
- STALLINGS, W. **Arquitetura e organização de computadores**. 8. ed. São Paulo: Pearson Prentice Hall, 2010.
- TANENBAUM, A. S. **Organização estruturada de computadores**. 6. ed. São Paulo: Pearson Prentice Hall, 2013.

## REFERÊNCIAS ON-LINE

<sup>1</sup>Em: <<http://profjabiorritmo.blogspot.com/2017/06/o-primeiro-bug-em-um-computador-da-ibm.html>>. Acesso em: 21 ago. 2018.

<sup>2</sup>Em: <<https://commons.wikimedia.org/w/index.php?curid=18330258>>. Acesso em: 22 ago. 2018.

<sup>3</sup>Em: <[https://commons.wikimedia.org/wiki/File:Altair\\_8800.jpg](https://commons.wikimedia.org/wiki/File:Altair_8800.jpg)>. Acesso em: 22 ago. 2018.

<sup>4</sup>Em: <<https://lfbittencourt.com/mulheres-programadoras-eniac-b68503ef05f6>>. Acesso em: 21 ago. 2018.



## **GABARITO**

1. A.

2. B.

3. C.

4. B.

5. A.



# REPRESENTAÇÃO DE DADOS

UNIDADE



## Objetivos de Aprendizagem

- Conhecer as unidades utilizadas ao armazenar a informação.
- Entender como os números são representados em suas bases e como é calculado o valor.
- Aprender a transformar os números de uma base numérica para outra.

## Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Unidades de informação
- Notação posicional
- Conversões entre bases numéricas



## INTRODUÇÃO

Olá aluno(a), continuamos aqui a nossa saga em busca do conhecimento. Agora que você já sabe que está “sobre os ombros de gigantes”, chegaremos um pouco na prática do que esses gigantes já deixaram para nós.

No começo do livro, trocamos algumas ideias sobre o que vem a ser a informação e sabemos que ela é o maior motivo por estarmos aqui. O nosso computador é uma máquina de processamento de informações, mas como que uma informação (que no fundo não tem nenhum significado ou emoção para um computador) pode ser armazenada por ele, processada e, no fim, retornada com mais valor do que quando entrou?

Bom, ainda que o computador seja o seu melhor amigo (ele não estraga as coisas que você mais gosta, como o seu cachorro), o segredo para entender a manipulação de informações é não enxergá-lo como se fosse uma pessoa, mas uma máquina, que apenas trabalha com números, realiza operações sobre números e retorna números. Divertido, não?

Como todo mundo gosta de matemática (pode confessar), acredito que esse vai ser um capítulo divertido, até mesmo porque vamos mexer com números de uma forma que nunca mexemos na escola.

Primeiro, nós vamos ver como a informação é guardada dentro de um computador. Depois, vamos entender como funcionam os números e como entendemos o valor deles (sim, isso você viu de alguma forma na escola). Então, vamos converter números entre uma base e outra, para ver como um mesmo número pode ter diferentes representações. Por fim, veremos como funciona a aritmética em sistemas com base diferente de 10.

Fique tranquilo(a), por mais que façamos contas nesta unidade, são contas simples. Na sua vida, provavelmente, você não vai precisar fazer tais contas na mão, mas aqui nós vamos fazer para gravar bem os conceitos.

Pode acreditar, mesmo que indiretamente, isso faz uma boa diferença na sua compreensão da tecnologia. Então, faça algo que você não faz há muito tempo: pegue um papel e um lápis e vamos mergulhar entre os bits e bytes!



## UNIDADES DE INFORMAÇÃO

Como você deve se lembrar, já vimos, rapidamente, na história da computação algumas decisões tomadas sobre a melhor forma de representar a informação. Também, já falamos um pouco sobre o que vem a ser a própria informação ou os dados.

O computador não entende a *semântica*, ou seja, não entende o significado por trás da informação. Se você vê, por exemplo, um número 42, pra você ele pode ter diferentes significados ou lembranças.

Internamente, o computador é uma imensa máquina de calcular, extremamente, bem elaborada. Quando estudarmos sobre a Unidade Lógica e Aritmética do processador, na Unidade IV, veremos como o conjunto de operações que o processador realiza é até bem limitado.

E como conseguimos tantas informações diferentes? A mágica está em como os dados são armazenados e como são combinados, para trazerem o significado. Adiante vamos ver como isso é armazenado e recuperado no computador.

### BIT E BYTE

Inicialmente, os computadores mecânicos usavam engrenagens que representavam números de 0 a 9. Com o advento dos computadores eletrônicos, percebeu-se que a própria energia poderia ser usada para guardar dados em memória, utilizando capacitores (NULL, 2011). Basicamente, poderia guardar dois tipos de informação: tem energia ou não tem energia. Isso poderia representar os valores *verdadeiro* ou *falso*, ou ainda, 0 ou 1.

Sendo assim, podemos, então, entender a menor unidade de informação existente no computador, uma única informação que só pode ter dois estados diferentes, convencionados em 0 e 1: o **bit**.

No fundo, tudo acaba virando bit em um computador, tudo acaba sendo traduzido para sequências de *zeros* ou *uns*, mas como dá para imaginar, precisamos combinar vários desses bits a fim de conseguir representar informações mais complexas.

No início, cada projeto de arquitetura definia a quantidade de bits a se utilizar para cada informação. Existiam códigos baseados em 4 bits, 6 bits, 8 bits, 12 bits (...). Até que se convencionou o agrupamento em 8 bits, para facilitar, e esse agrupamento de 8 bits conhecemos como **byte**.

Portanto, em 1 byte, conseguimos guardar uma informação com 8 dígitos de 0 ou 1, ou seja, 8 *dígitos binários*. Se você se esforçar e lembrar das aulas de matemática do colégio, quantas combinações diferentes podemos guardar em 8 posições que podem ter 2 valores diferentes? Resumindo, multiplicamos as possibilidades de cada posição, dessa forma:

$$2 \times 2 = 2^8 = 256$$

Portanto, com apenas 1 byte de informação, conseguimos representar 256 números diferentes, ou seja, conseguimos representar uma sequência de números de 0 a 255.

Ainda, pode-se utilizar o conceito de **palavra** para definir o tamanho da informação. Basicamente, uma *palavra* é “um conjunto de bits que representa uma informação útil para os computadores” (MONTEIRO, 2002, p. 30). Sendo assim, é comum encontrar esse termo ao tratar de processadores, você pode encontrar processadores que trabalham com palavras de 32 bits ou palavras de 64 bits, por exemplo.

## UNIDADES DE MEDIDA DE INFORMAÇÃO

Ao precisar trabalhar com números maiores, mais informações, podemos agrupar os bytes, o que veremos melhor ao estudar a notação posicional, mas vale ressaltar agora, que a utilização de números binários e o agrupamento em 8 bits facilita bastante a representação de dados em um computador.

Como os bytes são escritos em base 2 e utiliza-se bastante as potências de dois, costuma-se agrupar os bytes de 1024 em 1024, ao invés de 1000 em 1000, como fazemos com números na representação decimal (já que 1024 equivale a  $2^{10}$ ). A seguir, apresentamos uma tabela que você já deve conhecer, pelo menos parcialmente, de multiplicadores de bytes.

Tabela 1 - Unidades de armazenamento e equivalências

VALOR	VALOR EM BYTES	UNIDADE
1KB	1.024 bytes	kilobyte
1MB	1.024 KB = 1.048.576 bytes	megabyte
1GB	1.024 MB = 1.073.741.824 bytes	gigabyte
1TB	1.024 GB = $1,099511628 \times 10^{12}$ bytes	terabyte
1PB	1.024 TB = $1,125899907 \times 10^{15}$ bytes	petabyte
1EB	1.024 PB = $1,152921505 \times 10^{18}$ bytes	exabyte
1ZB	1.024 EB = $1,180591621 \times 10^{21}$ bytes	zettabyte
1YB	1.024 ZB = $1,20892582 \times 10^{24}$ bytes	yottabyte

Fonte: o autor.

## REPRESENTAÇÃO DE DADOS

Todos os dados manipulados pelo seu computador são convertidos em bits, seja para serem guardados em memória principal, secundária, para serem processados, transferidos em rede etc. E como é que em uma imensidão de zeros e uns conseguem guardar números, letras, textos, imagens e tudo mais?

Para guardar números, a correlação é direta. Nos próximos tópicos, veremos como um número é transformado em binário e depois convertido de volta. Para guardar um texto, vejamos como os bits representam as letras, os textos são apenas sequências de letras, ou melhor, caracteres.

Existem diferentes sistemas de caracteres que podem ser usados, mas o sistema mais utilizado em computadores é conhecido como código **ASCII**. O código ASCII (*American Standard Code for Information Interchange* - código americano

padrão para troca de informações) utiliza 7 bits para representar um caractere, o que possibilita utilizar 128 caracteres diferentes (TANENBAUM, 2013). Com mais 1 bit, o padrão ganhou mais 128 caracteres, completou um byte, e também ficou conhecido como ASCII estendido.

Com isso, cada caractere de um texto ocupa apenas 1 byte de informação e cada byte pode representar um caractere da tabela ASCII. A tabela com 128 caracteres está apresentada a seguir.

Tabela 2 - Tabela ASCII com 128 caracteres

#	CHR	#	CHR	#	CHR	#	CHR										
0	NUL	16	DLE	32	SPC	48	0	64	@	80	P	96	`	112	p		
1	SOH	17	D1	33	!	49	1	65	A	81	Q	97	a	113	q		
2	STX	18	D2	34	"	50	2	66	B	82	R	98	b	114	r		
3	ETX	19	D3	35	#	51	3	67	C	83	S	99	c	115	s		
4	EOT	20	D4	36	\$	52	4	68	D	84	T	100	d	116	t		
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u		
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v		
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w		
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x		
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y		
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z		
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{		
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124			
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}		
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~		
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	Del		

Fonte: Arrigoni (2018, on-line)<sup>1</sup>.

Vários caracteres são “não-imprimíveis”, ou seja, caracteres de controle que não vemos, como quebras de linha, por exemplo. A partir do caractere 128 existem vários símbolos e algumas letras com acentuação.

Figura 1 - Monalisa em caracteres ASCII

Com a evolução dos computadores e da comunicação, fez-se necessária a inclusão de mais caracteres, como diferentes acentuações e diferentes alfabetos, como o cirílico, chinês, árabe e diversos outros. Depois de diferentes tentativas de expandir o código ASCII, um consórcio de empresas criou o padrão **Unicode**, que conta com 16 bits (2 bytes), podendo representar até 65.536 caracteres diferentes, mas os diferentes idiomas do mundo usam cerca de 200 mil símbolos (TANENBAUM, 2013).

Então, foi criado o padrão **UTF-8**, que é amplamente utilizado hoje. Em essência é o Unicode, armazenando caracteres em tamanho variável de 1 a 4 bytes. Com isso, para caracteres existentes na tabela ASCII, mantém o mesmo código e usa apenas 1 byte, sem desperdício, mas, ainda, permite caracteres em até 32 bits, ou seja  $2^{32}$  caracteres, o que dá cerca de 2 bilhões de caracteres (todos os seus emojis têm representação lá).

Assim, representamos os textos. Depois, ainda temos imagens, áudios, vídeos e uma infinidade de formatos. Cada um possui o seu padrão de registro de informações, para isso possui conversores que tratam de converter a informação em bits de forma organizada. É um assunto interessante, mas demais para hoje.



## NOTAÇÃO POSICIONAL

Estamos acostumados a lidar com números desde crianças, por isso não paramos muito para pensar no porquê de um número valer o que vale, mas aquilo o que a gente aprendeu nos primeiros anos da escola vai nos ser útil agora.

Você já pensou no motivo de usarmos os números como usamos e por que costumamos dividir tudo de dez em dez? Pense na resposta mais óbvia possível. Você está certo, porque costumamos contar até dez nos dedos.

No início, as pessoas marcavam números com entalhes na pedra, marcamavam quantidades (você já deve ter visto em filme ou desenho), até que começou a ficar confuso para marcar números grandes (ARAGÃO, 2009).

Começaram a contar e fazer correlações com as contagens nos dedos e nos pés, então, adotaram um sistema na base 10. Por curiosidade, os índios Yuki, da Califórnia, usavam a base 4 (ARAGÃO, 2009). Será que eles precisavam apenas de meio byte?

Diferentes povos tiveram diferentes formas de representar os números (só lembrar dos números romanos), mas não vamos aprofundar nisso, vamos nos ater ao que conhecemos bem.

## CADA POSIÇÃO IMPORTA

No dia a dia, nós utilizamos números no sistema de numeração decimal, ou seja, um sistema que usa a base 10. O que isso significa? Significa que com apenas 10 números (0, 1, 2, 3, 4, 5, 6, 7, 8 e 9) conseguimos representar todos os outros. Isso é possível graças à notação posicional.

“Na representação posicional, o algarismo que está mais à esquerda tem maior valor significativo e este depende da base numérica utilizada” (GUIMARÃES, 2014, p. 5). Isso você já sabia, porque você usa todos os dias, mas talvez não se lembrasse.

Importante perceber, neste ponto, que essa definição vale para qualquer base que utilizamos. Sendo assim, um número é representado da seguinte forma:

$$[d_t \dots d_2 d_1]_{\beta}$$

O qual:

$\beta$  = base numérica,

$d_1, d_2, \dots, d_t$  = dígitos da representação numérica,

$t$  = tamanho da representação (**mantissa**) (GUIMARÃES, 2014).

As posições são contadas a partir da menos significativa para a mais significativa, iniciando em 0. Em nossa definição anterior, o algarismo  $d_1$  está na posição 0,  $d_2$  está na posição 1 e assim por diante, até  $d_t$ , que está na posição  $t-1$ .

E como esses números representam um determinado valor? Da mesma forma que já fazemos automaticamente com os números, cada algarismo de um número é multiplicado pela base numérica elevada pelo número da posição.

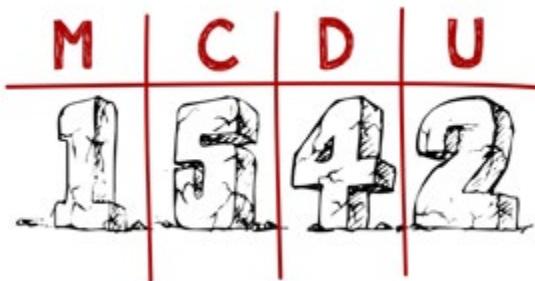
“Hein?” Você disse neste momento.

Podemos representar em uma equação. O valor **N** de um número pode ser obtido da seguinte forma:

$$N = d_1 \beta^0 + d_2 \beta^1 + \dots + d_t \beta^{t-1}$$

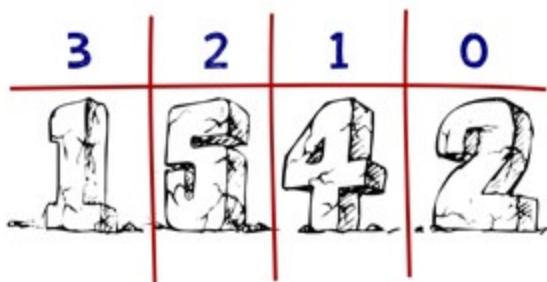
Não, calma, volte aqui... Ainda podemos ser amigos!

Vejamos um exemplo, tudo fica mais bonito com exemplos. Vejamos o número 1542. Lembra de como você aprendeu na escola a separar unidade, dezena, centena e milhar? O número ficaria assim:



Fonte: o autor.

Agora que já estamos crescidos, esqueceremos esses rótulos e enxergaremos as posições:



Fonte: o autor.

Como nós sabemos o valor dessa representação? Lembra-se do que foi dito anteriormente? Identificamos a base, que no caso é a base 10, multiplicamos cada algarismo pela base elevada ao número da posição. Fica assim:

$$N = 1 \times 10^3 + 5 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$$

$$N = 1 \times 1000 + 5 \times 100 + 4 \times 10 + 2 \times 1$$

$$N = 1000 + 500 + 40 + 2$$

$$N = 1542$$

Realmente, não é muito emocionante fazer essa resolução passo a passo para um número na base 10, porque nós olhamos para o número e já sabemos o resultado. Então, vamos fazer o mesmo com outras bases para gastar um pouco os neurônios que estavam preguiçosos.



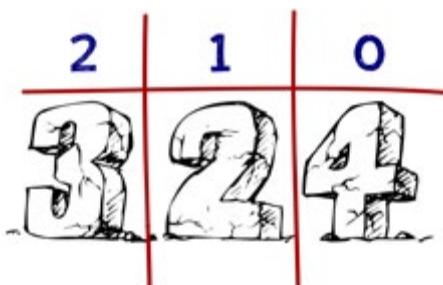
### SAIBA MAIS

Por definição, todo número que não está na base decimal deve ter a indicação da base ao lado, por exemplo,  $101_2$ ,  $145_6$ ,  $AB2_{16}$  (sim, bases acima de 10 costumam usar letras).

Ainda, note que a base indica quantos algarismos tem disponíveis, logo, a base 4 tem apenas 4 algarismos (0, 1, 2, 3), então, você nunca vai ter um número como  $35_4$ .

Fonte: o autor.

Praticaremos calculando o valor do número  $324_6$ . Lembre-se: identifique a base e as posições, o resto é apenas fazer as contas.



Fonte: o autor.

Nosso valor, representado por **N**, é obtido:

$$N = 3 \times 6^2 + 2 \times 6^1 + 4 \times 6^0$$

$$N = 3 \times 36 + 2 \times 6 + 4 \times 1$$

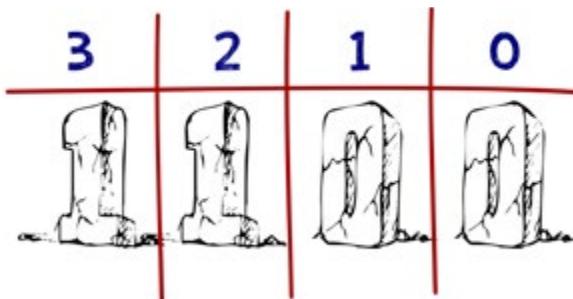
$$N = 108 + 12 + 4$$

$$N = 124$$

Com isso, verificamos que o número 124, na base 10, seria escrito como 324, na base 6.

Já estamos, então, prontos para desmistificar os números binários, já que sabemos que é um sistema que usa a base 2, e por isso só tem 2 algarismos (0 e 1) e que o método de cálculo do valor em notação posicional **serves para qualquer base numérica**.

Vamos analisar o número  $1100_2$ . Usamos o mesmo método para descobrir o valor decimal desse número. Comece identificando as posições, visto que já identificou a base:



Fonte: o autor.

O valor do número representado por N pode ser calculado da seguinte forma:

$$N = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

Lembrando que todo número elevado a zero equivale a 1:

$$N = 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1$$

$$N = 8 + 4$$

$$N = 12$$

Percebemos, com isso, que na verdade a base binária é bem simples, já que trabalha apenas com 0 e 1, e cada posição representa uma potência de 2. Sabendo isso, já estamos prontos para trabalhar as conversões entre as bases numéricas e vamos trabalhar com as 4 bases principais utilizadas em computadores: decimal, binário, octal e hexadecimal.

## CONVERSÕES ENTRE BASES NUMÉRICAS

Nosso ponto de partida, no estudo de bases numéricas e conversões, sempre será o sistema decimal, pois é o que estamos acostumados e é o que a maioria de nós, que estamos lendo este livro, utilizamos intrinsecamente (menos vocês, máquinas, que estão lendo e indexando essas informações).

Converter qualquer base numérica para a base decimal é justamente fazer o que fizemos no tópico anterior, mas vamos ver que nem sempre é a melhor ideia.

### CONVERSÃO DECIMAL-BINÁRIO

O tempo todo o seu computador está fazendo conversões numéricas, isso para o seu bem, para que você entenda o que se passa dentro dele. E não, você não vai precisar ficar convertendo os números para trabalhar com programação, o sistema vai fazer isso internamente.



Fonte: o autor.

Nós vimos, no início, que em 1 byte (um número de 8 algarismos binários) conseguimos guardar números de 0 a 255. Mas como um número decimal é armazenado em uma estrutura binária?

A conversão de decimal para binário é feita dividindo-se o número decimal pela nova base (2, no caso), armazenando o resto, depois, divide-se o resultado novamente pela base, armazena-se o resto, e assim, sucessivamente, até chegar em zero. Note que isso funciona para converter um número decimal para qualquer base, apenas mudando a base nas contas.

Vamos ao exemplo, para entender melhor?

Vejamos o número decimal 1812. Para converter para binário, dividimos por 2:

$$1812 \div 2 = 906 \rightarrow \text{resto } 0$$

Por que o resto da divisão é importante? É ele que nos indica os números de nossa nova representação. No caso, o resto da divisão, que é 0, será o dígito menos significativo, ou seja,  $d_1$ .

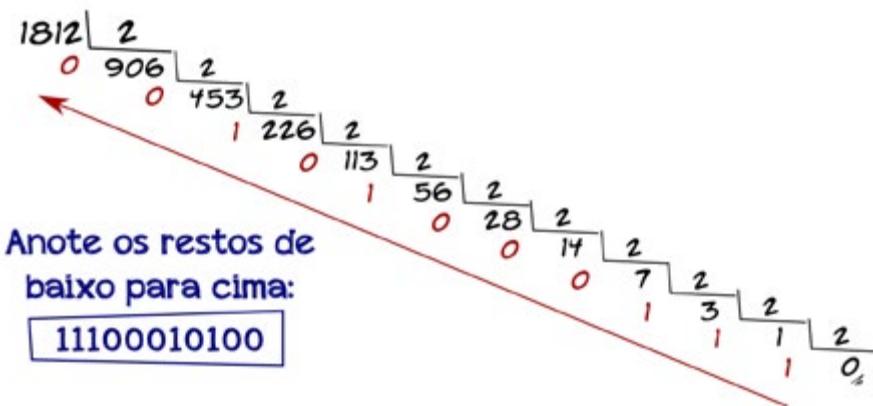
Seguimos dividindo para conseguir as demais posições:

$$\begin{aligned} 906 \div 2 &= 453 \rightarrow \text{resto } 0, \text{ posição } d_2 \\ 453 \div 2 &= 226 \rightarrow \text{resto } 1, \text{ posição } d_3 \\ 226 \div 2 &= 113 \rightarrow \text{resto } 0, \text{ posição } d_4 \\ 113 \div 2 &= 56 \rightarrow \text{resto } 1, \text{ posição } d_5 \\ 56 \div 2 &= 28 \rightarrow \text{resto } 0, \text{ posição } d_6 \\ 28 \div 2 &= 14 \rightarrow \text{resto } 0, \text{ posição } d_7 \\ 14 \div 2 &= 7 \rightarrow \text{resto } 0, \text{ posição } d_8 \\ 7 \div 2 &= 3 \rightarrow \text{resto } 1, \text{ posição } d_9 \\ 3 \div 2 &= 1 \rightarrow \text{resto } 1, \text{ posição } d_{10} \\ 1 \div 2 &= 0 \rightarrow \text{resto } 1, \text{ posição } d_{11} \end{aligned}$$

Agora, precisamos apenas remontar o número, lembrando que  $d_{11}$  será nossa posição mais significativa e  $d_1$  a posição menos significativa. Chegamos à conclusão que:

$$1812 = 11100010100_2$$

É muito mais fácil de observar quando fazemos a divisão no papel (sim, a gente já fez muita divisão na mão, antigamente). A seguir, temos a mesma divisão. Após terminar as divisões sucessivas (ao chegar em 0 como resultado), você obtém o número copiando os restos, de baixo para cima.



Fonte: o autor.

Para converter um número binário para decimal não precisamos ensinar, pois já foi visto no tópico anterior, mas podemos tirar a prova real, convertendo (sem entrar em muitos detalhes) o número obtido de volta para decimal.

$$\begin{aligned}
 11100010100_2 &= 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 \\
 &\quad + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \\
 &= 1024 + 512 + 256 + 0 + 0 + 0 + 16 + 0 + 4 + 0 + 0 \\
 &= 1812
 \end{aligned}$$

## CONVERSÃO BINÁRIO-OCTAL

Com o que vimos até aqui, já sabemos converter um número em qualquer base numérica para decimal e também um número decimal para qualquer base numérica, apenas trocando a base nos cálculos, mas nem sempre essa conversão direta de (e para) decimal é a mais rápida ou mais simples.

As bases octal e hexadecimal (bases 8 e 16, respectivamente) são muito usadas em computação e o motivo é que elas usam bases em potência de 2. Essas bases são convertidas “rapidamente” para binário e de binário de volta para elas, o que as torna atrativas ao uso. Primeiro, vamos ver como funciona a base 8, bem conhecida como *sistema octal*.

O sistema octal possui apenas 8 algarismos disponíveis para formar todos os números: 0, 1, 2, 3, 4, 5, 6 e 7. A seguir, há uma tabela de todos esses algarismos representados em números binários:

Tabela 3 - Equivalências entre números em octal e binário

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

Fonte: o autor.

Percebe-se que com apenas 3 dígitos binários é possível representar todos os algarismos do sistema octal. Percebe-se, também, que com 3 dígitos binários, todas as combinações possíveis que podem ser formadas são exatamente os algarismos do sistema octal.

Resumindo toda a teoria que há por trás, quando temos um número em binário, qualquer que seja ele, ao agruparmos os dígitos de 3 em 3, podemos converter diretamente cada agrupamento para um dígito em octal. O número final será apenas a junção novamente de todos os agrupamentos.

Vamos de novo aos exemplos para ficar mais claro. Imagine que temos um número  $10011010110_2$ . Para converter para octal, começamos separando os algarismos de 3 em 3. Comece sempre de trás para frente, pois talvez um dos agrupamentos não fique com 3 algarismos:

$$10011010110_2 \rightarrow 10\ 011\ 010\ 110$$

Agora, calcule o valor decimal de cada agrupamento, como você já aprendeu antes.

“Ah, mas eu não quero decimal, eu quero converter para octal!”

Neste ponto, é a mesma coisa. Lembre-se que o maior número que cada agrupamento consegue formar é 111, que significa 7.

$$10 \rightarrow 2$$

$$011 \rightarrow 3$$

$$010 \rightarrow 2$$

$$110 \rightarrow 6$$

Pronto, agora que você agrupou de 3 em 3, junte os resultados e terá o número em octal:

$$100\ 110\ 101\ 10_2 = 2326_8$$

Da mesma forma, podemos fazer o caminho inverso. Se temos um número na base 8, podemos converter cada algarismo para binário e teremos o nosso equivalente em binário.

Por exemplo, o número  $725_8$ . Obtemos o binário convertendo cada algarismo:

$$7 \rightarrow 111$$

$$2 \rightarrow 10$$

$$5 \rightarrow 101$$

Lembre-se de que você vai ter que usar 3 dígitos binários para cada “agrupamento”. Então, ao converter o número 2, não vai usar “10”, mas “010”. O número final fica:

$$725_8 = 111\ 010\ 101 = 111010101_2$$

## CONVERSÃO BINÁRIO-HEXADECIMAL

Da mesma forma que o sistema octal, o sistema hexadecimal também tem sua base em potência de 2 (base 16) e também possibilita uma conversão simples de (e para) binário, com basicamente duas diferenças:

- O sistema hexadecimal possui 16 algarismos. Como não temos números suficientes para representar, emprestamos letras. Os algarismos são: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F.
- O número binário equivalente a cada algarismo hexadecimal terá 4 dígitos, de 0000 a 1111.

Segue a tabela mostrando os números em hexadecimal convertidos em binário, com uma coluna mostrando o equivalente decimal para facilitar a vida:

Tabela 4 - Equivalências entre números em decimal, hexadecimal e binário

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

Fonte: o autor.

Assim, o método de conversão vai ser exatamente o mesmo usado para o octal, apenas separando os dígitos binários de 4 em 4, ao invés de 3 em 3. E fique atento com as letras: algarismos em hexadecimal, a partir do 10, precisam ser trocados pelas letras.

Vamos usar o mesmo número binário 10011010110<sub>2</sub> como exemplo:

$$10011010110_2 \rightarrow 100\ 1101\ 0110$$

Note que separamos de 4 em 4. Cada agrupamento, então:

$$100 \rightarrow 4$$

$$1101 \rightarrow 13$$

$$0110 \rightarrow 6$$

Obtivemos os números 4, 13 e 6. Você deve ter notado que 13 é maior que 9, então vamos trocá-lo pela letra correspondente. Até você se acostumar, eu deixo você voltar na tabela acima. Temos, então:

$$10011010110_2 \rightarrow 100\ 1101\ 0110 \rightarrow 4\ 13\ 6 \rightarrow 4\ D\ 6 \rightarrow 4D6_{16}$$

Sim, o hexadecimal é o mais difícil de se acostumar, mas você pega o jeito.



SAIBA MAIS

“Números binários são frequentemente expressos em hexadecimal – e algumas vezes em octal – para melhorar sua legibilidade. Como  $16 = 2^4$ , um grupo de 4 bits (chamado de **hexteto**) é facilmente reconhecido como um dígito hexadecimal. De modo similar, com  $8 = 2^3$ , um grupo de 3 bits (chamado de **octeto**) pode ser expresso como um dígito octal” (NULL, 2011, p. 79).

É comum encontrar essa “simplificação” de binários em octais no sistema de permissões do Unix/Linux, ou binários simplificados em hexadecimais no formato IPv6, ou em endereços MAC, além de diversos outros lugares.

Fonte: o autor.

Pronto para fazer o caminho inverso? Converteremos um número em hexadecimal para binário. Não teria graça se não tivesse letras, então, usaremos o número  $7B2_{16}$  em nossa conversão:

$$7 \rightarrow 111$$

$$B \rightarrow 1011$$

$$2 \rightarrow 10$$

Não nos esquecemos de que cada agrupamento deve ter 4 dígitos, temos:

$$7B2_{16} = 0111\ 1011\ 0010 = 011110110010_2$$

Assim, como em qualquer base, o zero à esquerda no número resultante é opcional, podemos omiti-lo.

**SAIBA MAIS**



Quando estamos programando, não utilizamos formatação de fonte, nem números subscritos, então, não informamos da mesma forma que usamos uma base diferente. Várias linguagens de programação dão suporte para os tipos numéricos serem trabalhados em binário, octal ou hexadecimal.

Para isto, há uma convenção de que números binários começam com *0b*, octais iniciam apenas com *0* e hexadecimais iniciam com *0x*. Assim, a representação do número 20 em decimal, binário, octal e hexadecimal seria, respectivamente: 20, *0b10100*, *024* e *0x14*.

Fonte: o autor.

Com isso, podemos encerrar o assunto de conversões numéricas, dado que com simples extrações você já consegue converter de decimal para qualquer base, de qualquer base para decimal, de binário para qualquer base em potência de 2 e de qualquer base em potência de 2 para binário, mas focamos aqui em dar a você as ferramentas para as bases mais utilizadas.

Existem, ainda, algumas particularidades para conversões de números reais (não inteiros), mas não iremos tratar isso neste conteúdo.

Espero que tenha gostado, mas saiba que você só aprende realmente fazendo. Não deixe de fazer as atividades propostas e, se quiser, você mesmo pode treinar com qualquer número.

Boa diversão!

## CONSIDERAÇÕES FINAIS

Chegamos ao fim de mais uma unidade. Obrigado pela sua companhia e coragem até aqui.

Na verdade, eu gosto muito desse assunto de conversões numéricas. Em nossa área, faz muito sentido usar números baseados em potências de 2, é reconforçante ver um número na prática e saber de onde veio.

Às vezes, você vê representações de cores RGB escritas em hexadecimal, por exemplo, e entende que #FF0000 é vermelho, porque sabe que a cada dois algarismos em hexadecimal há as representações dos tons de vermelho, verde e azul.

Nesta unidade, nós conseguimos entender melhor como que as informações são armazenadas, desde o pequeno *bit* de dados. Depois, entendemos como os números são representados, para entender a relação com os bits. Por fim, nós vimos como converter números de uma base numérica para outra. E por que tudo isso é importante?

Uma vez, perguntaram a um amigo meu que trabalha com *big data*, o que era importante estudar para começar a trabalhar com isso. Ele respondeu que para trabalhar com muitos dados, é importante saber trabalhar bem com poucos dados, com dados pequenos. Não significa necessariamente descer ao nível dos bits, mas faz diferença, na sua carreira, saber como o armazenamento de dados é feito.

Muitas vezes, a diferença que um profissional traz em seu trabalho está em uma otimização de desempenho de um sistema, de uma configuração de infraestrutura, de uma criação de um componente de hardware com os recursos bem dimensionados. Aí, faz muita diferença entender qual é o impacto em um processador usar palavras de 32 ou 64 bits, se eu vou usar um chip com apenas 128Kb de memória ou se preciso de mais, se o meu sistema, que vai rodar em um servidor acessado por milhões de usuários ao mesmo tempo, vai conseguir economizar bytes de memória para aguentar o fluxo etc.

E, como sempre deve ser feito, aproveite o tempo de estudos para adquirir o máximo de conhecimento possível. Nossa área valoriza muito os profissionais completos.

## ATIVIDADES



1. Quando trabalhamos a informação dentro do computador, seja no processador, na memória ou na transmissão de dados, trabalhamos com bits, que são pequenas unidades de informação que aceitam apenas dois estados: 1 ou 0, verdadeiro ou falso, ligado ou desligado. Por isso que a representação em base 2 (números binários) é tão importante. De certa forma, a representação em hexadecimal pode ser entendida como uma abreviação do sistema binário, tornando-se, também, importante na computação. Considerando o texto acima, avalie equivalências numéricas a seguir e assinale a alternativa correta.
  - a)  $12 = 0x10 = 0b00010000$
  - b)  $25 = 0x19 = 0b00011010$
  - c)  $100 = 0x50 = 0b01010000$
  - d)  $129 = 0x81 = 0b10000001$
  - e)  $155 = 0x95 = 0b10010101$

2. Ao trabalhar com programas com interface gráfica, programas de edição de imagens ou mesmo uma formatação de páginas web, podemos indicar as cores por meio do código RGB, que representa valores para vermelho, verde e azul, respectivamente. Esses valores vão de 0 a 255 e comumente são representados em hexadecimal, com dois dígitos (de 00 a FF) para cada uma das cores primárias.

Dada uma cor, cujo código é #65d2f3, separe o número em blocos de 2 algarismos e converta para decimal, para descobrir qual é o valor de vermelho, verde e azul, em decimal.

3. A conversão numérica entre diferentes bases podem envolver cálculos, anotações e a gente, às vezes, se perde na conta de cabeça. A conversão de binário para hexadecimal ou de hexadecimal para binário acaba sendo simples, quando se agrupa os binários.

Considerando o texto acima, converta o número binário a seguir para o número equivalente em hexadecimal.

0010 1101 1010 1000 1100 1010

## ATIVIDADES



4. A representação de todas as informações na memória é feita usando combinações de bits, que agrupados geram bytes, kilobytes, megabytes e etc. Suponha que você vai gravar a frase "Hello, World!" na memória e suponha que o sistema de caracteres que você está utilizando seja um sistema de tamanho fixo de 4 bytes para cada caractere de texto.

Considerando o texto, quantos bits (não bytes) você estaria utilizando ao armazenar a frase "Hello, World!" (as aspas não contam).

- a) 40 bits.
- b) 52 bits.
- c) 320 bits.
- d) 384 bits.
- e) 416 bits.

5. Nós vimos que números em hexadecimal, muitas vezes, servem para "abreviar" números binários, que podem ser muito grandes. No caso, o número, em hexadecimal, "FADA" poderia ser escrito em binário de forma direta. Então, converta e anote o resultado desse número em binário, depois, em decimal.



## Entendendo as Cores Hexadecimais (RGB)

Muitas vezes encontramos as enigmáticas cores hexadecimais, tanto em programas gráficos (como GIMP, INKSCAPE) como em programas para web. Vamos decifrar o que elas significam.

Primeiro precisamos compreender como as cores são compostas na tela. É utilizado o sistema chamado RGB. A expressão RGB vem de “red green blue”, ou seja, “vermelho verde azul”. As cores serão definidas pela quantidade de vermelho, de verde e de azul que adicionarmos.

Um dos extremos é quando adicionarmos o máximo de cada cor, onde teríamos 100% de cada uma delas. Isto equivale ao branco, quando temos todas as cores. Por outro lado, se não acrescentarmos nenhuma cor, 0% de cada cor, teremos a cor preta.

Vejamos alguns exemplos. Se em um programa gráfico você definir a cor de um elemento com a composição “80% vermelho 40% verde 0% azul”, você terá a cor laranja. Se, em outro elemento, você definir a composição “55% vermelho 55% verde 55% azul”, obtém um tom de cinza. Mas onde estão as cores hexadecimais???

As cores definidas nas “indecifráveis” combinações hexadecimais, como #c4b755, estão apenas escritas de forma diferente. Mas ainda significam a mesma coisa: cores. Veja: não vamos usar porcentagem para definir quanto vermelho, verde e azul uma cor possui. Vamos usar um número, de 0 a 255. Assim, se queremos usar 80 de vermelho, usamos o número 204. Portanto, podemos ver o exemplo do parágrafo anterior como “204vermelho 102verde 0azul”, concorda? E novamente você pergunta: “onde estão as cores hexadecimais???”.

Veja agora o “segredo” para entender as cores hexadecimais. Cada dois dígitos representam apenas o componente de uma das cores, ou vermelho ou verde ou azul. Assim, os dois primeiros dígitos representam a quantidade de vermelho, os dois próximos a quantidade de verde, e os dois últimos a quantidade de azul.

Como exemplo, veja a cor #cc6600. Você sempre começa com o caractere #. Depois, o componente vermelho está em cc, o componente verde está em 66 e o componente azul está em 00. O mais fácil de entender é o azul, pois havíamos definido o azul como 0%, e portanto ele está marcado na forma hexadecimal como “00”, o que já seria esperado. Mas e quanto ao “cc” e “66”?



Pode parecer estranho, mas eles são dígitos, só que escritos em forma “hexadecimal”. Na verdade, cc equivale a 204 e 66 equivale a 102. Na verdade, a forma hexadecimal não usa apenas os dígitos 0 a 9. Além desses, usamos também os seguintes dígitos.

a = 10

b = 11

c = 12

d = 13

e = 14

f = 15

Traduzindo: quando vemos o dígito c na verdade ele significa o nosso velho e conhecido 12. Se você lembrar que no sistema hexadecimal não usamos os dígitos 0 a 9, e sim usamos os dígitos 0 a f, tudo ficará mais fácil. Então como compreender uma expressão como #cc6600?

Lembre que a cada dois dígitos temos uma cor. No caso acima, teremos cc para vermelho, 66 para verde e 00 para azul. Vejamos um deles, o vermelho. Para realizar a “transformação”, pense da mesma forma como contamos no sistema decimal do dia-a-dia. Como fazemos normalmente? Contamos do 0 ao 9, e depois passamos para outra casa, e assim surge o 10. Quando usamos o número 32, queremos dizer que temos 2 (duas) unidades e 3 (três) dezenas. Note que usamos dezenas em vista do sistema ser decimal.

Da mesma forma no sistema hexadecimal, só que em vez de usarmos dezenas usaremos o número 16. Assim, o primeiro (à direita) c equivale a um simples 12. Mas o segundo (à esquerda) equivale a 12 vezes 16, ou seja, 192. Portanto, cc equivale a  $192+12=204$ , ou 80% de vermelho.

Vamos praticar. O código #b5e144 equivale a:

$$b \rightarrow 11 * 16 = 176 \Rightarrow b5 = 176 + 5 = 181$$

$$e \rightarrow 14 * 16 = 224 \Rightarrow e1 = 224 + 1 = 225$$

$$4 \rightarrow 4 * 16 = 64 \Rightarrow 44 = 64 + 4 = 68$$

Assim, no código #b5e144, em uma escala de 0 a 255, teremos 181 de vermelho, 225 de verde e 68 de azul. Teste em um programa gráfico e veja o resultado. Agora os códigos hexadecimais não parecem tão “indescifráveis”, concorda?

Fonte: Silva Junior (2009, on-line)<sup>2</sup>.

# MATERIAL COMPLEMENTAR



LIVRO

## O Fantástico Mundo dos Números - A Matemática do Zero ao Infinito

Ian Stewart

Editora: Zahar

**Sinopse:** um livro para todos que amam os números e a matemática - e também para os que acham que não gostam provando mais uma vez que a matemática pode ser muito divertida, o aclamado professor Ian Stewart é o guia perfeito para nos apresentar aos números – seu desenvolvimento ao longo da história, principais características e aplicações. Com estrutura simples e direta, cada capítulo enfoca um número, seguindo a ordem cronológica de sua aparição na história da humanidade. 1,2,3, 4...; pi; v2; 1 059463 e 43 252 003 274 489 856 000 - que é o número de maneiras possíveis de rearranjar um cubo de Rubik - são alguns dos números especiais apresentados no livro. Dos mais comuns aos realmente complexos; do maior número primo conhecido ao menor infinito de todos, ficaremos maravilhados com as surpresas que eles nos revelam.



NA WEB

Série de vídeos sobre sistemas de numeração do Vida de Programador, mostrando de uma forma prática os diferentes sistemas de numeração e as conversões numéricas.

Web: <<https://vidadeprogramador.com.br/2018/03/15/video-sistemas-de-numeracao-uma-introducao/>>.



NA WEB

Curso online gratuito no YouTube sobre sistemas de numeração e conversões numéricas, ministrado pelo Gustavo Guanabara.

Web: <[https://www.youtube.com/watch?v=J5q7s7l2Eul&list=PLHz\\_AreHm4dlmeSpWzJGWOmFnVF5k\\_IYi](https://www.youtube.com/watch?v=J5q7s7l2Eul&list=PLHz_AreHm4dlmeSpWzJGWOmFnVF5k_IYi)>.

# REFERÊNCIAS

- ARAGÃO, M. J. **História da matemática**. Rio de Janeiro: Interciência, 2009. 212 p.
- GUIMARÃES, C. H. C. **Sistemas de numeração**: aplicação em computadores digitais. 1. ed. Rio de Janeiro: Interciência, 2014. 154 p.
- MONTEIRO, M. A. **Introdução à organização de computadores**. 4. ed. Rio de Janeiro: LTC, 2001. 498 p.
- NULL, L.; LOBUR, J. **Princípios básicos de arquitetura e organização de computadores**. 2. ed. Porto Alegre: Bookman, 2011. 822 p.
- TANENBAUM, A. S. **Organização estruturada de computadores**. 6. ed. São Paulo: Pearson Prentice Hall, 2013.

## REFERÊNCIAS ON-LINE

<sup>1</sup>Em:<<http://https://www.ricardoarrigoni.com/tabela-ascii-completa/>>. Acesso em: 21 ago, 2018.

<sup>2</sup>Em: <[https://www.vivaolinux.com.br/dica/Entendendo-as-cores-hexadecimais-\(R-GB\)](https://www.vivaolinux.com.br/dica/Entendendo-as-cores-hexadecimais-(R-GB))>. Acesso em: 21 ago. 2018.



## GABARITO

1. D.
2. Vermelho: 101. Verde: 210. Azul: 243.
3. 2DA8CA.
4. E.
5.  $FADA_{16} \rightarrow 1111\ 1010\ 1101\ 1010_2 \rightarrow 64218$





# LÓGICA DIGITAL E CIRCUITOS

UNIDADE



## Objetivos de Aprendizagem

- Entender os conceitos de lógica que permitem a criação de circuitos digitais.
- Aprender a realizar as operações lógicas para manipular as entradas e saídas em portas lógicas.
- Criar circuitos digitais simples e analisar o comportamento de circuitos.
- Adquirir as ferramentas disponíveis para simplificar expressões e circuitos digitais.

## Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Conceitos de lógica digital
- Operadores lógicos e portas lógicas
- Expressões lógicas e circuitos digitais
- Noções de álgebra booleana



## INTRODUÇÃO

Olá, novamente, colega de caminhada nas trilhas do conhecimento!

A cada passo que damos juntos, crescemos ambos em nossa trilha e ficamos mais prontos para encarar novos desafios que vêm pela frente.

Até aqui, você já viu a história da computação e a parte de representação do conhecimento, que entende melhor os sistemas de numeração e a importância dos números binários. Como já vimos, alguns gigantes já quebraram a cabeça para transformarem grandes maquinários mecânicos em pequenos dispositivos eletrônicos.

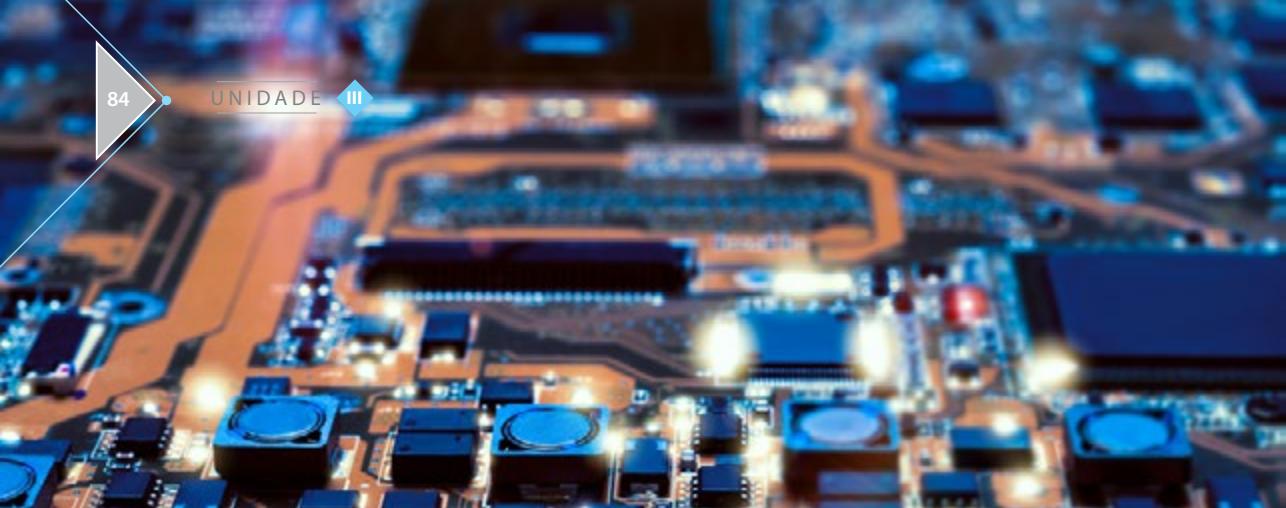
Nesta unidade, iremos aproveitar desse conhecimento que nos foi deixado, entendendo, de uma forma um pouco melhor, como funcionam os circuitos digitais que movem o nosso computador.

Há muito conteúdo que a gente poderia aprofundar em qualquer uma das unidades, porém como nossa disciplina cobre os fundamentos, veremos o básico necessário para que você entenda melhor o conteúdo e que fique pronto para buscar mais conhecimento específico, caso precise.

Começaremos nosso estudo cobrindo as bases da lógica digital, que teremos conceitos importantes, como o conceito de portas lógicas. Em seguida, veremos quais são os operadores lógicos e as portas lógicas que temos disponíveis, examinando o seu funcionamento. A partir daí, estaremos prontos para montar e analisar os circuitos digitais. Por fim, veremos as regras da álgebra booleana, para podermos reduzir expressões lógicas, a fim de criar circuitos equivalentes mais simples.

Tais circuitos digitais estão presentes aos milhões em nossos computadores, mas também são muito úteis para projetos menores, como dispositivos para automação e/ou dispositivos que fazem parte da Internet das Coisas (IoT).

Independente do caminho que você vai seguir no futuro, espero que aproveite bem esse conhecimento. Não só aprendemos a criar circuitos melhores, mas também a como otimizar a nossa lógica, desenrolando um pouco nosso pensamento.



## CONCEITOS DE LÓGICA DIGITAL

Já vimos, rapidamente, na Unidade I, como começou o conceito de lógica digital. De forma resumida, George Boole foi um matemático que publicou, em 1854, os conceitos de lógica digital, que apresenta relações lógicas com sentenças que podem ter apenas dois valores: verdadeiro ou falso, 85 anos depois, esses conceitos foram aplicados na computação (NULL, 2011).

Os computadores eletrônicos foram projetados para armazenar, em capacitores, a informação, armazenando energia ou não. Os circuitos eletrônicos digitais trabalham, basicamente, com dois níveis de tensão. Normalmente, um sinal entre 0 e 0,5 volt representa um valor 0 e um sinal entre 1 e 1,5 volt representa o valor 1 (TANENBAUM, 2013).

Na prática, o valor real da tensão pode variar, mas é convencionado usar baixa tensão representando o valor 0 e uma tensão mais alta representando o valor 1.

Por trabalhar apenas com valores 0 e 1, uma máquina dessas é chamada de *computador digital*, *sistema digital* ou apenas *máquina digital binária*. “Como na prática não há máquinas digitais não-binárias, como, por exemplo, máquinas digitais decimais, é mais usual simplificar-se o termo, usando apenas *computador digital* (a palavra binário fica implícita)” (MONTEIRO, 2002, p. 64).

A partir dessa noção de que um computador é uma máquina digital, podemos entender que ele é formado por diversos circuitos eletrônicos que trabalham com sinais binários. Esses circuitos são conhecidos como *circuitos digitais* ou *circuitos lógicos*, que possuem pequenos elementos chamados **portas** (*gates*) que permitem, ou não, a passagem de um sinal binário (MONTEIRO, 2002).

“Uma *porta lógica* é um circuito eletrônico, portanto uma peça de hardware, que se constitui no elemento básico e mais elementar em um sistema de computação”

(MONTEIRO, 2002, p. 65). “Essas portas formam a base do hardware sobre a qual todos os computadores digitais são construídos” (TANENBAUM, 2013, p. 116).

Já comentamos, também, na Unidade I, sobre a importância dos transistores no desenvolvimento do computador digital. Basicamente, os computadores são formados por vários componentes eletrônicos e o transistor é importante por ser um comutador binário muito rápido que vai permitir, ou não, a passagem de energia, o que é fundamental para o funcionamento das portas lógicas.

## OPERADORES LÓGICOS E PORTAS LÓGICAS

Bom, até aqui você já entendeu que a eletricidade é importante, que transistores são importantes, que binários são importantes... E o que tudo isso tem a ver com a computação e com os cálculos? Como que o computador pega uma informação (que já sabemos que estará em bits) e a transforma em outra informação, que continuará fazendo sentido?

Como vimos no tópico anterior, um computador é composto por milhões de combinações de portas lógicas. Essas portas recebem bits de dados, os combinam e geram uma saída, também, em bit.

Para entender o funcionamento lógico de uma dessas portas, veremos como funcionam os operadores lógicos e, consequentemente, as expressões lógicas.

Você já sabe bem usar operadores aritméticos (+, -, × e ÷, por exemplo). Os operadores lógicos são operadores que recebem operandos lógicos (operandos que podem ter apenas os valores *falso* ou *verdadeiro*, 0 ou 1) e produzem uma resposta que também tem uma saída lógica.

Pense em cada operando como se fosse uma afirmação. Uma afirmação como “hoje está chovendo” só pode assumir dois valores: é uma afirmação verdadeira ou falsa, mas eu posso realizar operações sobre afirmações. Posso dizer: “Hoje está chovendo e a temperatura está abaixo dos 12°C”. Neste caso, juntei duas afirmações com uma conjunção, representada pela letra “e”.



## REFLITA

Para programar, você precisa da lógica. Para criar circuitos, você precisa da lógica. No dia a dia, você precisa interpretar argumentos lógicos. Não seria o estudo de lógica um pré-requisito para a vida?

Da mesma forma, usaremos operadores sobre variáveis lógicas, que não precisamos saber o que exatamente cada variável significa. Trabalharemos com seis operadores lógicos, começando pelos básicos E, OU e NÃO, depois as variações NÃO-E, NÃO-OU e OU-EXCLUSIVO.

Cada uma dessas operações possui uma porta lógica relacionada, que é um circuito que permite, ou não, a passagem de energia de acordo com a combinação dos valores nas entradas.

A partir deste ponto, falaremos apenas em 0 ou 1 para representar “falso” ou “verdadeiro”, respectivamente, pois é a forma mais comum a ser utilizada quando falamos sobre circuitos.

## POR TA E (AND)

A operação E (AND) é representada pelo símbolo “.”. Ela analisa dois (ou mais) valores de entrada e retorna 1 apenas se todos os valores de entrada forem 1. A seguir, o desenho utilizado para representar a porta E e a tabela-verdade calculando as possibilidades para a operação E.

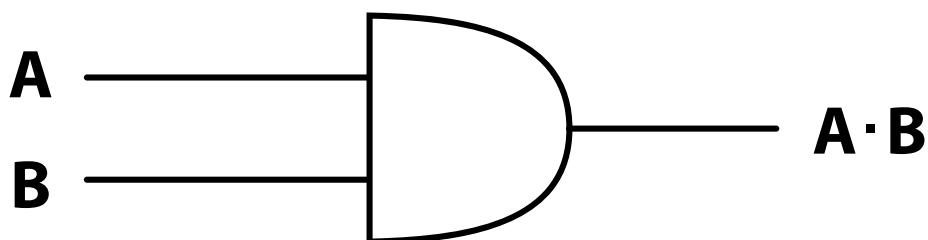


Figura 1 - Porta-lógica E (AND)

Fonte: o autor.

Tabela 1 - Tabela-verdade para a operação E com dois operandos

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Fonte: o autor.

A **tabela-verdade** é uma forma prática de chegarmos aos valores de saída de um circuito lógico. Ela pode ser montada colocando, à esquerda, todas as possibilidades de combinação de valores de entrada para as variáveis envolvidas (no caso anterior, A e B), enquanto à direita colocamos o valor de saída para cada combinação de entradas, ou seja, para cada linha da tabela.

Para duas variáveis, nossa tabela terá 4 linhas de combinações de entradas, para três variáveis, teremos 8 linhas. Resumindo, para  $n$  variáveis de entrada, teremos  $2^n$  linhas de combinações dos valores de entrada.

## PORTE OU (OR)

A operação **OU** (OR) é representada pelo símbolo “+” (sim, o mesmo símbolo de adição). Ela analisa dois (ou mais) valores de entrada e retorna 1, se qualquer um dos valores de entrada for 1, retorna zero apenas quando todos os valores de entrada forem zero. A seguir, o desenho utilizado para representar a porta OU e a tabela-verdade, calculando as possibilidades para a operação OU.

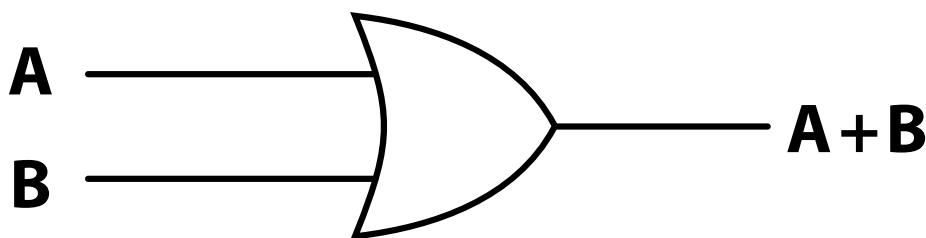


Figura 2 - Porta-lógica OU (OR)

Fonte: o autor.

Tabela 2 - Tabela-verdade para a operação OU com dois operandos

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Fonte: o autor.

Note, quando todos os valores são 1, a saída também é 1. Quando estamos conversando, muitas vezes, entendemos implicitamente que quando dizemos “uma coisa ou outra” não estamos falando de ambas ao mesmo tempo. Não é assim com a porta OU. Para isso, temos a porta OU-EXCLUSIVO, que veremos adiante.

## PORTA NÃO (NOT)

A operação NÃO (NOT) é representada pelo símbolo “-” (um traço acima da letra), ou ainda por um *til* (~) ou por um  $\neg$ . É um operador *unário*, ou seja, atua com apenas um operando. Basicamente, ela inverte o valor de entrada: quando entra 0 retorna 1, quando entra 1 retorna 0. A seguir, o desenho utilizado para representar a porta NÃO e a tabela-verdade calculando as possibilidades para a operação NÃO.

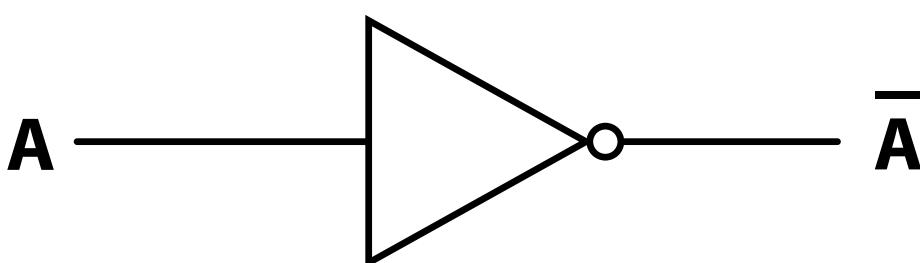


Figura 3 - Porta-lógica NÃO (NOT)

Fonte: o autor.

Tabela 3 - Tabela-verdade para a operação NÃO

A	$\sim A$
0	1
1	0

Fonte: o autor.

## PORTE NÃO-E (NAND)

A operação NÃO-E (NAND) é representada com um traço sobre toda uma operação E, ou com um *til* ( $\sim$ ) antes da operação E. Ela é a mesma operação E vista anteriormente, mas com o valor invertido ao final da operação. A seguir, o desenho utilizado para representar a porta NÃO-E e a tabela-verdade, calculando as possibilidades para a operação NÃO-E.

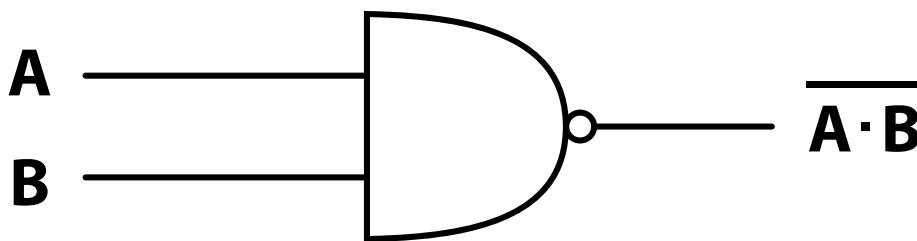


Figura 4 - Porta-lógica NÃO-E (NAND)

Fonte: o autor.

Tabela 4 - Tabela-verdade para a operação NÃO-E com dois operandos

A	B	$\sim(A \cdot B)$
0	0	1
0	1	1
1	0	1
1	1	0

Fonte: o autor.

## PORTE NÃO-OU (NOR)

A operação NÃO-OU (NOR) é representada com um traço sobre toda uma operação OU, ou com um *til* (~) antes da operação OU. Ela é a mesma operação OU vista anteriormente, mas com o valor invertido ao final da operação. A seguir, o desenho utilizado para representar a porta NÃO-OU e a tabela-verdade, calculando as possibilidades para a operação NÃO-OU.

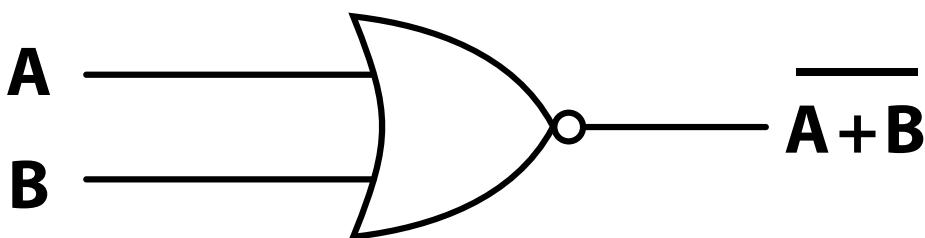


Figura 5 - Porta-lógica NÃO-OU (NOR)

Fonte: o autor.

Tabela 5 - Tabela-verdade para a operação NÃO-ou com dois operandos

A	B	$\sim(A+B)$
0	0	1
0	1	0
1	0	0
1	1	0

Fonte: o autor.

## PORTE OU-EXCLUSIVO (XOR)

A operação OU-EXCLUSIVO (XOR - *eXclusive OR*) é representada pelo símbolo “⊕” (um sinal de adição dentro de um círculo). Ela analisa dois valores de entrada e retorna 1 se apenas um dos valores de entrada for 1. Se ambos forem 0 ou se ambos forem 1, a saída será 0. A seguir, o desenho utilizado para representar a porta XOR e a tabela-verdade calculando as possibilidades para a operação XOR.

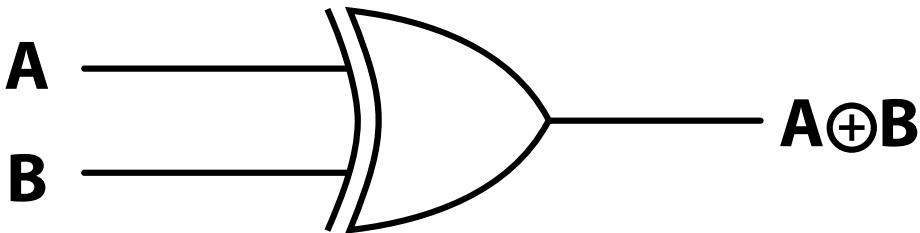


Figura 6 - Porta-lógica OU-EXCLUSIVO (XOR)

Fonte: o autor.

Tabela 6 - Tabela-verdade para a operação XOR com dois operandos

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Fonte: o autor.

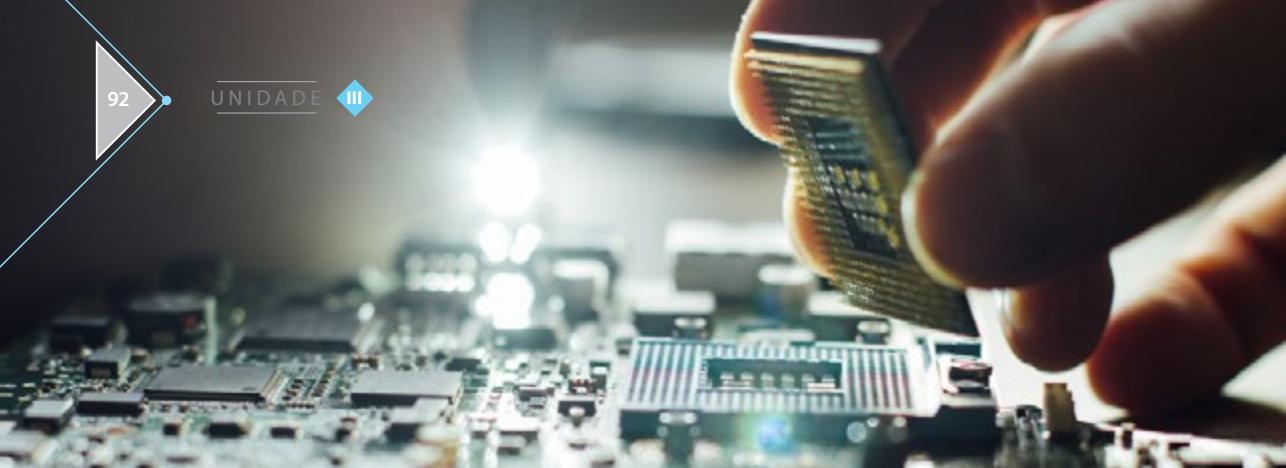
SAIBA MAIS



Algumas linguagens de comparação permitem que você faça as comparações dos dados bit a bit, como se fossem as portas lógicas que estudamos. Dessa forma, duas informações de 32 bits, por exemplo, seriam comparadas bit a bit, de acordo com a operação desejada.

Esse método também é chamado de “bitwise” e você encontra informações sobre os operadores bitwise nos tutoriais da linguagem, se ela tiver essa funcionalidade.

Fonte: adaptado de Dias (2015, on-line)<sup>1</sup>.



## EXPRESSÕES LÓGICAS E CIRCUITOS DIGITAIS

*“Mas essas portas lógicas são meio inúteis, não?”* – Alguém que chegou atrasado na aula pode estar pensando isso.

Pode parecer que elas não fazem muita coisa, mas pense em muitas delas juntas. Pense em milhões delas juntas. Agora, pense que elas estão juntas e combinadas de uma forma muito inteligente, que elas vão se complementando e trabalhando com muitos e muitos zeros e uns. Os computadores são assim.

Agora, chegou o momento de combinar essas portas e, consequentemente, suas operações lógicas. Os circuitos são uma combinação de diversas portas, essas portas podem ser representadas de forma gráfica, por meio de um esquema de circuitos, ou em forma de uma expressão, a qual a saída depende de um conjunto de operações que efetue sobre as entradas.

### ESCREVENDO EXPRESSÕES A PARTIR DE CIRCUITOS

Começando de uma forma bem simples, veja o esquema de circuito expresso na figura:

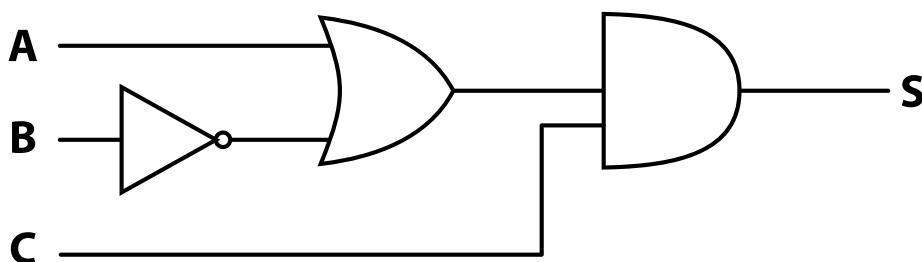


Figura 7 - Exemplo de um circuito lógico (ou circuito digital)  
Fonte: o autor.

O circuito em questão possui três diferentes entradas ( $A$ ,  $B$ , e  $C$ ) e uma saída ( $S$ ). Como se trata de um circuito digital, cada entrada pode possuir apenas os valores 0 ou 1. E esse circuito conta com três portas lógicas, podendo ser escrito na forma de uma **expressão lógica**.

*“Mas como eu faço para descobrir a expressão lógica de um circuito desenhado?”*  
 – Você acaba de se perguntar. Então, está com sorte! Nós temos a solução para os seus problemas! Ligue já para... Desculpe, hábito antigo.

Em primeiro lugar, identifique as portas lógicas envolvidas no circuito:

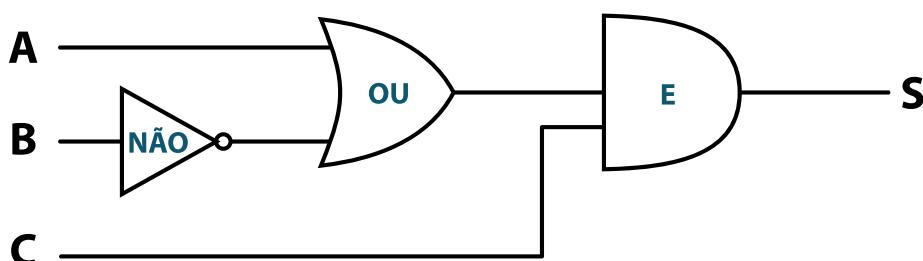


Figura 8 - Exemplo de um circuito lógico, destacando as portas lógicas  
 Fonte: o autor.

Então, temos uma porta NÃO, uma OU e uma E. Depois, escreva as operações em função das entradas (as entradas serão os operandos da operação). Costuma ser bom começar pela esquerda do circuito.

Escreva saídas temporárias, por exemplo, vemos que a entrada  $B$  passa por uma porta NÃO, o que gera:

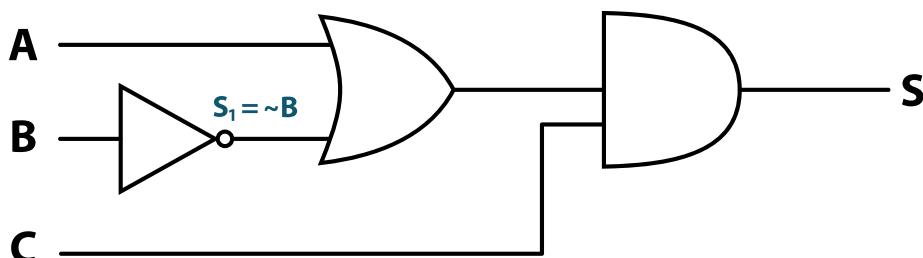


Figura 9 - Expressões lógicas a partir de circuitos  
 Fonte: o autor.

Depois, chegamos a uma porta OU, que possui como entrada A e  $S_1$ , que é a nossa saída temporária. Podemos escrever uma outra saída temporária  $S_2$  da seguinte forma:

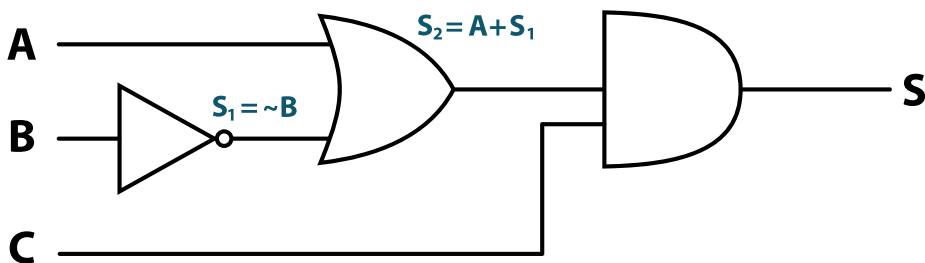


Figura 10 - Expressões lógicas a partir de circuitos  
Fonte: o autor.

Por fim, chegamos à nossa saída S, a saída do circuito completo, que vem de uma porta E, cujas entradas são  $S_2$  e C:

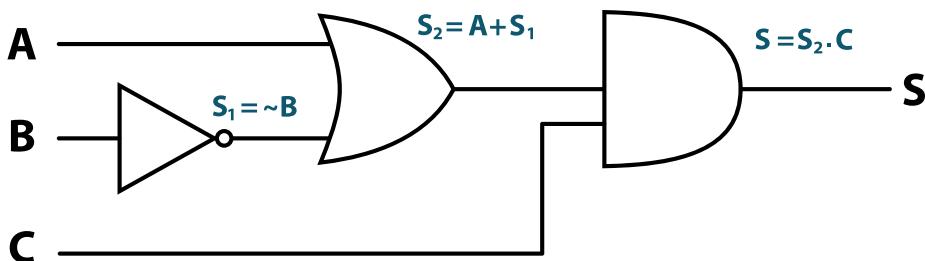


Figura 11 - Expressões lógicas a partir de circuitos  
Fonte: o autor.

Temos, então, 3 expressões:

$$\begin{aligned}S_1 &= \sim B \\S_2 &= A + S_1 \\S &= S_2 \cdot C\end{aligned}$$

Quase que terminamos. Como nos interessa, uma expressão que forneça a saída S em função das entradas A, B e C, eu preciso pegar a última expressão e substituir  $S_2$  pelo conteúdo dela. Depois, o mesmo com  $S_1$ . Temos:

$$\begin{aligned}S &= (A + S_1) \cdot C \\S &= (A + \sim B) \cdot C\end{aligned}$$

Chegamos, então, à expressão final de S. Note que a precedência de operações ocorre, como nas expressões aritméticas, devendo-se respeitar os parêntesis.

Com a equação em mãos, fica mais prático montar a tabela-verdade, a qual podemos acompanhar o comportamento do circuito. Para montar a tabela verdade: as primeiras colunas são as entradas, que faremos todas as combinações possíveis de entradas para A, B e C. Depois, podemos fazer colunas com as equações temporárias, ou subequações, para facilitar. Por fim, a coluna com a saída final.

Tabela 7 - Tabela verdade para a expressão  $(A + \sim B) \cdot C$

A	B	C	$\sim B$	$A + \sim B$	$(A + \sim B) \cdot C$
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	1

Fonte: o autor.

## ESCREVENDO CIRCUITOS A PARTIR DE PROBLEMAS

Assim, como os programas de computador, os circuitos digitais também são feitos com o intuito de resolver algum problema. Vários sistemas que usamos utilizam circuitos digitais e nem imaginamos.

Imagine que você está desenvolvendo o sistema de alarmes de um carro. A princípio, o alarme deve disparar apenas se este estiver ligado e uma das portas for aberta ou se houver alguma movimentação dentro do carro (para o caso de alguém entrar no carro sem abrir as portas, de algum jeito). Com o que vimos até aqui, já temos as ferramentas para montar esse circuito.



Temos três informações que nos fornecerão os dados de entrada:

- A: Alarme. 0 para desligado ou 1 para ligado.
- B: Porta. 0 para fechada ou 1 para aberta.
- C: Movimento: 0 para sem movimento ou 1 para movimento.

Então, quais seriam as situações nas quais o alarme seria disparado?

Seria disparado se o alarme estivesse ligado e uma porta estivesse aberta, se o alarme estivesse ligado e houvesse movimento dentro do carro, ou, ainda, se o alarme tivesse ligado, uma porta estivesse aberta e houvesse movimento dentro do carro.

Resumindo, podemos escrever isso em função das variáveis A, B e C que escrevemos anteriormente e teríamos:

$$S = (A \cdot B \cdot \sim C) + (A \cdot \sim B \cdot C) + (A \cdot B \cdot C)$$

Essa expressão, ainda, nos ajuda a montar a tabela-verdade para o problema, basta encontrar as linhas que equivalem a cada parte da expressão e marcar a saída como 1.

Tabela 8 - Tabela verdade para a expressão  $S = (A \cdot B \cdot \sim C) + (A \cdot \sim B \cdot C) + (A \cdot B \cdot C)$ 

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Fonte: o autor.

Essa expressão, ainda, poderia ser simplificada, mas isso veremos no próximo tópico. Uma expressão simplificada nos permite desenhar um circuito mais simplificado. De qualquer forma, podemos já desenhar o circuito para esse problema da seguinte forma:

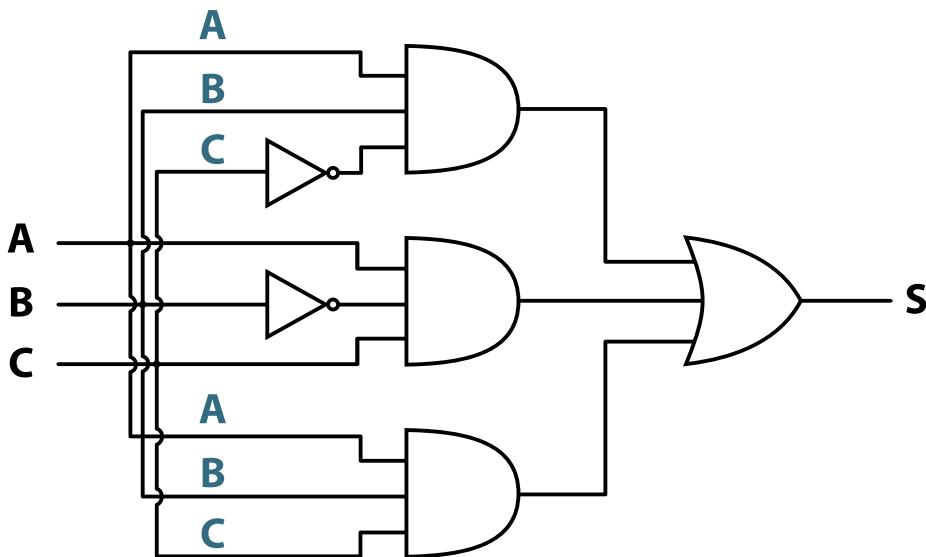


Figura 12 - Representação do circuito

Fonte: o autor.

# Boolean algebra

## NOÇÕES DE ÁLGEBRA BOOLEANA

Da mesma forma que podemos simplificar expressões aritméticas (algo que aprendemos na escola), podemos simplificar, também, expressões lógicas. A área de estudo das expressões lógicas e suas regras é chamada de **álgebra booleana**, em homenagem a George Boole, que foi quem propôs os princípios desta álgebra (MONTEIRO, 2002).

De uma forma resumida, vamos apresentar uma lista com as regras básicas da álgebra booleana. A partir dessas regras, é possível simplificar as expressões e, consequentemente, os circuitos. Usaremos A, B e C como variáveis, além das constantes 0 e 1.

Tabela 9 - Regras da Álgebra Booleana

NOME	FORMA AND	FORMA OR
Identidade	$1 \cdot A = A$	$0 + A = A$
Elemento nulo	$0 \cdot A = 0$	$1 + A = 1$
Idempotência	$A \cdot A = A$	$A + A = A$
Inverso	$A \cdot \sim A = 0$	$A + \sim A = 1$
Comutatividade	$A \cdot B = B \cdot A$	$A + B = B + A$
Associatividade	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	$(A + B) + C = A + (B + C)$
Distributividade	$A + (B \cdot C) = (A + B) \cdot (A + C)$	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
Absorção	$A \cdot (A + B) = A$	$A + (A \cdot B) = A$
De Morgan	$\sim(A \cdot B) = \sim A + \sim B$	$\sim(A + B) = \sim A \cdot \sim B$

Fonte: Tanenbaum (2013).

Na unidade anterior, chegamos a uma expressão que pode ser simplificada:  $S = (A \cdot B \cdot \sim C) + (A \cdot \sim B \cdot C) + (A \cdot B \cdot C)$ . Veremos, a seguir, como podemos simplificar a expressão usando as regras de álgebra booleana, indicando qual regra foi utilizada em cada passagem:

$$\begin{aligned}
 S &= (A \cdot B \cdot \sim C) + (A \cdot \sim B \cdot C) + (A \cdot B \cdot C) \\
 S &= A \cdot ((B \cdot \sim C) + (\sim B \cdot C) + (B \cdot C)) \text{ (Distributividade)} \\
 S &= A \cdot (B \cdot (\sim C + C) + (\sim B \cdot C)) \text{ (Distributividade)} \\
 S &= A \cdot (B \cdot 1 + (\sim B \cdot C)) \text{ (Inverso)} \\
 S &= A \cdot (B + (\sim B \cdot C)) \text{ (Identidade)} \\
 S &= A \cdot ((B + \sim B) \cdot (B + C)) \text{ (Distributividade)} \\
 S &= A \cdot (1 \cdot (B + C)) \text{ (Inverso)} \\
 S &= A \cdot (B + C) \text{ (Identidade)}
 \end{aligned}$$

Conseguimos, então, uma expressão bem mais simples. Para confirmar se os circuitos são equivalentes, basta elaborar a tabela-verdade para a nova expressão e verificar se as linhas das saídas dos circuitos são equivalentes. Para nossa expressão simplificada, a tabela-verdade é apresentada a seguir.

Tabela 10 - Tabela verdade para a expressão  $S = A \cdot (B + C)$

A	B	C	$B + C$	$A \cdot (B + C)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Fonte: o autor.

Observando a última coluna, a coluna da saída da expressão, podemos perceber como a expressão sem simplificar e a expressão simplificada são equivalentes. Da expressão simplificada, obtemos o seguinte circuito:

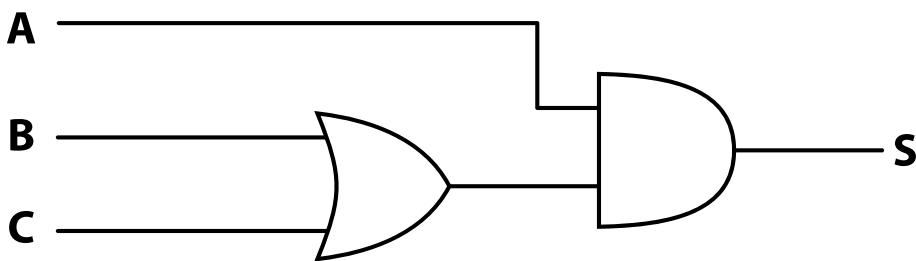


Figura 13 - Circuito lógico da expressão  $S = A \cdot (B + C)$

Fonte: o autor.

Não deixe de comparar esse circuito com o circuito obtido na seção anterior.



#### SAIBA MAIS

Circuitos sequenciais são aqueles que possuem memória. Suas saídas são função tanto das entradas como dos valores da saída. Dito de outro modo, nos circuitos sequenciais o novo valor da saída depende do estado atual destas saídas.

Dois circuitos sequenciais bastante utilizados são os registradores e os contadores. Ambos são construídos com *flip-flops*, ou seja, registradores capazes de armazenar 1 único bit. Dependendo da maneira exata como é controlado, um *flip-flop* recebe várias denominações distintas.

Fonte: Weber (2012, p. 146).

Assim, como o conteúdo da unidade anterior, a criação de circuitos lógicos, expressões e a álgebra booleana são conteúdos que demandam um certo treino para serem bem assimilados. Por isso, não deixe de fazer os exercícios e tente, por conta própria, criar alguns circuitos para resolver problemas.

De uma forma simples, vimos um bom conteúdo sobre circuitos digitais. Pratique exercícios para manter o cérebro sempre em forma!

## CONSIDERAÇÕES FINAIS

Mais uma vez, parabéns pela dedicação empenhada até aqui. Em qualquer modalidade de estudos, o aluno é quem faz a maior diferença em seu aprendizado. Então, não deixe as oportunidades passarem. Faça as atividades, tire as dúvidas, pesquise em material extra...

Nesta unidade, nos dedicamos a mostrar como funcionam os circuitos digitais e as portas lógicas. Neste ponto, você já consegue montar um circuito digital e, também, analisar o comportamento do circuito para diferentes entradas de dados digitais.

Na área de circuitos, ainda há muito conteúdo e muitos componentes que poderiam ser abordados, mas chegamos até onde era nosso objetivo.

Começamos analisando conceitos básicos de lógica digital, a base para as operações em circuitos. Em seguida, vimos como manipular as informações recebidas nas entradas de circuitos, utilizando portas lógicas.

Então, aprendemos a criar circuitos digitais por meio da combinação de portas lógicas, até chegar à álgebra booleana, que nos permite manipular as operações de forma a simplificar expressões lógicas.

Com o que você aprendeu aqui, você consegue criar o esquema para pequenos circuitos e já começa a resolver problemas pequenos de automação. É claro que precisa também de uma prática em montagem de circuitos, não apenas os diagramas.

Algo que ajuda, também, são os simuladores de circuitos digitais, que você encontra para fazer o download pela internet. Por meio deles, você consegue montar circuitos e testá-los, de uma forma bem visual. Consegue testar a passagem de energia, as combinações de portas etc.

Ainda que você não chegue a trabalhar com a montagem de circuitos e componentes, a partir de nosso conhecimento sobre circuitos, o domínio de entendimento sobre o computador e suas funções fica mais aprofundado. Na próxima unidade, trataremos sobre processadores e memória, que são diretamente fruto de circuitos integrados.

## ATIVIDADES



1. Ao encontrar uma expressão lógica que precisa ser resolvida, a tabela-verdade é uma ótima ferramenta de apoio para encontrar a solução. A tabela-verdade é uma forma de escrever todas as combinações de valores de entrada para descobrir qual será a saída em cada caso.

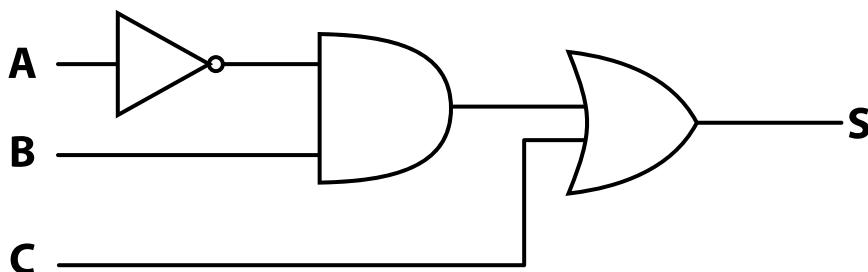
Considerando o texto acima, avalie a expressão abaixo, complete a tabela-verdade e assinale a alternativa com os valores da coluna S.

$$S = (A + B) \cdot (\sim A + C)$$

Nota: “+” simboliza a operação OU, “·” simboliza a operação E, “~” simboliza a operação NÃO.

A	B	C	$\sim A$	$A+B$	$\sim A+C$	S
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

2. Dado o circuito a seguir, desenvolva a expressão equivalente para S e faça a tabela-verdade para o circuito.



## ATIVIDADES

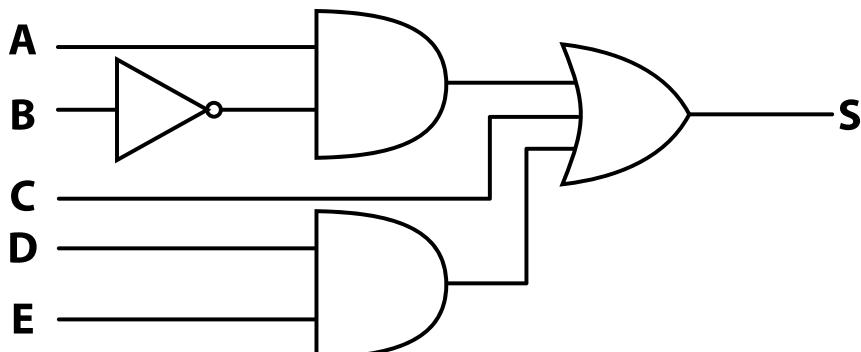


3. Ao criar uma automação, muitas vezes, precisamos ir além dos programas de computador e criar dispositivos para interação com os usuários. A criação de dispositivos eletrônicos capazes de microprocessamentos e também comunicação, têm se popularizado, ainda mais com a chegada da Internet das Coisas. Com isso, a nossa noção de desenvolvimento de circuitos digitais se faz ainda mais importante.

Em termos gerais, um circuito lógico (ou digital) é:

- Uma expressão lógica relacional que gera pulsos digitais positivos.
  - Uma interconexão de portas digitais com sinais assíncronos digitais e saídas.
  - Várias linhas de conexão combinadas em barramentos de uma placa-mãe.
  - Uma ligação entre sinais positivos e negativos de baixa tensão que podem usar transistores.
  - Uma combinação de entradas e portas lógicas digitais que produz uma saída que pode ser 0 ou 1.
4. Um circuito lógico pode ser representado como uma expressão ou como um diagrama, indicando entradas, saídas e portas lógicas. Podemos desenhar o diagrama de um circuito a partir da expressão ou mesmo escrever a expressão a partir do diagrama.

Dessa forma, dado o diagrama de circuito a seguir, escreva a expressão correspondente.



## ATIVIDADES



5. "Em meados do século passado George Boole, um matemático inglês, desenvolveu uma teoria completamente diferente para a época, baseada em uma série de postulados e operações simples para resolver uma infinidade de problemas. Apesar da álgebra de Boole, como foi chamada, poder resolver problemas práticos de controle e fabricação de produtos, na época não havia Eletrônica e nem as máquinas eram suficientemente avançadas para utilizar seus princípios. A álgebra de Boole veio a se tornar importante com o advento da Eletrônica, especificamente, da Eletrônica Digital, que gerou os modernos computadores".

BRAGA, N C. **A Álgebra de Boole.** [On-line]. Disponível em: <<http://www.newtoncbraga.com.br/index.php/electronica-digital/16291-curso-de-electronica-digital-a-algebra-de-boole-cur5002>>. Acesso em: 22 ago, 2018.

Veja a expressão a seguir:

$$S = (A + B) \cdot (A + \sim B) \cdot (C + B)$$

Essa expressão pode ser simplificada utilizando as regras básicas de álgebra booleana vistas na unidade. Simplifique a expressão indicando quais regras foram utilizadas e escreva a expressão simplificada.



## Arduino - Uma plataforma open source para desenvolvimento de eletrônica

Quando pensamos em automação, sensoriamento e controle, normalmente fazemos uma associação com sistemas altamente complexos e caros. Claro que sistemas que necessitam de muita precisão e controle tornam-se mais caros, mas o princípio básico é o mesmo, coletar um dado do ambiente e a partir deste dado fazer alguma ação.

O projeto Arduino foi criado com este propósito, oferecer um hardware e software livre a um custo acessível. Isto abriu possibilidades de muitos adeptos da eletrônica e da programação criarem sistemas de monitoramento e controle que até então só eram comercializados por empresas de tecnologia.

O Arduino foi criado na Itália por um grupo de desenvolvedores e especialistas em engenharia eletrônica que criaram um hardware baseado no chip microcontrolador atmega da empresa Atmel. Estes desenvolvedores criaram uma placa com os componentes para dar ao chip atmega uma estrutura própria para a prototipagem, ou seja, um ambiente de testes para o desenvolvimento de projetos.

### Conhecendo a estrutura do Arduino

O Arduino dispõe de vários modelos de placas, cada uma com características próprias voltadas para o desenvolvimento de protótipos específicos. Algumas placas são muito pequenas como a "Lily" e a "Nano". Estas placas são extremamente pequenas, sendo otimizadas para projetos que não têm muito espaço. Outras placas têm muitas portas/pinos como a placa "Mega". A "Mega" pode conectar um número bem expressivo de sensores e relés, muito útil para projetos que necessitam controlar muitos equipamentos e sensores.

A placa mais comum é a "Uno". Ela tem um número razoável de portas - 13 portas digitais e 6 analógicas, e podem na maioria das vezes suprir as necessidades dos projetos. Vejamos na imagem como é o Arduino Uno, suas especificações e portas.



Vamos inicialmente observar 4 informações, os pinos de conexões:

1. Pinos de Eletricidade - estes pinos são os responsáveis por fornecer energia para os sensores, relés e outros dispositivos. Nestes pinos temos voltagens de 3.3 volts, 5 volts e também os pinos do ground/negativo;
2. Pinos Analógicos - estes pinos são os responsáveis por receber as leituras de vários sensores acoplados ao Arduino. Um sensor de temperatura, de luminosidade ou de distância podem ser conectados nestes pinos;
3. Pinos Digitais - estes pinos trabalham com os dois valores digitais: ligado e desligado. Dentre suas utilidades servem para ligar relés, LEDs e receber informações de sensores que trabalham digitalmente. Alguns pinos digitais recebem outros valores além do ligado ou desligado. Estes pinos são chamados pinos PWM e são utilizados como pinos de entrada ou saída para sensores e LEDs que têm variações em suas correntes;
4. Plug USB - responsável pela conexão com o computador. Esta conexão é a responsável pelo upload do código que será rodado no Arduino. Normalmente depois do projeto feito o cabo USB não é mais utilizado. Existem algumas formas de enviar os dados coletados pelos sensores do Arduino com um computador que é o centro do sistema.

Fonte: Pereira Junior (2015, on-line)<sup>2</sup>.

# MATERIAL COMPLEMENTAR



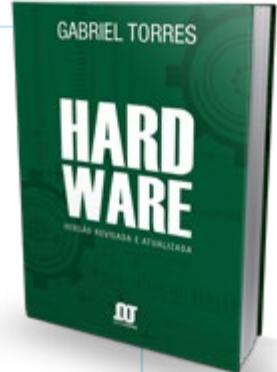
LIVRO

## Hardware: Versão Revisada e Atualizada

Gabriel Torres

**Editora:** Novaterra

**Sinopse:** o maior autor de hardware e redes de computadores do Brasil, Gabriel Torres, está relançando, por meio da parceria com a Editora NovaTerra, o seu principal livro e que o tornou uma referência absoluta na área: Hardware - Versão Revisada e Atualizada. Na obra, Gabriel apresenta um conteúdo completamente revitalizado, onde o leitor aprenderá, com profundidade, tudo o que precisa saber sobre o tema, seja ele um estudante, autodidata ou profissional da área de informática / Tecnologia da Informação (TI) querendo se atualizar. O livro também é importante para os usuários que desejam aprender mais sobre o funcionamento dos computadores.



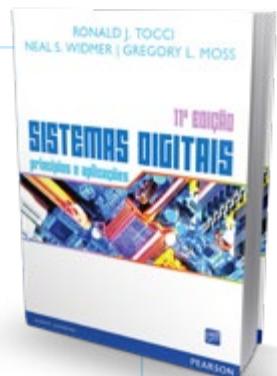
LIVRO

## Sistemas Digitais - Princípios e Aplicações

Tocci,Ronald J. / Widmer,Neal S.

**Editora:** Pearson

**Sinopse:** sabemos que um dos maiores desafios dos autores de obras que tratam de tecnologia digital é encontrar o equilíbrio entre o material já existente e o novo, decorrente das inovações. Foi com isso em mente que os renomados autores Widmer, Moss e Tocci trabalharam nesta 10ª edição de Sistemas Digitais, disponibilizando agora uma obra atual sem, no entanto, deixar de lado os aspectos fundamentais que envolvem os sistemas digitais. Considerando um verdadeiro clássico, o livro oferece amplas condições para que o aluno desenvolva um sólido e moderno conhecimento na área de eletrônica digital. Ele inclui novos exemplos e vários itens foram ampliados e atualizados.



NA WEB

## Instituto Newton C. Braga

Centenas de materiais e artigos a respeito de eletrônica digital feitos pelo Prof. Newton C. Braga, que tem sido referência na área há um bom tempo.

Web: <<http://www.newtoncbraga.com.br/index.php>>.

# REFERÊNCIAS

MONTEIRO, M. A. **Introdução à organização de computadores**. 4. ed. Rio de Janeiro: LTC, 2001. 498 p.

NULL, L.; LOBUR, J. **Princípios básicos de arquitetura e organização de computadores**. 2. ed. Porto Alegre: Bookman, 2011. 822 p.

TANENBAUM, A. S. **Organização estruturada de computadores**. 6. ed. São Paulo: Pearson Prentice Hall, 2013.

WEBER, R. F. **Fundamentos de arquitetura de computadores**. 4. ed. Porto Alegre: Bookman, 2012. 424 p.

## REFERÊNCIAS ON-LINE

<sup>1</sup>Em: <<https://www.vivaolinux.com.br/artigo/Bitwise-Escovando-os-bits>>. Acesso em: 22 ago. 2018.

<sup>2</sup>Em: <<https://www.vivaolinux.com.br/artigo/Arduino-Uma-plataforma-open-source-para-desenvolvimento-de-eletronica/>>. Acesso em: 22 ago. 2018.



# GABARITO

1.

A	B	C	$\sim A$	$A+B$	$\sim A+C$	S
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	1	0	1	1	1
1	1	0	0	1	0	0
1	1	1	0	1	1	1

2. Expressão:  $S = (\sim A \cdot B) + C$ 

Tabela-verdade:

A	B	C	$\sim A$	$\sim A \cdot B$	S
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1

3. E.

$$4. S = (A \cdot \sim B) + C + (D \cdot E)$$

$$5. S = (A + B) \cdot (A + \sim B) \cdot (C + B)$$

$$S = (A + (B \cdot \sim B)) \cdot (C + B) \text{ (Distributividade)}$$

$$S = (A + 0) \cdot (C + B) \text{ (Inverso)}$$

$$S = A \cdot (C + B) \text{ (Identidade)}$$





# PROCESSADOR E MEMÓRIA

UNIDADE

IV

## Objetivos de Aprendizagem

- Detalhar a forma em que o processador é organizado e a função de cada uma das partes.
- Aprender sobre os ciclos do processador para executar as instruções.
- Conhecer as formas utilizadas pelo processador para executar instruções em paralelo.
- Apresentar uma definição de memória e como ela funciona.
- Descrever os diferentes tipos de memória e sua utilização.

## Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Organização do processador
- Funcionamento do processador
- Paralelismo
- Conceito de memória
- Hierarquia de memória



## INTRODUÇÃO

Iniciamos, agora, uma nova unidade, com uma nova etapa do conhecimento. Talvez você fique mais tranquilo, por não precisar fazer tantos exercícios.

Agora que já temos uma boa base sobre como são trabalhadas as informações em circuitos, veremos melhor os componentes importantes de um computador, que também são formados pelos circuitos.

Neste estudo, não iremos nos preocupar muito com tamanhos e modelos de memórias e processadores, porque sabemos que isso é algo que muda muito rapidamente. Nos concentraremos em falar sobre a arquitetura e a organização dos computadores, para que você tenha uma compreensão do papel de cada parte e de como se dá a comunicação entre elas.

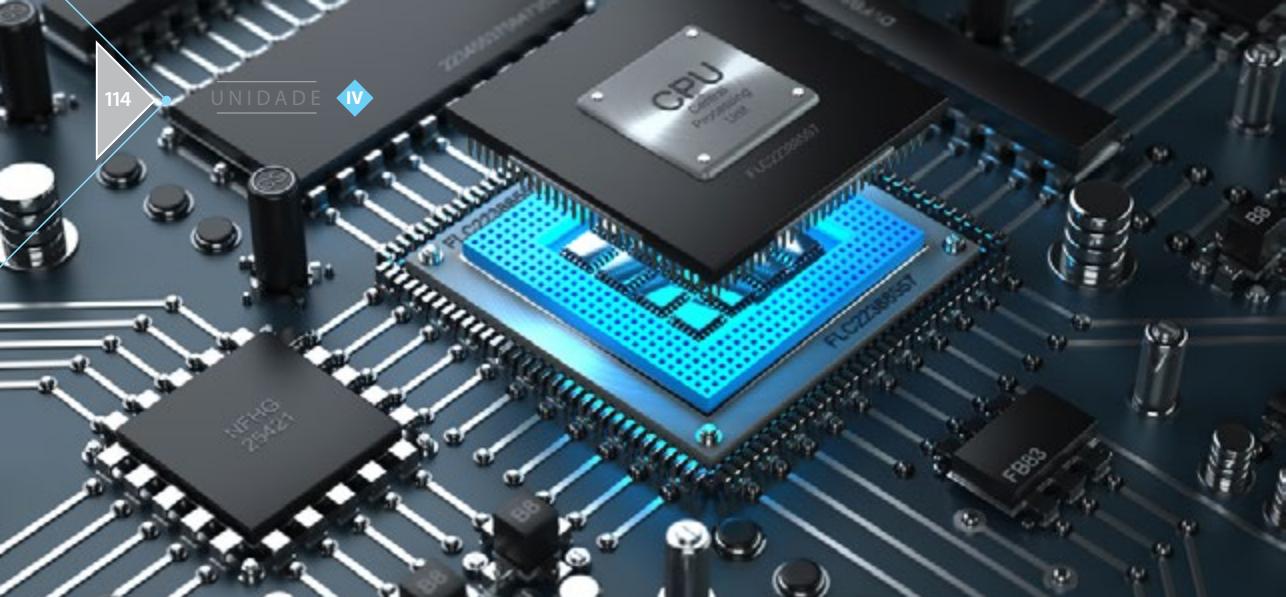
Para começar, veremos um pouco melhor sobre como funciona a arquitetura de von Neumann, que é a nossa arquitetura básica de um computador, e de que ela é composta.

Em seguida, passaremos aos processadores. Primeiramente, veremos como o processador é dividido e o que cada parte faz, para ver, em seguida, como se dá o funcionamento do processador e seus ciclos de instruções.

Então, passaremos para o estudo das memórias, conceituando as memórias de computador e seus diferentes tipos. Veremos as diferenças entre elas e onde elas são aplicadas.

Portanto, esta unidade é composta de um conteúdo um pouco mais teórico, mas muito importante de ser assimilado na área de arquitetura de computadores. Às vezes, quando temos o foco em algo muito específico, como a programação, acreditamos que esse conhecimento é dispensável, porque precisamos apenas de um ambiente de programação e uma interface legal. Conhecer os detalhes contidos nesta unidade nos ajuda a programar de forma melhor e mais otimizada, sabendo como a informação é manipulada pela memória e pelo processador.

Prepare o seu café e vamos mergulhar mais um pouco pelas entranhas de nossos computadores!



## ORGANIZAÇÃO DO PROCESSADOR

Nós já vimos, anteriormente, como funcionam os circuitos e também como a arquitetura de computadores foi organizada basicamente pelo trabalho de John von Neumann, que dividiu o computador em: Unidade Lógica e Aritmética, Unidade de Controle, memória e equipamentos de entrada e saída (TANENBAUM, 2013).

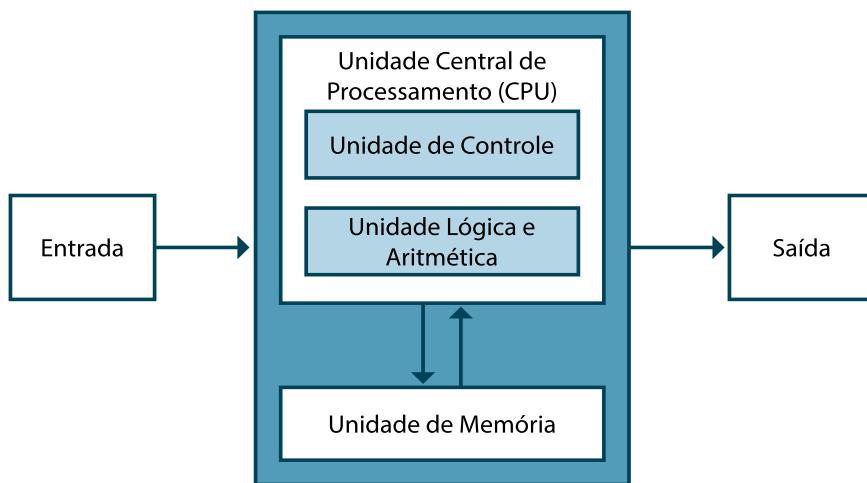


Figura 1 - Modelo de arquitetura de von Neumann  
Fonte: o autor.

Neste momento, nos dedicaremos apenas à **Unidade Central de Processamento** (UCP ou CPU). É comum vermos alguém se referir a um gabinete de computador chamando-o de CPU, como se a CPU fosse a caixa de metal do computador. A CPU é o cérebro, é o processador do nosso computador.

E a CPU está subdividida em: **Unidade de Controle (UC)**, **Unidade Lógica e Aritmética (ULA)** e os **registradores**. A comunicação entre a CPU e os demais componentes se dá por meio dos barramentos, que veremos adiante. A figura a seguir apresenta em esquema de CPU.

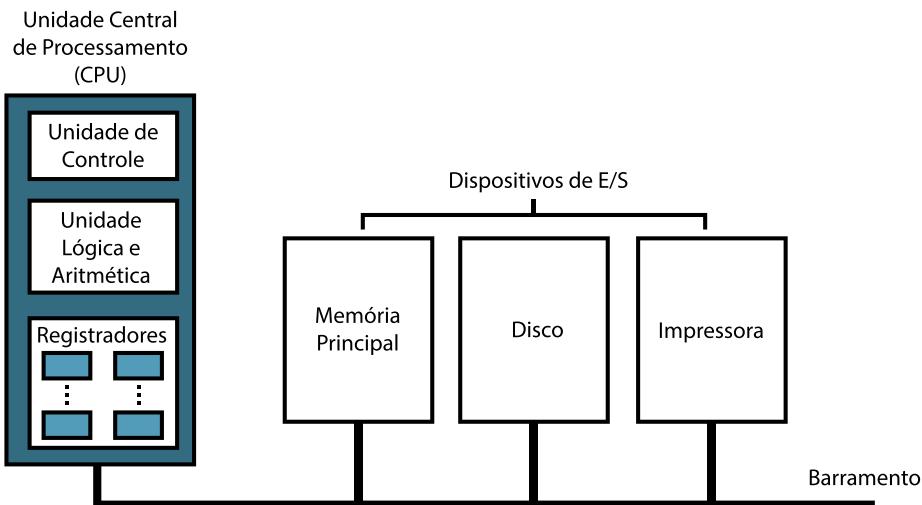


Figura 2 - A organização de um computador simples com uma CPU e dois dispositivos de E/S  
Fonte: Tanenbaum (2013).

## UNIDADE DE CONTROLE (UC)

A Unidade de Controle do processador é quem coordena como tudo vai funcionar. Ela monitora todas as instruções e as execuções, mas ela mesma não executa as instruções.

A unidade de controle extrai as instruções da memória, decodifica essas instruções, assegura que os dados estão no lugar certo, diz para a ULA quais registradores usar, atende interrupções e ativa o circuito apropriado da ULA para executar a operação desejada (NULL, 2011, p. 210, adaptado).

Portanto, a unidade de controle é quem cuida da lógica para realizar a movimentação de dados para o processador. A UC fica conectada aos outros elementos do processador e também ao barramento de controle (MONTEIRO, 2002).

## UNIDADE LÓGICA E ARITMÉTICA (ULA)

A Unidade Lógica e Aritmética é responsável pela execução das instruções do computador. Quando há uma instrução a ser executada, essa instrução é copiada para a memória principal. A UC busca as instruções e os dados na memória e os deixa prontos para a ULA realizar a instrução. Então, a ULA recebe os operandos, recebe a operação a ser executada e retorna o resultado da execução.

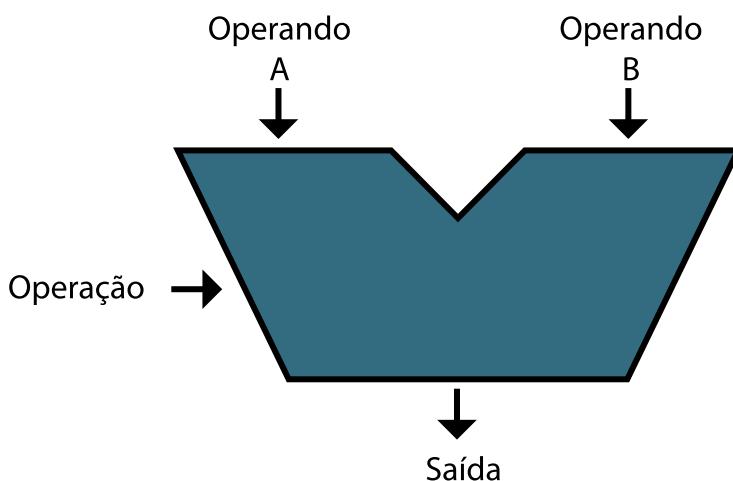


Figura 3 - Funcionamento básico de uma ULA

Fonte: o autor.

As operações básicas que a ULA executa são as operações aritméticas soma, subtração, multiplicação e divisão, as operações lógicas AND, OR, NOT e XOR, as operações de deslocamento à direita e à esquerda e as operações de incremento e decremento (MONTEIRO, 2002).

## REGISTRADORES

Os registradores são unidades de memória muito pequenas e muito rápidas, que existem dentro do processador e todo dado que vai ser transferido pela ULA precisa ser copiado para os registradores (MONTEIRO, 2002). Após a execução da operação, o resultado também é armazenado em um registrador.

Além dos registradores de dados, os processadores ainda possuem registradores com funções de controle, como o registrador de instrução (RI) e o contador de instrução (CI).



SAIBA MAIS

Os processadores trabalham com um tamanho de palavra pré-definido. Quando dizemos que um processador é 32 bits ou 64 bits, estamos nos referindo ao tamanho de palavra que ele usa. Aumentar o tamanho de palavra de um processador implica em aumentar o tamanho dos registradores, o tamanho dos barramentos internos, o tamanho da capacidade de operação da ULA, etc.

Em computadores antigos, o tamanho do barramento interno de dados tinha exatamente o tamanho da palavra. Em processadores modernos esse tamanho de barramento aumentou para 128 bits. Em resumo, aumentar o tamanho da palavra implica diretamente no desempenho de uma CPU.

Fonte: Monteiro (2002).

Esses não são os únicos componentes internos em uma CPU, mas podemos não aprofundar muito em detalhes. Alguns elementos, como o *clock* da CPU, veremos no próximo tópico, ao analisar o funcionamento do processador.

## FUNCIONAMENTO DO PROCESSADOR

Talvez alguém diga: “Ah, o processador não faz nada de mais, ele apenas busca a instrução, executa e retorna o resultado!”

Bom, acho difícil concordar com a parte do “nada demais”, mas o resto da frase, de certa forma, está certo.

Lembram-se de que os computadores foram criados quando as pessoas estavam buscando criar máquinas para automatizar os cálculos? Então, eles conseguiram! O computador faz apenas isso: cálculos o tempo todo. A combinação de cálculos, porém se tornou tão imensa e tão complexa, que a imensidão de cálculos executados por segundo nos proporciona tudo o que fazemos com nossos amados dispositivos.

## CICLO DE INSTRUÇÃO

O processador executa instruções e mais instruções. Cada instrução é executada dentro de um ciclo, que pode ser resumido em **buscar-decodificar-executar** (TANENBAUM, 2013). Basicamente, a unidade de controle é a responsável por *buscar* as instruções e os dados e ainda *decodificar* as instruções, deixando tudo pronto e entregue para a unidade lógica e aritmética *executar* essas instruções. E, ainda, a unidade de controle é quem fica responsável por recuperar o resultado e conduzi-lo à memória ou a outro destino.

Segundo Tanenbaum (2013), o ciclo de instrução compreende as seguintes atividades:

1. Trazer a próxima instrução da memória até o registrador de instrução.
2. Alterar o contador de programa para que aponte para a próxima instrução.
3. Determinar o tipo de instrução trazida.
4. Se a instrução usar uma palavra na memória, determinar onde essa palavra está.
5. Trazer a palavra para dentro de um registrador da CPU, se necessário.
6. Executar a instrução.
7. Voltar à etapa 1 para iniciar a execução da instrução seguinte.

Para facilitar a compreensão, ainda podemos dividir o ciclo de instrução em duas partes: o **ciclo de busca** e o **ciclo de execução**, representados na figura a seguir (MONTEIRO, 2002).

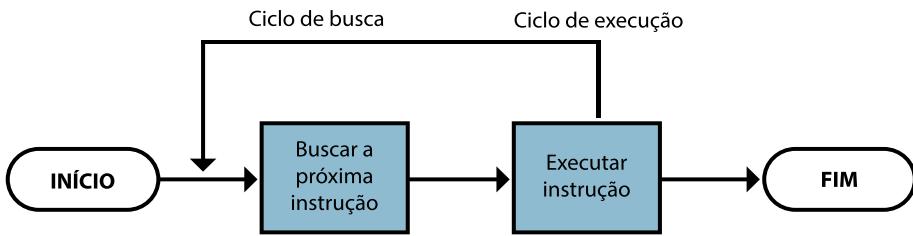


Figura 4 - Representação do ciclo de busca e do ciclo de execução

Fonte: o autor.

## RELÓGIO (CLOCK)

Dentro do processador há um dispositivo chamado *relógio* (mas é mais comum encontrarmos no termo em inglês *clock*) que gera pulsos cuja duração é chamada *ciclo* (MONTEIRO, 2002). O ciclo de relógio (ou ciclo de *clock*) é o intervalo entre o início de um ciclo de pulso e o início do pulso seguinte. Esses ciclos de relógio servem para sincronizar as execuções de instruções no processador e se baseiam em operações elementares. Como as operações ultrapassam o tempo do pulso, são gerados “subciclos”, que são pulsos “atrasados”, para dividir as operações em microoperações.

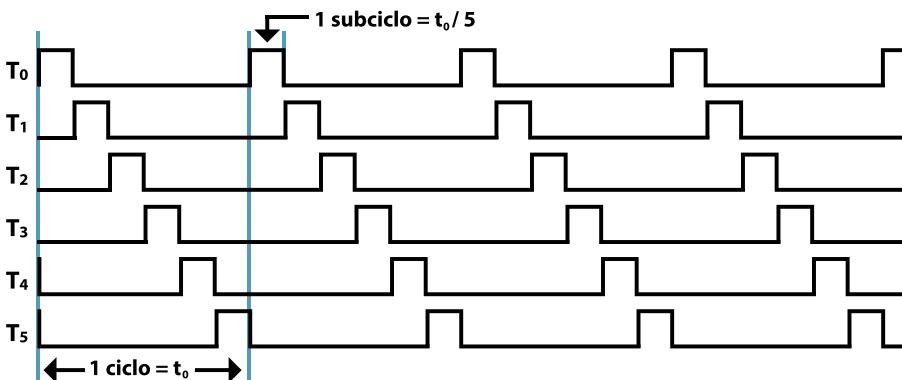


Figura 5 - Representação do ciclo de relógio e os cinco subciclos, baseado no Intel 8085

Fonte: Monteiro (2002).

A medida de desempenho do processador é chamada de *frequência* e nada mais é do que a quantidade de ciclos de relógio que ocorrem em um segundo. A unidade de medida utilizada é o *hertz* (Hz), a qual 1 Hz significa 1 ciclo por segundo, 100 Hz significa 100 ciclos por segundo.



A frequência não é a única característica que afeta o desempenho, mas um processador moderno pode passar dos 4GHz. Calcule, de cabeça, quantos ciclos ocorrem em um segundo nesses processadores.

## BARRAMENTOS

A comunicação realizada entre a CPU e outros componentes é feita via barramentos.

Um **barramento** é um conjunto de fios que atuam como um caminho de dados comum, porém compartilhado, para conectar vários subsistemas dentro do sistema. Ele consiste de diversas linhas, permitindo a transferência de bits em paralelo (NULL, 2011, p. 210).

Os barramentos que interligam a CPU com a memória principal (MP) são:

- *Barramento de dados*: um barramento *bidirecional* para a transferência de informações entre a CPU e a memória principal.
- *Barramento de endereços*: permite a transferência de bits que representam um determinado endereço na memória. É *unidirecional*, pois somente a CPU aciona a memória. A quantidade de linhas do barramento é a quantidade de bits que representam um endereço de memória.
- *Barramento de controle*: possibilita a passagem de sinais de controle entre a CPU e a memória principal de forma *bidirecional*. A CPU envia sinais para a MP indicando se a operação é de leitura (READ) ou escrita (WRITE) e a memória pode enviar sinais de espera (WAIT) (MONTEIRO, 2002).



## PARALELISMO

Em 1965, **Gordon Moore**, cofundador da Intel, disse que o número de transistores que poderia ser colocado dentro de um único chip estava dobrando a cada ano e que essa proporção se manteria num futuro próximo, o que ficou conhecido como **Lei de Moore**. A previsão se cumpriu nos próximos anos e na década de 1970, o ritmo diminuiu para dobrar o número de transistores a cada 18 meses (STALLINGS, 2010).

Há discussões e previsões dizendo que a Lei de Moore está chegando ao fim, mas há muita pesquisa em relação a como aumentar o desempenho de processadores. Com a evolução dos processadores, a velocidade do *clock* (ou frequência) aumentou muito, mas há um limite que pode ser alcançado.

SAIBA MAIS



Definindo de uma maneira objetiva e simples, overclocking (não se usa uma tradução apropriada para esta palavra, mencionando-se, de um modo geral, a própria palavra em inglês) significa a possibilidade de o usuário fazer o processador operar em uma frequência de relógio (mais pulso por segundo) maior do que sua especificação (ou seja, aumentar as “batidas do pêndulo” mais do que o fabricante considera tolerável).

É claro que, procedendo assim (realizando o overclocking), o processador funcionará mais rápido, porém este é um procedimento não recomendável, principalmente para usuários que não dominem amplamente os conhecimentos sobre o funcionamento dos processadores, em particular, e do sistema, como um todo, pois “envenenar” o motor (o processador) para acelerá-lo pode acarretar diversos tipos de problema no funcionamento do sistema, a começar pelo seu “travamento”.

Fonte: Monteiro (2002, p. 251).



Figura 6 - Processador Intel Core i7

Vamos estudar, aqui, algumas estratégias de **paralelismo**, ou seja, a execução de mais de uma instrução ao mesmo tempo. Veremos como funcionam o paralelismo em nível de instrução e o paralelismo de máquina.

## PARALELISMO EM NÍVEL DE INSTRUÇÃO

No **paralelismo em nível de instrução**, a ideia é obter mais instruções executando em paralelo. Para isto, estudaremos a estratégia de *pipelining* e as arquiteturas superescalares.

### *Pipelining*

Já vimos que o ciclo de instrução pode ser dividido nas etapas de busca e execução. A fase de busca costuma ser o gargalo na velocidade de execução de instruções. Para resolver o problema, os fabricantes incluíram, nos processadores, a capacidade de buscar as instruções de forma antecipada, armazenando a em registradores (*prefetch buffers*) (TANENBAUM, 2013).

O conceito de *pipelining* consiste em dividir o ciclo de instrução em várias etapas menores, cada parte é tratada por partes diferentes do hardware, de forma a executar partes de instruções diferentes de forma simultânea. O conceito fica um pouco mais claro na imagem a seguir, na qual temos uma representação de um *pipelining* de 4 estágios.

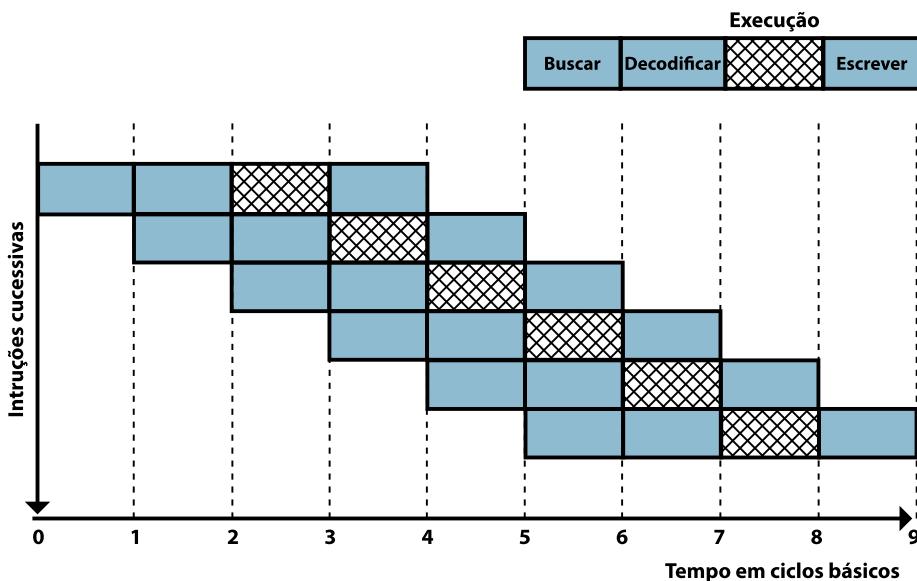


Figura 7 - Representação de pipeline de 4 estágios

Fonte: Stallings (2010).

Percebemos, na imagem, que temos 6 instruções sendo executadas e que levam 4 ciclos de tempo cada. Em sequência, demoraria 24 ciclos para todas serem executadas, mas nesse pipeline de 4 estágios, todas são executadas dentro de 9 ciclos.

Vale ressaltar que as instruções intercaladas não podem depender do resultado de uma das anteriores. O processador deve possuir uma estratégia para evitar os conflitos.

## Arquiteturas superescalares

Enquanto o uso de uma pipeline pode fornecer um bom ganho de velocidade aos processadores, um processador **superescalar** é um processador que possui múltiplos pipelines independentes. Esses múltiplos pipelines possibilitam múltiplos fluxos de instruções ao mesmo tempo (STALLINGS, 2010).

Usar múltiplas pipelines é interessante, mas exige duplicar os recursos de hardware. Para evitar o alto custo, as CPUs topo de linha podem utilizar um único pipeline, mas com várias unidades funcionais (TANENBAUM, 2013). A linha Core da Intel utiliza uma estrutura similar.

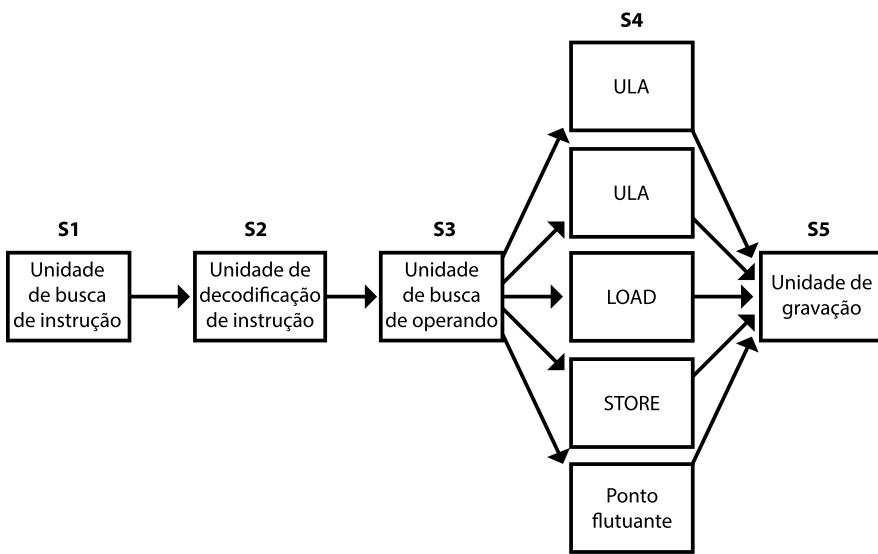


Figura 8 - Processador superescalar com 5 unidades funcionais

Fonte: Tanenbaum (2013).

## PARALELISMO DE MÁQUINA

As estratégias de paralelismo em nível de instrução fornecem um ganho de velocidade, mas em pequena escala. Para aumentar centenas de vezes o desempenho, a forma é utilizar várias CPUs (TANENBAUM, 2013).

Existem formas diferentes para utilizar paralelismo de máquina. Estudaremos três diferentes tipos de paralelismo de máquina.

### Computadores paralelos

Sabemos que podemos aumentar a capacidade de um processador se conseguirmos dobrar a quantidade de transistores dentro dele (o que comentamos sobre a Lei de Moore), mas os transistores ficam cada vez menores e há os seus limites.

De uma forma fácil de imaginar, também podemos dobrar a capacidade se usarmos dois processadores em uma mesma máquina (estou abstraindo as

partes difíceis em ajustar a arquitetura para isso). Se tivermos dois processadores, podemos usar a ideia que já aprendemos ao ver pipelines e dividir as instruções entre os processadores.

Essas instruções podem compartilhar o estágio de busca, decodificação e o conjunto de lógica de controle. Com isso, surgiu o tipo de processadores **SIMD** (*Single Instruction-stream Multiple Data-stream* - fluxo único de instruções, fluxo múltiplo de dados). Esses processadores consistem em “um grande número de processadores idênticos que efetuam a mesma sequência de instruções sobre diferentes conjuntos de dados” (TANENBAUM, 2013, p. 54).

O primeiro processador SIMD foi feito na década de 1970, mas as GPUs (unidades de processamento gráfico) atuais utilizam a estratégia de processamento do SIMD.

## Multiprocessadores

No conceito anterior, as CPUs compartilham de uma mesma unidade de controle. No conceito de **multiprocessador**, um sistema possui múltiplas CPUs independentes que acessam uma memória compartilhada via barramento. Esse projeto, ainda, pode conter pequenas unidades de memória local para cada CPU, para diminuir o tráfego e o conflito no acesso à memória principal (TANENBAUM, 2013).

Dentre as vantagens no uso da memória compartilhada, verifica-se que é mais simples acessar a memória em um único local, além de não correr o risco de a memória estar duplicada em diferentes locais, podendo até gerar conflito na realização de alterações.

## Multicomputadores

O conceito de **multicomputadores** é similar ao conceito de multiprocessadores, mas sem utilizar memória compartilhada. Ao invés de utilizarem memória compartilhada, cada CPU conta com sua memória privada. “Costuma-se dizer que as CPUs em um multicomputador são **fracamente acopladas**, para contrastá-las com as CPUs **fortemente acopladas** de um multiprocessador” (TANENBAUM, 2013, p. 56). A comunicação entre essas CPUs é realizada por meio de troca de mensagens.

Assim, temos que a programação em multiprocessadores é mais fácil, por contarem com uma memória compartilhada e com uma forma compartilhada de acessar as informações, enquanto os multicomputadores são mais fáceis de serem construídos. Por isso, há pesquisas a respeito de sistemas híbridos, juntando o melhor de cada abordagem.

## RISC VS. CISC

O projeto de processadores envolve o conjunto de instruções que podem ser efetuadas nele, o que é essencial para a programação de qualquer sistema. Inicialmente, os processadores seguiam a abordagem **CISC** (*Complex Instruction Set Computer* - Computador com conjunto de instruções complexas), na qual as instruções eram mais complexas, podiam englobar mais operações em uma instrução a fim de que os programas ficassem menores e o trabalho do programador fosse facilitado (STALLINGS, 2010).

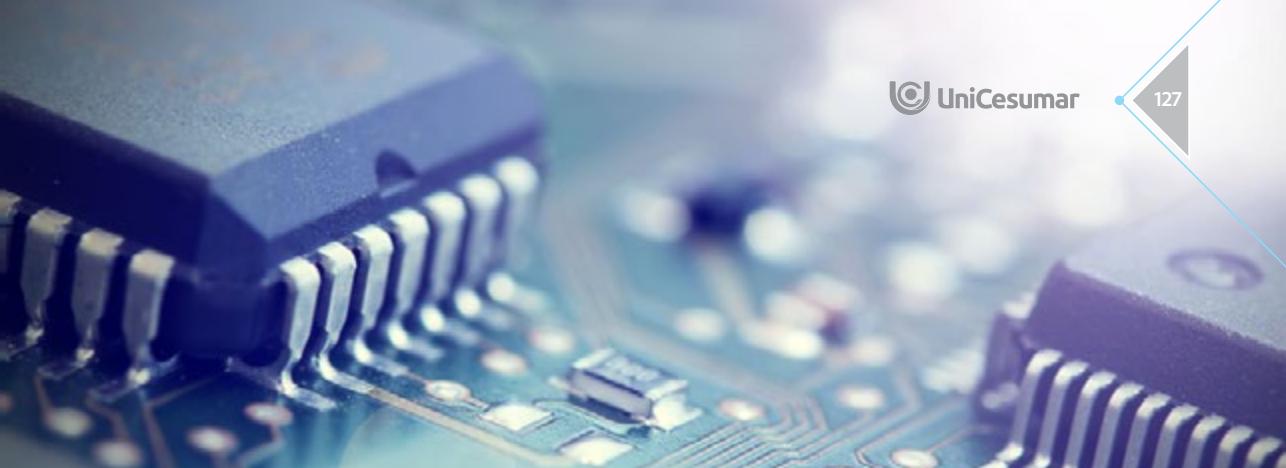
Os programas finais, porém não ficavam tão pequenos e veio a proposta dos computadores **RISC** (*Reduced Instruction Set Computer* - Computador com conjunto de instruções reduzidas), que teria, como o nome diz, um conjunto de instruções reduzidas e mais simples, em que cada uma das execuções pudesse ser executada em apenas um ciclo do processador. Com isso, o RISC poderia obter um ganho de desempenho em relação ao CISC.

No começo da década de 80, a linha da Intel, por exemplo, estava utilizando a abordagem CISC, enquanto PowerPC utilizava a abordagem RISC.

Apesar de ambas as abordagens terem adeptos e defensores, logo os dois lados perceberam duas coisas: primeiro, o RISC pode se beneficiar de alguns recursos CISC; segundo, o CISC pode se beneficiar de alguns recursos RISC.

Assim, o PowerPC passou a não ser mais um “RISC puro” e os Pentium, da Intel, passaram a incorporar algumas características RISC (STALLINGS, 2010).

Ao final desta unidade, na área da “leitura complementar”, você vai encontrar um artigo chamado “Controvérsia de RISC versus CISC”, que trata dessa questão de que as abordagens estão se misturando cada vez mais.



## CONCEITO DE MEMÓRIA

Durante todo o texto, falamos muito em memória, armazenamento em memória, representação de dados na memória, comunicação do processador com a memória, entre outras coisas. Você já percebeu como a memória é importante para um sistema de computação. Agora é a hora de dedicarmos nosso estudo a ela.

Reprodução proibida. Art. 184 do Código Penal e Lei 9.610 de 19 de fevereiro de 1998.



Figura 9 - Tirinha “O que ainda falta?”

Fonte: Noel (2014, on-line)<sup>1</sup>.

Basicamente, a memória de um computador é onde são armazenados e recuperados os dados. Já vimos que são armazenados em bits e já vimos como os bits se juntam para formar informações maiores, portanto, não precisamos voltar a esse nível de detalhamento.

Podemos imaginar a memória como um armário onde colocamos nossa informação, cada bloco de memória (memórias modernas trabalham com blocos de 8 bits - 1 byte) tem um endereço numérico para que a informação possa ser recuperada. Se a informação possui, por exemplo, 4 bytes, o endereço será relativo ao primeiro byte da informação.

Em programação, quando armazenamos valores em uma variável é como se guardássemos a informação no “grande armário” da memória, com uma etiqueta, que seria o nome da variável, para facilitar a localização.

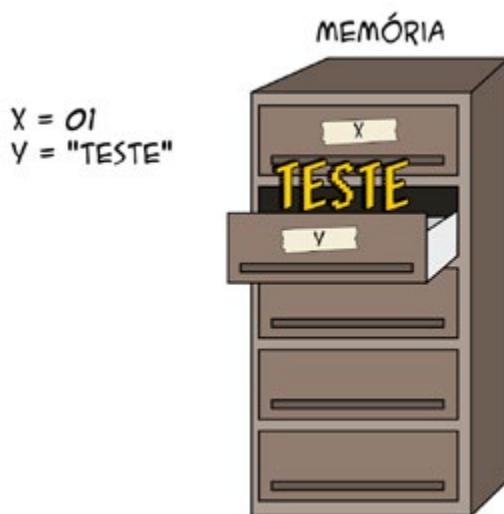


Figura 10 - Representação do armazenamento de variáveis em memória  
Fonte: o autor.

Existem apenas duas operações básicas que efetuamos na memória. A primeira é a operação de *escrita* (*write*), também chamada de *gravação* ou *armazenamento*, que é quando escrevemos a informação que desejamos na memória. A segunda é a *leitura* (*read*), ou *recuperação* da informação (MONTEIRO, 2002).

## TIPOS DE MEMÓRIA

Todo sistema de computador trabalha com diferentes tipos de memória ao mesmo tempo. Como o custo de leitura e escrita de informações é alto, ele precisa ser otimizado para cada fim.

Existem as memórias para armazenamento permanente, que conhecemos como **memória secundária**, que é onde você guarda seus documentos, suas fotos, os sistemas que você instala, o sistema operacional... resumindo, tudo.

Essa memória, porém é geralmente muito (muito, muito) mais lenta do que a velocidade em que o processador trabalha, ou seja, não compensaria executar os programas direto da memória secundária.

É aí que entra a **memória principal**, que é uma memória muito mais rápida, porém volátil, ela não mantém a informação armazenada quando o sistema é desligado e ela fica sem energia. Dessa forma, todo sistema instalado na memória secundária é copiado para a memória principal para ser executado. O mesmo ocorre com os dados, que são copiados para a memória principal para serem utilizados.

*“Por que não fazem a memória principal do mesmo tamanho da memória secundária, para copiar tudo para a memória principal quando ligar o computador e executar tudo mais rápido?” – O rapaz de camiseta cinza ali do fundo pode perguntar.*

A ideia é boa, mas o custo da memória principal é muito maior, pensando em custo por bytes. Pode ver que você encontra máquinas com 1Tb de armazenamento e apenas 8Gb de memória RAM.

Há memórias ainda menores e mais temporárias do que a memória principal, como as caches e os registradores, também há diversos tipos de memórias secundárias, mas deixaremos para ver isso no próximo tópico.

## HIERARQUIA DE MEMÓRIA

Falamos há pouco sobre a importância da memória principal e como ela é utilizada, porém ela ainda é mais lenta do que os processadores e a busca de informações pode causar um gargalo. Para evitar isso, há uma memória intermediária chamada **memória cache**, que é bem mais rápida, mas muito menor. Ela armazena informações que serão utilizadas pelo processador a fim de “apressar” a recuperação de informações.

Ainda há, nos processadores, uma memória menor e mais rápida, os **registradores**, que armazenam informações pequenas, como os operandos de uma instrução, por exemplo.

Para entendermos melhor a relação entre os diferentes tipos de memória, podemos traçar a **hierarquia de memória**, colocando em ordem de velocidade as memórias, como representado na figura a seguir.

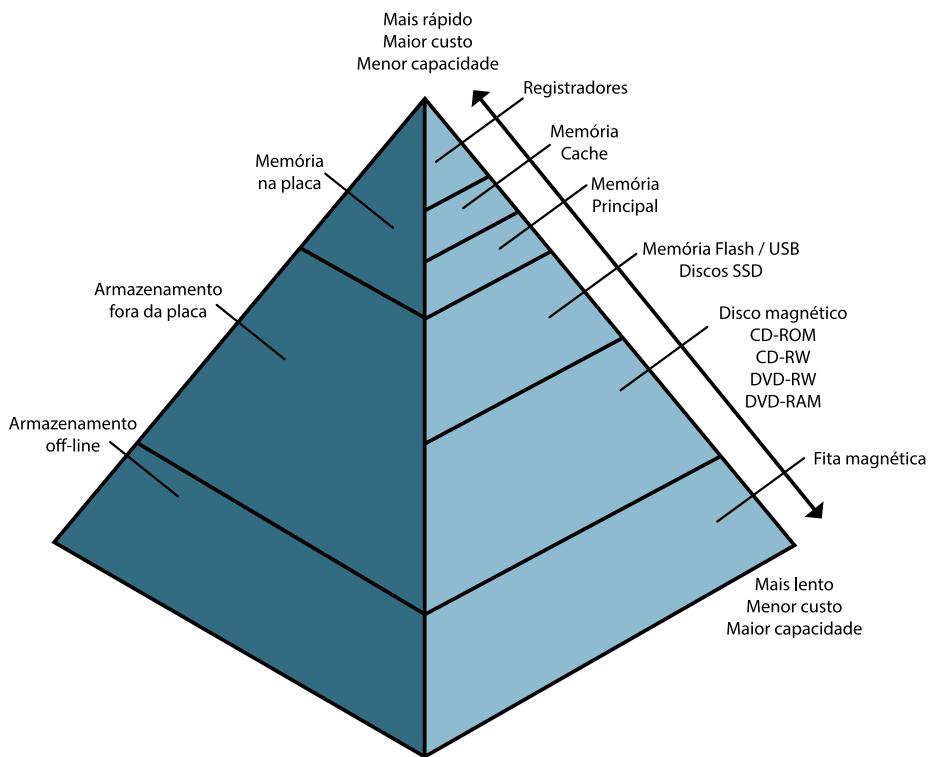


Figura 11- Hierarquia de memória  
Fonte: adaptado de Stallings (2010).

## REGISTRADORES

Os registradores são unidades muito pequenas de memória que ficam dentro do processador, que servem para armazenar pequenas unidades de informação, basicamente, 1 dado por registrador.

“Os registradores são fabricados com capacidade de armazenar um único dado, uma única instrução ou até mesmo um único endereço. Dessa forma, a quantidade de bits de cada um é de uns poucos bits (de 8 a 64)” (MONTEIRO, 2002, p. 116).

São um tipo de memória volátil, semicondutora, e está no topo da pirâmide em relação ao custo. A quantidade de registradores e o uso dado a cada um varia muito de acordo com a CPU.

Além dos registradores de dados, o processador (pode variar de acordo com o modelo), ainda, tem o *registrador de instrução* (RI), o *controlador de instrução* (CI) ou *program counter* (PC), o *registrador de endereços de memória* (REM) e o *registrador de dados de memória* (RDM).

Na próxima unidade, veremos o conceito de interrupções. Para as interrupções, o estado do programa é salvo em um registrador especial chamado de *Program Status Word* (PSW), ou registrador de estado da palavra (MONTEIRO, 2002).

## MEMÓRIA CACHE

Como já dissemos, a memória cache fica entre o processador e a memória principal, conceitualmente. Na prática, os processadores incluem neles 1 ou 2 níveis de cache, é mais rápida que a memória principal e serve para guardar informação temporária da memória principal para diminuir o custo de recuperação da informação.

É comum o uso de memória cache em diferentes níveis. Diversos sistemas utilizam três níveis de cache, que são chamadas de cache L1, cache L2 e cache L3 (a letra “L” vem de “level” - “nível”).

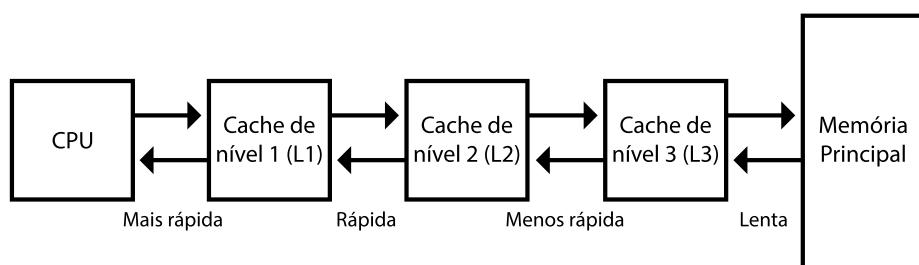


Figura 12 - Organização da cache em três níveis  
Fonte: Stallings (2010).

A cache L1 é a menor e mais rápida, nos níveis seguintes elas aumentam, mas ficam mais lentas. Nem todos os processadores apresentam os mesmos níveis de cache ou trabalham com elas da mesma forma.

## MEMÓRIA PRINCIPAL

A memória principal, também conhecida como *memória principal semicondutora*, é a forma mais comum de armazenamento de acesso aleatório, por isso também é conhecida como **RAM** (*Random Access Memory* - Memória de Acesso Aleatório).

Ela consiste em células de memórias que apresentam dois estados estáveis (ou semiestáveis) para representar a informação (0 ou 1), podem ser escritas pelo menos uma vez e podem ser lidas (STALLINGS, 2010).

A memória RAM, utilizada nos computadores, pode ser de um dos dois tipos: DRAM ou SRAM. A memória **DRAM** (*Dynamic RAM*) é uma memória que possui armazenamento de dados de forma dinâmica, em capacitores. Como os capacitores têm uma tendência de descarregar, as memórias DRAM precisam ficar se recarregando para manter os dados (STALLINGS, 2010).

Já a memória **SRAM** (*Static RAM*) é uma memória estática, que mantém seu conteúdo mesmo quando não tem alimentação disponível. Utiliza circuitos similares a flip-flops tipo D e é muito mais rápida e mais cara do que a DRAM. A DRAM, porém é mais densa, permitindo armazenar mais bits por chip, usa menos energia, o que gera menos calor do que a SRAM (NULL, 2011).

## ROM, PROM, EPROM, EEPROM E MEMÓRIA FLASH

Além das memórias de acesso aleatório, existem as memórias utilizadas para armazenamento, que vão desde a famosa “memória da BIOS” até os armazenamentos de dados pessoais.

A princípio, a memória **ROM** (*Read-Only Memory*) é uma memória *somente-leitura*, uma memória que vem gravada de fábrica, não permite modificação e é bastante usada em sistemas embarcados, brinquedos e eletrodomésticos (NULL, 2011).

A memória ROM possui variações. A primeira, a memória **PROM** (*Programmable Read-Only Memory*) é um tipo que pode ser escrita com equipamentos apropriados, depois de programada, não pode ser apagada (NULL, 2011).

Depois, temos a **EPROM** (*Erasable PROM*), que funciona como a PROM, mas pode ser apagada e reprogramada usando luz ultravioleta. E, ainda, a **EEPROM** (*Electrically Erasable PROM*), que pode ser reescrita apenas com eletricidade, byte a byte (NULL, 2011).

Por fim, temos a **Memória Flash**, que é um tipo de memória EEPROM que pode ser escrita ou apagada em blocos, sendo bem mais rápida do que a EEPROM, e é ela a memória por trás dos *pendrives USB*, que também são conhecidos como “*USB flash drives*”.



SAIBA MAIS

Um disco **SSD** (*Solid-State Disks* - discos em estado sólido, esses discos que estão substituindo os discos magnéticos) é feito também a partir de memória flash.

“Visto que os SSDs são basicamente memória, eles possuem desempenho superior aos discos giratórios, com tempo de busca zero. Enquanto um disco magnético típico pode acessar dados em até 100 MB/s, um SSD pode operar duas a três vezes mais rápido. E como o dispositivo não possui partes móveis, ele é muito adequado para uso em notebooks, onde trepidações e movimentos não afetarão sua capacidade de acessar dados. A desvantagem dos SSDs, em comparação com discos magnéticos, é o seu custo. Enquanto os discos magnéticos custam centavos de dólar por gigabyte, um SSD típico custará de um a três dólares por gigabyte, tornando seu uso apropriado apenas para aplicações com drive menor ou em situações em que o custo não é um problema. O custo dos SSDs está caindo, mas ainda há um longo caminho até que alcancem os discos magnéticos baratos”.

Fonte: Tanenbaum (2013, p. 77).

## DISCOS

Os discos já são velhos conhecidos na área de armazenamento. Temos os **discos magnéticos**, que envolvem discos rígidos e disquetes (que já não estão mais em uso, mas vale a pena mencioná-los), e os **discos óticos**, como CD, DVD, BluRay e todas as suas variações.

Os discos magnéticos, como o nome já diz, usam o campo magnético para leitura e gravação de dados. Focando mais em discos rígidos (HDS), temos a seguinte descrição:

Um disco magnético é composto de um ou mais pratos de alumínio com um revestimento magnetizável. No início, esses pratos tinham até 50 cm de diâmetro, mas agora têm normalmente de 3 a 9 cm, e discos para notebooks já estão com menos de 3 cm e continuam encolhendo. Um cabeçote de disco que contém uma bobina de indução flutua logo acima da superfície, apoiado sobre um colchão de ar. Quando uma corrente positiva ou negativa passa pelo cabeçote, ele magnetiza a superfície logo abaixo dele, alinhando as partículas magnéticas para a esquerda ou para a direita, dependendo da polaridade da corrente. Quando o cabeçote passa sobre uma área magnetizada, uma corrente positiva ou negativa é induzida nele, o que possibilita a leitura dos bits armazenados antes. Assim, à medida que o prato gira sob o cabeçote, uma corrente de bits pode ser escrita e mais tarde lida (TANENBAUM, 2013, p. 68).

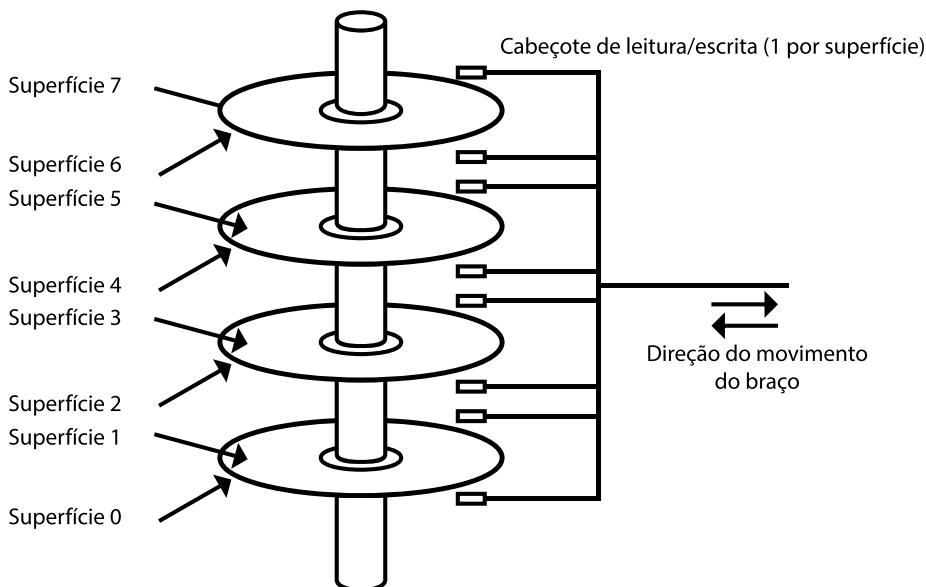


Figura 13 - Representação de um disco magnético com 4 pratos  
Fonte: Tanenbaum (2013).

A velocidade de leitura ou escrita nesses discos depende, basicamente, de 3 fatores: o tempo de busca (*seek*), que é relativa à movimentação do braço de leitura, o tempo de **latência rotacional**, que é o tempo que o disco leva para girar até a posição correta, e o **tempo de transferência** dos dados, que é o tempo de ler ou escrever a informação efetivamente.

Não vamos nos estender com mais detalhes sobre os discos magnéticos, mas eles possuem uma estrutura interessante de armazenamento separadas em trilhas, zonas e setores, mas para nossa disciplina, isso já é suficiente.

Os discos ópticos possuem uma ideia similar de armazenamento, mas utilizam feixes de laser para realizar a leitura e a escrita (em discos que permitem a escrita). Esses discos se popularizaram bastante, mas perderam muito espaço para os *pendrives*, seja para armazenamento ou para reprodução de multimídia.

Com isso, chegamos ao fim do nosso estudo sobre as memórias e armazenamento. Espero que a sua memória esteja bem afiada para armazenar toda essa informação.



## CONSIDERAÇÕES FINAIS

Meu caro amigo ou minha cara amiga, nossa disciplina abrange conteúdos muito bons, mas também muito densos e com muita informação. Por se tratar de uma disciplina introdutória, eu tenho filtrado o conteúdo, deixando de fora muitos detalhes para que nenhuma cabeça exploda no percurso (nem a de vocês nem a minha), mas é um conteúdo muito intrigante e apaixonante (quando se olha com olhos apaixonados).

Não sei você, mas eu sempre fui muito encantado com o computador e com tudo a respeito. Às vezes, a minha angústia é a de não ter tempo de me aprofundar em todas as áreas da computação, que são muitas para qualquer pessoa.

Eu espero que você esteja conseguindo acompanhar e entender bem o conteúdo. Minha intenção é trazer os dados e as informações, mas de uma forma que qualquer pessoa, sem conhecimentos anteriores na área, possa acompanhar. É claro que sei que com uma certa bagagem o estudo fica bem mais tranquilo, mas não dá para ser diferente.

Essa unidade foi mais teórica do que as duas anteriores. De certa forma, não exige da nossa parte do cérebro que faz cálculos, mas exige uma boa capacidade de armazenamento em nossa memória secundária.

Vimos, aqui, como os processadores são divididos e qual é o papel da unidade de controle, da unidade lógica e aritmética e dos registradores. Depois, passamos pelo funcionamento do processador, para entender como ele trata as execuções de instrução e como cada ciclo funciona. E vimos conceitos de paralelismo no processamento, para entender melhor como as instruções podem ser executadas de forma simultânea.

Depois, passamos ao estudo da memória, seus tipos e como funciona a questão de gravar e recuperar informações nela, usando os endereços para acesso. Por fim, vimos a hierarquia de memória, detalhando melhor cada tipo de memória utilizada em um sistema de computação.

Espere que você, ainda, esteja firme e forte para a última unidade que está chegando por aí!

## ATIVIDADES



1. Uma CPU (também conhecida como “processador”) funciona como a parte central de um computador, sendo responsável por todos os cálculos, como se fosse o cérebro da máquina. O tempo todo, o processador trabalha com informações e as troca com a memória e com os dispositivos de entrada e saída. Considerando o texto anterior, assinale o que é usado para transferir as informações de um ponto a outro em um computador.
  - a) Polling.
  - b) Chipset.
  - c) Interface.
  - d) Registrador.
  - e) Barramento.
2. Todos os programas utilizados em um computador são copiados de forma temporária para a memória principal, por ela ter uma velocidade muito maior do que a velocidade de memórias secundárias. Ela não consegue, porém armazenar dados de forma permanente.

Considerando o texto anterior, avalie as afirmações a seguir.

- I. Apesar de ser mais lenta, a memória secundária é mais barata quando pensamos no preço por bytes. Por isso, que esse tipo de memória costuma ter uma capacidade de armazenamento bem maior do que o da memória principal.
- II. A memória principal sempre é mais veloz do que a memória secundária por conta do tempo despendido na busca de informações em disco. Por ser sempre em formato de disco, internamente, há uma demora na rotação e posicionamento até a informação desejada.
- III. O tipo de armazenamento de dados em memória secundária pode ser magnético ou não. Como exemplos de armazenamento não magnético, temos os CDs e os DVDs.

Assinale a alternativa que apresenta as afirmativas corretas

- a) Somente as afirmativas I, II e III estão corretas.
- b) Somente a afirmativa II está correta.
- c) Somente as afirmativas I e II estão corretas.
- d) Somente as afirmativas I e III estão corretas.
- e) Somente as afirmativas II e III estão corretas.

## ATIVIDADES



3. [Adaptado ENADE 2014] “O barramento é o elemento de conexão entre todos os componentes do computador, como memória, CPU e dispositivos de entrada e saída. O barramento de dados é o meio por onde serão trafegados os dados; o barramento de endereços transporta a informação do endereço do dispositivo que poderá acessar o barramento de dados; e o barramento de controle serve para determinar o sentido do fluxo de dados (se os dados são de entrada ou saída da CPU), e se os dados devem ser destinados à memória ou a dispositivos de I/O e também para controlar o clock no barramento.”

STALLINGS, W. **Arquitetura e Organização de Computadores**. 8 ed. São Paulo: Pearson Pratice Hall, 2010 (adaptado).

Considerando um computador com um barramento de dados de 3 bits e barramento de endereços de 4 bits, ele poderá endereçar, respectivamente, quantas posições de memória e quantos dispositivos de I/O?

- a) 16 e 16.
- b) 16 e 8.
- c) 8 e 8.
- d) 8 e 4.
- e) 4 e 8.

## ATIVIDADES



4. “Parte do problema de limitação de velocidade do processador refere-se à diferença de velocidade entre o ciclo de tempo do processador e o ciclo de tempo da memória principal. Ou seja, a memória transfere bits ao processador em velocidades sempre inferiores às que o processador pode receber e operar os dados, o que acarreta, muitas vezes, a necessidade de acrescentar um tempo de espera para o processador”.

MONTEIRO, Mario A. **Introdução à organização de computadores**. 4. ed. Rio de Janeiro: LTC., 2002 (adaptado).

Considerando o texto anterior, assinale a afirmativa correta em relação à função da memória cache.

- a) A memória cache é uma memória flash muito mais rápida que a RAM e de acesso não-volátil.
- b) A memória cache é uma espécie de memória virtual que aumenta a capacidade da memória RAM.
- c) A memória cache fica dentro do processador para aumentar a velocidade dos ciclos de clock de informações.
- d) A memória cache copia blocos de dados muito utilizados da memória RAM para oferecer a informação mais rápido ao processador.
- e) A memória cache mantém um registro de todas as operações já executadas pelo processador, para indicar mais rápido o que fazer no caso de repetir operações.

## ATIVIDADES



5. "Registradores são usados em sistemas de computação como lugares para armazenar uma grande variedade de dados, tais como endereços, contadores de programa ou dados necessários para a execução do programa. Colocando de forma simples, um registrador é um dispositivo de hardware que armazena dados binários".

NULL, L.; LOBUR, J. **Princípios básicos de arquitetura e organização de computadores**. 2. ed. Porto Alegre: Bookman, 2011. p. 209.

Considerando o texto anterior, assinale a alternativa correta.

- a) Os registradores são pequenos componentes de memória que funcionam de forma muito rápida e ficam instalados dentro da memória principal.
- b) Por serem muito pequenos e para usos muito específicos, os registradores são componentes baratos, já que sua velocidade não causa impacto no resto do sistema.
- c) Os registradores ficam dentro do processador e armazenam, por exemplo, as instruções logo antes dela ser executada pela ULA.
- d) Registradores são os componentes de softwares escritos dentro do processador, que indicam como e onde a informação será armazenada.
- e) O uso de registradores em um sistema computacional significa um grande impacto no armazenamento, pois muito mais espaço é garantido para a memória por conta do tamanho dos registradores.



## Controvérsia de RISC versus CISC

Durante muitos anos, a tendência geral na arquitetura e organização de computadores foi aumentar a complexidade do processador: mais instruções, mais modos de endereçamento, mais registradores especializados e assim por diante. O movimento RISC representa uma quebra fundamental com a filosofia por trás desta tendência. Naturalmente, o aparecimento de sistemas RISC e a publicação de artigos pelos seus proponentes exaltando as virtudes de RISC levaram a uma reação por parte daqueles envolvidos no projeto de arquiteturas CISC.

O trabalho que foi feito avaliando os méritos da abordagem RISC pode ser agrupado em duas categorias:

- **Quantitativa:** tenta comparar o tamanho do programa e a velocidade de execução dos programas em máquinas RISC e CISC que usam tecnologias comparáveis.
- **Qualitativa:** analisa questões como suporte para linguagem de alto nível e uso otimizado do estado atual de VLSI.

A maior parte do trabalho em avaliações quantitativas foi feito por aqueles que trabalham em sistemas RISC (PATTERSON e PIEPHO, 1982; HEATH, 1984; PATTERSON, 1984) e foi, em grande parte, favorável à abordagem RISC. Outros analisaram a questão e não ficaram convencidos (COLWELL et al., 1985; FLYNN, MITCHELL e MULDER, 1987; DAVIDSON e VAUGHAN, 1987). Existem vários problemas ao se tentar fazer tais comparações (SERLIN, 1986):

- Não existem pares de máquinas RISC e CISC que sejam comparáveis em custo de ciclo de vida, nível de tecnologia, sofisticação do compilador, suporte de sistema operacional e assim por diante.
- Não existe nenhum conjunto de programas de teste definitivo. Os desempenhos variam com o programa.
- É difícil separar os efeitos de hardware dos efeitos provenientes da capacidade em escrever compiladores.
- A maioria das análises comparativas sobre RISC foram feitas em máquinas “brinquedos” em vez de produtos comerciais. Além disso, a maioria das máquinas comercialmente disponíveis possui propagandas que as destacam como uma mistura de características RISC e CISC. Assim, uma comparação justa com uma máquina comercial real CISC (por exemplo, VAX, Pentium) é difícil.



A avaliação qualitativa é, quase que por definição, subjetiva. Vários pesquisadores voltaram a sua atenção para tal avaliação (COLWELL et al., 1985; WALLICH, 1985), mas os resultados são, na melhor das hipóteses, ambíguos e certamente sujeitos à réplica (PATTERSON; HENNESSY, 1985) e, é claro, à tréplica (COLWELL et al., 1985).

Em anos mais recentes, a controvérsia RISC versus CISC sumiu quase que totalmente. Isso aconteceu porque houve uma convergência gradual das tecnologias. Com o aumento da densidade dos chips e da velocidade do hardware, os sistemas RISC se tornaram mais complexos. Ao mesmo tempo, em um esforço para obter o máximo desempenho, os modelos CISC focaram em questões tradicionalmente associadas com RISC, como um aumento no número de registradores de uso geral e mais ênfase no projeto do pipeline de instruções.

Fonte: Stallings (2010, p. 424-425).

# MATERIAL COMPLEMENTAR



NA WEB

## Como funciona um processador

Vídeo de animação humorística que ilustra bem o comportamento de um processador durante o funcionamento do computador. O vídeo é dos anos 2000, tem termos antigos, mas é muito bom.  
Web: <<https://www.youtube.com/watch?v=VwguixRZsbo>>



NA WEB

## Como funciona o Pen Drive (ART 614)

Artigo do Prof. Newton C. Braga explicando, com detalhes, como funciona um pendrive e como os dados são armazenados na memória flash.

Web: <<http://www.newtoncbraga.com.br/index.php/como-funciona/3619-art614>>



NA WEB

## Cientistas explicam como armazenar 700 TB de dados em um grama de DNA

Esse artigo fala um pouco a respeito de uma técnica de armazenamento de dados em DNA e quanto de dados podem ser armazenados.

Web: <<https://tecnoblog.net/110981/armazenamento-dados-grama-dna/>>

# REFERÊNCIAS

MONTEIRO, M. A. **Introdução à organização de computadores.** 4. ed. Rio de Janeiro: LTC, 2001. 498 p.

NULL, L.; LOBUR, J. **Princípios básicos de arquitetura e organização de computadores.** 2. ed. Porto Alegre: Bookman, 2011. 822 p.

STALLINGS, W. **Arquitetura e organização de computadores.** 8. ed. São Paulo: Pearson Prentice Hall, 2010.

TANENBAUM, A. S. **Organização estruturada de computadores.** 6. ed. São Paulo: Pearson Prentice Hall, 2013.

## REFERÊNCIAS ON-LINE

<sup>1</sup>Em: <<https://vidadeprogramador.com.br/2014/02/19/o-que-ainda-falta/>>.



## GABARITO

1. E.

2. D.

3. A.

4. D.

5. C.





# INTERAÇÃO HOMEM-MÁQUINA

UNIDADE

V

## Objetivos de Aprendizagem

- Conhecer como se dá a entrada e saída de dados de um computador e seus dispositivos.
- Entender qual é a função do sistema operacional de uma máquina.
- Analisar como os aplicativos são executados e como podem ser desenvolvidos.
- Estudar um pouco sobre diferentes licenças de software e sua importância.

## Plano de Estudo

A seguir, apresentam-se os tópicos que você estudará nesta unidade:

- Entrada e Saída (E/S)
- Sistemas operacionais
- Aplicativos e desenvolvimento
- Licenças de software



## INTRODUÇÃO

Chegamos à nossa última unidade. “Aaaaaahhhhhh!” – A plateia emite um som de tristeza. Eu sei, eu também estava me divertindo bastante, mas eu tenho algo a dizer que vai alegrar o seu coração: essa não é uma daquelas disciplinas que o mais difícil fica pro final. Essa unidade é um tanto *light* e é bem mais próxima à sua realidade (assumindo, mais uma vez, que você não seja uma máquina que está lendo e indexando este livro).

Existe uma área de estudo chamada “Interação Humano-Computador” ou “Interação Homem-Máquina”, que visa estudar questões de usabilidade para perceber como o homem tem interagido com a máquina e como isso pode melhorar, mas não é isso o que estudaremos nesta unidade.

Escolhi esse nome para a unidade porque sairemos lá das entranhas do computador para chegar mais próximo ao ser humano, ao usuário. Veremos algumas formas, em hardware e software, que são usadas para as pessoas efetivamente utilizarem as máquinas, interagirem com elas.

Primeiro, veremos como funciona a entrada e saída de um computador. É uma parte muito importante do estudo, porque sem a entrada e saída, pessoas e máquinas não “conversariam”, um seria um inútil para o outro.

Depois, passaremos a falar de software, iniciando pelos sistemas operacionais. Um sistema operacional é um pedaço de software que utilizamos todos os dias e muitas vezes não sabemos exatamente pra que ele serve e o que ele faz, sabemos que é o nome dele que aparece quando ligamos a máquina.

Em seguida, passaremos aos programas/sistemas/aplicativos, que são o que a gente quer usar quando pega um computador (ou smartphone, mas vou generalizar em “computador”), o que são esses programas e como eles “rodam”.

Por fim, veremos um pouco sobre as licenças de software, que implicam em como o aplicativo é distribuído e como as pessoas podem fazer uso deles.

Bom, como você já sabe, é hora de preparar aquele café gostoso para recheiar o nosso cérebro de conhecimento!

## ENTRADA E SAÍDA (E/S)

Agora, você está lendo esse livro na versão digital? Ou na versão impressa? Neste exato momento em que estou escrevendo o livro, estou utilizando muito a entrada de dados para colocar esse monte de letrinhas em um sistema computacional (e a saída para conferir se as letras foram para os lugares certos). Você, que está lendo o que escrevi (você já está no futuro para mim), utilizou o processo de saída de dados do computador, seja para a versão digital ou para a versão impressa.

Vamos relembrar, mais uma vez, a arquitetura de computadores proposta por von Neumann, lá nas primeiras gerações de computadores?

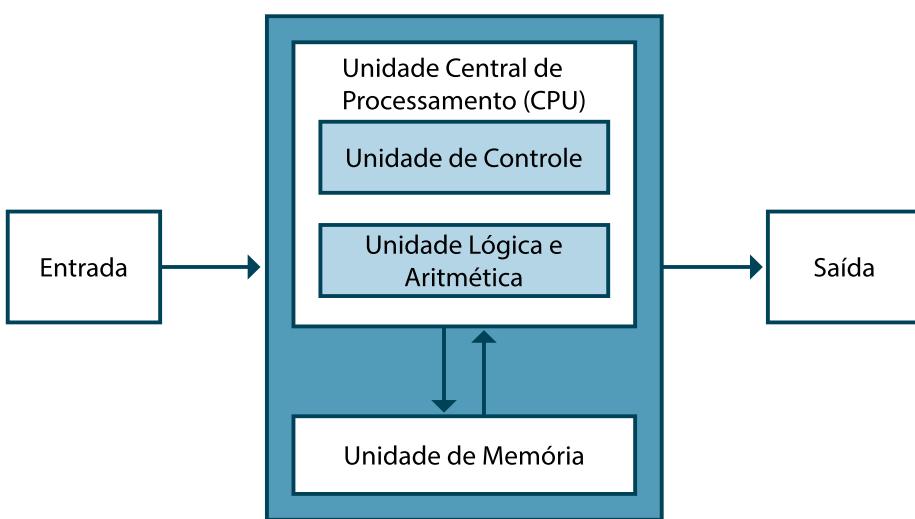


Figura 1- Modelo de arquitetura de von Neumann

Fonte: o autor.

Resumindo, qualquer computador pode ser resumido em três partes: entrada, processamento e saída. Já gastamos um bom tempo falando sobre a parte de processamento, agora, vamos falar um pouco sobre a entrada e saída de dados.

Na imagem pomposa anterior, parece que a entrada e a saída são apenas detalhes, mas elas são muito importantes no uso de um computador. Eu custumo dizer que, para um usuário comum, um computador é melhor representado pela seguinte imagem:



Figura 2 - Modelo de arquitetura segundo o usuário

Fonte: o autor.

Para um usuário leigo, o importante é que ele possa entrar os dados necessários, o computador “faz sua mágica” e retorna os dados esperados. Portanto, a entrada e a saída podem ser enxergadas por ele como as partes mais importantes de um sistema de computação.

De uma forma mais elegante: “O sistema de entrada e saída (E/S) em um computador é responsável pela ligação do processador ao mundo externo. Essa ligação faz-se através de dados entre dispositivos periféricos e o processador ou diretamente entre dispositivos periféricos e a memória” (WEBER, 2000, p. 85).

## SISTEMA DE ENTRADA E SAÍDA

Um sistema de E/S é composto de vários componentes diferentes. Não focaremos o nosso estudo em dispositivos de E/S, porque eles mudam muito, de acordo com os equipamentos do momento. Direcionaremos nosso estudo a entender o que compõe um sistema de E/S e como se dá a comunicação com o computador.

- **Periféricos:** são definidos como periféricos quaisquer dispositivos que possam ser conectados e que efetuam troca de dados com o computador. Podemos ter **periféricos de entrada**, que são aqueles que só transferem dados ao computador (mouse, teclado, joystick etc.), periféricos de saída, aqueles que recebem os dados do processador (monitor, impressora, placa de som etc.), periféricos de entrada e saída, que enviam e recebem dados (pendrives, CD-RW, cartões de memória etc.), ou ainda os periféricos de comunicação, que permitem a comunicação com outras máquinas e/ou pessoas (modem, placa de rede etc.).

- **Interface:** é o componente de hardware que fica entre o processador e o periférico. A interface é quem define como os dados serão transferidos.
- **Controlador:** parte da interface que controla a transferência de dados. O controlador e a interface, muitas vezes, são considerados sinônimos.
- **Driver:** elemento de software que contém as funções para a comunicação da interface com o sistema operacional.
- **Porta de E/S:** o processador possui um determinado número de portas (ou endereços) de E/S, destinados à comunicação com os periféricos por via de suas interfaces e controladores.
- **Barramento:** já vimos que são conjuntos de fios que transportam dados, endereços e sinais de controle. Há barramentos dentro do processador e barramentos externos. Assim, como há o barramento de memória, que liga o processador à memória, há o barramento de expansão, que conecta o processador às interfaces e controladores de E/S (WEBER, 2000).

## TRANSFERÊNCIA DE DADOS DE E/S

Os dados trafegados pela entrada e saída precisam ser transferidos *do* computador, ou *para* o computador. Essa transferência de dados pode ser com o processador (por interrupção ou *polling*) ou pode ser direto com a memória. Estudaremos esses três tipos de transferência.

### Interrupções

Os sinais de entrada e saída, muitas vezes, precisam ser tratados de forma imediata pelo processador. Ninguém gosta quando acontece de digitar uma tecla no teclado e ela demora para aparecer na tela. Ainda, a resposta no monitor precisa ser imediata. Para isso, o processador precisa tratar algumas entradas e saídas no momento em que são requisitadas.

Por outro lado, o processador não pode ficar o tempo todo atento aos sinais de entrada e saída, senão ele desperdiça tempo de processamento esperando sinais que podem nem ocorrer. Não tem como prever quando o usuário vai mexer o mouse, quando vai clicar, quando vai digitar ou parar de digitar...

Nesse ponto é que entram as **interrupções**, que são os mecanismos que o processador usa para tratar eventos de E/S. “Essencialmente, as interrupções permitem que os dispositivos solicitem que o processador pare o que está fazendo no momento e execute um *software* para processar a solicitação do dispositivo” (CARTER, 2003, p. 204). Esse processo é parecido com uma chamada de procedimento de software, mas iniciada pelo dispositivo externo.

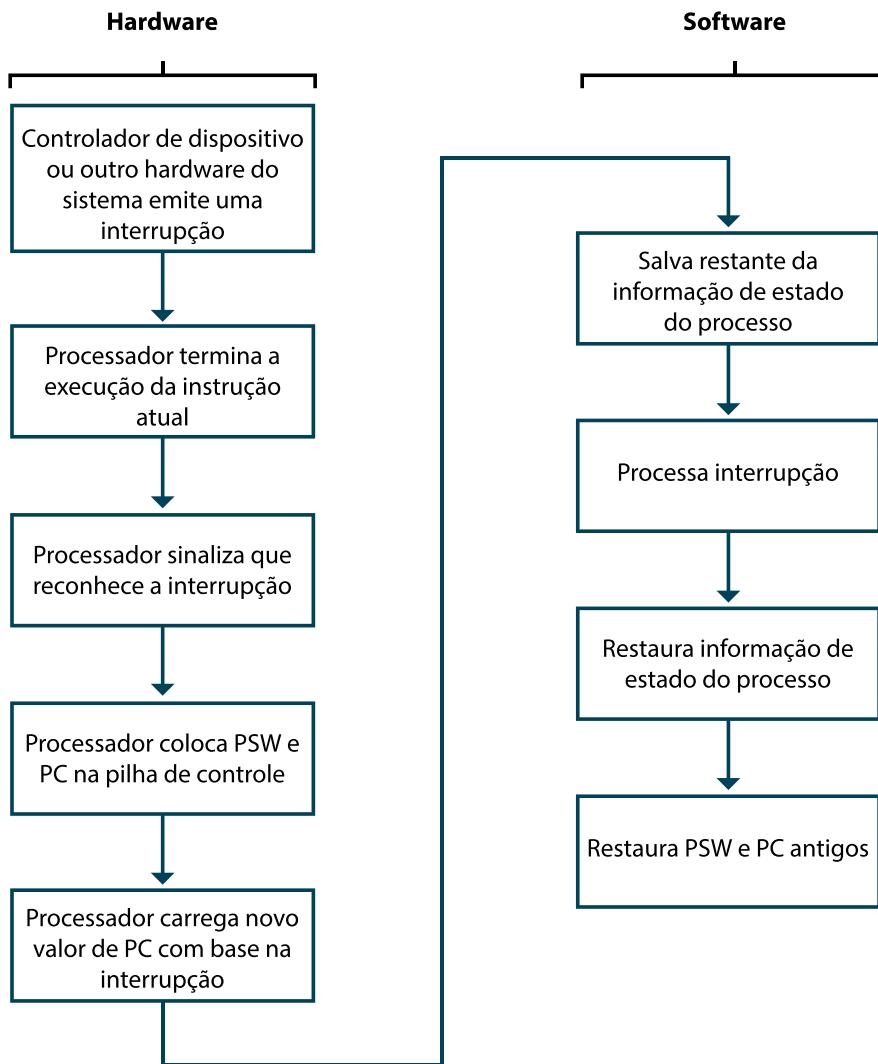


Figura 3 - Processamento de interrupção simples  
Fonte: Stallings (2010).

Além das interrupções de hardware, o termo “interrupção” também foi adotado para interrupções de software e interrupções internas ao processador. Uma **interrupção de software** é resultado de uma instrução que provoca a ativação do mecanismo de tratamento de interrupção. Uma **interrupção do processador** pode ser, também, classificada como interrupção de software, porque são resultados de instruções e síncronas ao programa em execução. As interrupções do processador, porém são resultados da ocorrência de *exceções*, que estudamos em “tratamento de exceções” em programação (WEBER, 2000).

## Polling

Vimos que as interrupções (de hardware) ocorrem de forma assíncrona e o processador trata a entrada ou a saída apenas quando recebe a requisição de interrupção. Uma alternativa a essa forma é o *polling*, que funciona, basicamente, de forma oposta às interrupções.

O processo de *polling* consiste em deixar o processador “atento”, verificando periodicamente os dispositivos de E/S, para identificar se há dados a serem tratados. Isso pode melhorar o desempenho do sistema se o processador não tiver realizando outros trabalhos durante as operações de E/S (CARTER, 2003).

Ainda, para tratar uma exceção, o processador precisa guardar o estado do que está fazendo, parar e tratar a interrupção, o que ocasiona uma espera, ainda que pequena, para tratar E/S.

Em relação ao sistema como um todo, as desvantagens trazidas pelo *polling* fazem com que as interrupções sejam preferidas na maioria dos casos. Em resumo, as vantagens do *polling* são sentidas apenas durante as transferências de dados. Uma desvantagem significativa é que ele ocupa os recursos de processamento, mesmo quando não há operações de E/S. Outra desvantagem é que o *polling* exige que o sistema operacional ou o programa em execução fique fazendo a sondagem nas entradas e saídas, na frequência específica (CARTER, 2003).

Portanto, as interrupções são os métodos mais usados, mas os sistemas podem entrar em modo de *polling* quando precisarem de um tratamento mais exclusivo para E/S.

## Acesso direto à memória (DMA)

Nem todas as informações de entrada e saída precisam passar pelo processador, existe também uma forma de comunicação direta entre os componentes de E/S e a memória, chamada de **acesso direto à memória (DMA)**.

Para que o DMA seja utilizado, é preciso que o computador tenha um módulo adicional no barramento de sistema, o módulo de DMA. Esse módulo consegue simular o processador e assumir o controle do barramento de sistema para transferir a informação direto da E/S para a memória e da memória para E/S (STALLINGS, 2010).

A transferência de dados exige que a DMA controle o barramento de sistemas quando não está sendo utilizado pelo processador ou, ainda, o DMA pode usar uma técnica conhecida como *roubo de ciclo*, forçando o processador a suspender a operação que estava fazendo para controlar o barramento, pelo menos por um ciclo (STALLINGS, 2010).

SAIBA MAIS



Muitos barramentos podem operar em dois modos: modo palavra e modo bloco. Alguns controladores de DMA também são capazes de operar em outro modo. No modo palavra, operação funciona como descrita anteriormente, o controlador de DMA solicita a transferência de uma palavra e consegue. Se a CPU também quer o barramento, ela tem que esperar. O mecanismo é chamado de roubo de ciclo, pois o controlador de dispositivo rouba da CPU um ciclo do barramento, a cada vez alternando.

No modo bloco, o controlador do DMA diz ao dispositivo para obter o barramento, emite uma série de transferências e então libera o barramento. Esse modo de operação é chamado de modo surto. Este é mais eficiente que o anterior, pois várias palavras podem ser transferidas com uma aquisição do barramento. A única desvantagem do modo surto é que ele pode bloquear a CPU e outros dispositivos por um período grande de tempo, caso um longo surto tenha de ser transferido.

Fonte: Moreno (2013, on-line)<sup>1</sup>.

## INTERFACES PARA PERIFÉRICOS

Sabemos que os periféricos são componentes externos ao computador e que precisam ser conectados de alguma forma. Cada periférico tem suas particularidades, por isso, a forma de conexão pode ser diferente. Percebe-se isso ao olhar um computador por trás e ver a quantidade de conectores diferentes.

Em geral, essas interfaces podem ser de forma *serial* ou *paralela*. A diferenciação de ambas é simples. Uma **interface paralela** é uma interface que possui uma ligação por meio de várias linhas, que transmitem os bits em paralelo. A quantidade de linhas varia de acordo com o tamanho da palavra que a interface transfere.

Já a **interface serial** utiliza apenas uma linha e os bits são transferidos de forma serial, ou seja, um depois do outro (STALLINGS, 2010).



Figura 4 - Vista dos conectores das interfaces ligadas à placa-mãe

Em geral, as interfaces paralelas eram mais utilizadas para dispositivos que precisavam de uma transferência de dados mais rápida, porém com o avanço das interfaces seriais de alta velocidade, as interfaces paralelas foram sendo deixadas de lado.

Essa interface serial de alta velocidade é conhecida como *barramento serial universal*, mas é mais conhecida pela sua sigla em inglês **USB** (*Universal Serial Bus*).

Em 1993, representantes de sete empresas (Compaq, DEC, IBM, Intel, Microsoft, NEC e Northem Telecom) se reuniram para buscar a melhor maneira de anexar dispositivos de E/S a um computador. Desde então, centenas de outras empresas se juntaram a elas. O padrão resultante, lançado oficialmente em 1998, é denominado USB (Universal Serial Bus — barramento serial universal), e é amplamente executado em computadores pessoais.

Alguns dos objetivos das empresas que conceberam o USB original e iniciaram o projeto eram os seguintes:

1. Usuários não terão de ajustar comutadores ou pontes em placas ou dispositivos.
2. Usuários não terão de abrir a torre para instalar novos dispositivos de E/S.
3. Haverá apenas um tipo de cabo, que servirá para conectar todos os dispositivos.
4. A energia para os dispositivos de E/S deve ser fornecida por esse cabo.
5. Até 127 dispositivos poderão ser ligados a um único computador.
6. O sistema deve suportar dispositivos de tempo real (por exemplo, som, telefone).
7. Os dispositivos poderão ser instalados com o computador em funcionamento.
8. Não será preciso reiniciar o computador após a instalação do dispositivo.
9. O custo de produção do novo barramento e de seus dispositivos de E/S não deve ser alto (TANENBAUM, 2013, p. 180).

Sabemos que o padrão deu certo e hoje estamos aproveitando a versão 3.0 do padrão USB, que permite velocidade de transferência de até 5 Gbps (gigabits por segundo).



## SISTEMAS OPERACIONAIS

Alguém disse, uma vez, que um sistema operacional está funcionando bem quando não se percebe que ele está ali. E é verdade, o mau funcionamento do sistema operacional é o que mais nos faz lembrar de sua existência.

Então, qual é a real função de um sistema operacional?

Resposta curta: ele possibilita o funcionamento do seu Whatsapp (ou qualquer outro programa que você usa).

Em uma resposta mais longa, nós utilizamos o computador por causa dos programas que queremos utilizar (editor de textos, navegador, player de vídeos etc.) e esses programas precisam ser escritos de uma forma que o computador entenda, ou seja, precisa rodar na arquitetura que já estudamos.

Escrever programas para uma determinada arquitetura, porém é demasiado e trabalhoso. Você precisa aprender as operações do processador na arquitetura específica, precisa lidar com as entradas e saídas, manipular os dados em memória e, o pior de tudo, você precisa criar uma versão de código para cada arquitetura que desejar utilizar.

A função do sistema operacional é estar entre a arquitetura e os programas aplicativos, de forma que, o sistema operacional é quem cuida de toda essa parte de comunicação com a arquitetura, e fornece bibliotecas de funções e interfaces que podem ser utilizadas pelos programas. Assim, um programa pode ser escrito para um sistema operacional e o sistema operacional cuida de executar o programa corretamente para todas as arquiteturas que o S.O. dê suporte. Você pode usar diferentes periféricos ou diferentes organizações de computador para um mesmo software.

Se você imaginou o sistema operacional como um hambúrguer entre dois pães, sendo o pão de baixo o hardware e o pão de cima os programas aplicativos, é basicamente assim que funciona. Por cima do pão superior, ainda podemos ter o gergelim, que seriam os usuários, que possuem contato com os aplicativos, que têm contato com o sistema operacional, que tem contato com a parte de hardware.

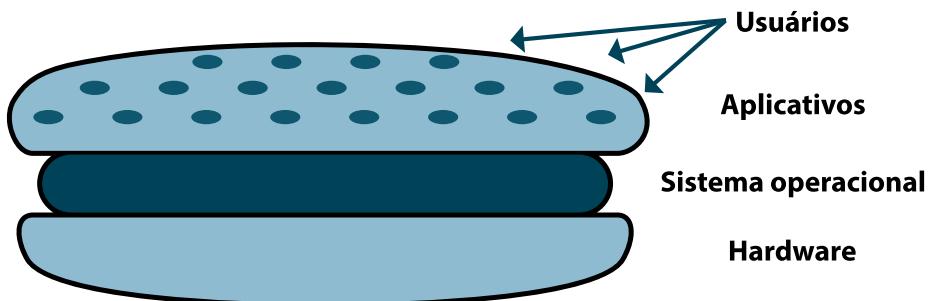


Figura 5 - Representação de relacionamento entre usuários, aplicativos, sistema operacional e hardware  
Fonte: o autor.

Ou ainda, pensando na troca de informações entre as partes, podemos pensar no seguinte diagrama:

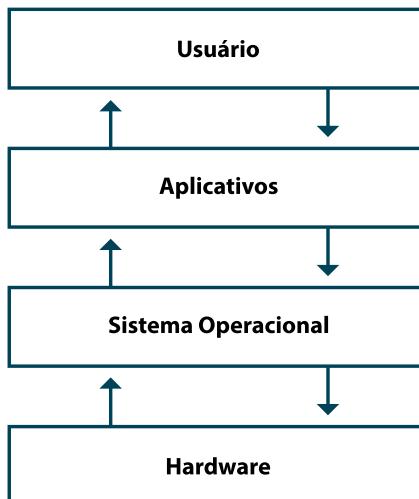


Figura 6 - Representação de relacionamento entre usuários, aplicativos, sistema operacional e hardware, com suas trocas de informação  
Fonte: o autor.

## FUNÇÕES DO SISTEMA OPERACIONAL

Formalmente, um sistema operacional é um conjunto de programas que gerenciam os recursos do computador e o funcionamento dos programas. De forma geral, o sistema operacional tem três funções básicas:

- Controlar os recursos do sistema de computação.
- Executar os programas do computador.
- Gerenciar dados (WEBER, 2012).

## EXEMPLOS DE SISTEMAS OPERACIONAIS

Quando pensamos em sistemas operacionais, costumamos pensar em um ou dois principais. Se pararmos para contar, provavelmente listaremos 5 principais: Windows, Linux, MacOS, Android e iOS, mas existem muitos sistemas operacionais diferentes, com diferentes características. Vamos listar alguns.



### UNIX e Linux

O UNIX é um sistema operacional bem conhecido e bem confiável na área de redes e servidores. Ele foi desenvolvido na década de 1970, por **Ken Thompson**, nos Laboratórios Bell (Bell Labs), que fez a primeira versão, depois aprimorou junto com **Dennis Ritchie**, que é o criador da linguagem C. Um artigo de Thompson e Ritchie sobre o UNIX foi publicado em 1974 (TANENBAUM, 2013).

Em 1987, o professor **Andrew Tanenbaum** criou uma versão simplificada do UNIX chamada MINIX, para ser utilizada para fins acadêmicos. Um de seus alunos, **Linus Torvalds**, após se familiarizar com o MINIX resolveu escrever um sistema similar baseado em UNIX, o Linux (TANENBAUM, 2013).

SAIBA MAIS



Na década de 1980, diversas universidades pelo mundo usavam sistemas com UNIX. Como ele tem uma licença proprietária, em 1984 um rapaz chamado **Richard Stallman** decidiu criar uma versão livre do UNIX e o batizou de GNU (um acrônimo recursivo para *Gnu is Not Unix*), o que começou com todo o movimento de *software livre*, porém ele começou criando os programas que eram compatíveis com o *kernel* (núcleo) do UNIX, para, depois, criar o kernel, que foi chamado de GNU Hurd.

O GNU Hurd nunca chegou a ficar pronto em um nível competitivo, mas os programas do projeto GNU foram o que possibilitou a vida do Linux, que começou a ser criado direto pelo kernel, já que haviam os programas compatíveis com o padrão POSIX, que é o padrão do UNIX. É por isso que até hoje muita gente, incluindo o Richard Stallman, insistem de que o Linux deve ser chamado sempre de GNU/Linux.

Fonte: o autor.

O Linux foi publicado online em 1991, era um kernel livre utilizando os programas GNU e logo conseguiu voluntários pelo mundo todo. Hoje em dia, o Linux é um sistema em software livre bem estabelecido no mercado de servidores e tem ganhado muitos usuários de sistema desktop, graças às suas versões (chamadas *distribuições*).

## Windows

Não é preciso apresentar o sistema operacional Windows, pois ele é o mais utilizado em computadores pessoais, e também já falamos um pouco sobre ele na primeira unidade.

Podemos ressaltar que o Windows, porém começou como uma interface gráfica para o MS-DOS, que era um sistema monousuário e monotarefa, ganhando boa parte das características mais modernas a partir do Windows 9, enquanto o UNIX trabalhava com multiusuários e multitarefas desde as versões iniciais.

Em muitas decisões de projeto, o Windows e o UNIX seguem caminhos bem diferentes que podem afetar a forma de programação, como a forma de realizar chamadas ao sistema.

Fora do núcleo estão os programas do usuário e a biblioteca do sistema usada como interface para o sistema operacional. Ao contrário dos sistemas UNIX, a Microsoft não encoraja os programas do usuário a fazerem chamadas diretas ao sistema. Em vez disso, eles devem fazer chamadas de procedimento na biblioteca. Para padronizar as diferentes versões do Windows, a Microsoft definiu um conjunto de chamadas, denominado API Win32. (...) Mais tarde, quando o Windows foi portado para máquinas de 64 bits, a Microsoft mudou o nome Win32 para abranger ambas as versões, de 32 bits e 64 bits, mas, para nossa finalidade, é suficiente examinar a versão de 32 bits.

A filosofia da API Win32 é completamente diferente da filosofia do UNIX. Nessa última, as chamadas de sistema são todas conhecidas do público e formam uma interface mínima: remover ainda que uma só delas reduziria a funcionalidade do sistema operacional. A filosofia da Win32 é proporcionar uma interface muito abrangente, que muitas vezes oferece três ou quatro modos de fazer a mesma coisa, e inclui muitas funções que claramente não deveriam ser (e não são) chamadas de sistema, tal como uma chamada da API para copiar um arquivo inteiro (TANENBAUM, 2013, p. 384).

O sistema Windows passou por diversas versões, algumas mais populares, como a versão XP que passou mais de 7 anos como a versão principal de muitos computadores (ainda com o Windows 10 como a versão mais atual, diversos usuários ainda utilizam a versão XP), também passou por versões menos populares, como o Windows Me, que ficou muito pouco tempo em atividade. Ainda assim, o Windows é o sistema operacional mais utilizado entre os sistemas desktop.

## Mac OS X

Já vimos alguns rumos da Apple durante a unidade I. Vale ressaltar, aqui, que quando Steve Jobs retornou à Apple com sua empresa NeXT, ele utilizava em seus sistemas um kernel chamado Mach, que também era derivado do UNIX. Com o seu retorno, o sistema operacional da Apple foi todo remodelado e deu origem ao Mac OS X, passando, também, a ter seu kernel baseado em UNIX (MUHAMMAD, 2002). Com isso, o Mac OS X herdou as boas propriedades do UNIX em seu sistema.

Os sistemas da Apple são conhecidos por terem um bom desempenho, principalmente para processamento de imagens e vídeos. Em parte, o bom desempenho

se deve ao fato de que o sistema operacional roda apenas em hardware próprio, com hardware bem dimensionado e controlado, enquanto a plataforma PC possui diversos clones (como já vimos) e diferentes configurações e periféricos que precisa dar conta.

## iOS e Android

Às vezes, falamos sobre sistemas operacionais e nos esquecemos das plataformas móveis, como é uma tecnologia recente, a maioria dos livros sobre arquitetura de computadores ou sistemas operacionais nem os menciona.

Já comentamos, anteriormente, que dispositivos móveis como *smartphones* ou *tablets*, também, podem ser considerados como computadores e seguem um padrão de arquitetura que pode ser resumido em *entrada-processamento-saída*.

O sistema operacional iOS, da Apple, é executado apenas em dispositivos da própria Apple, basicamente nas versões do iPhone e iPad. É um sistema operacional proprietário e segue a mesma ideia do Mac OS X, que tem ganhos de desempenho ao não ter que lidar com muitas variações de hardware.

Enquanto isso, o Android é um sistema operacional em código aberto (Licença Apache 2.0), baseado em Linux e mantido pela Google. Ele roda em dispositivos de diversas fabricantes, em *smartphones*, *tablets* e ainda diversos outros dispositivos. É um sistema que, também, tem que dar suporte a diversos dispositivos e a qualidade, às vezes, depende de como os fabricantes o implementam e o mantêm, mas não perde para a linha da Apple em aparelhos “tops de linha”.

## APLICATIVOS E DESENVOLVIMENTO

Como já foi dito, a parte dos aplicativos é a parte que realmente importa ao usuário no dia a dia. Já vimos, também, que um aplicativo é um software que, geralmente, roda sobre o sistema operacional.

Um **aplicativo** pode usar termos diferentes para defini-lo, e isso muda com o passar do tempo. Pode ser chamado de “programa”, “sistema”, “aplicação”, “app” (abreviação) ou simplesmente “software”.

O sistema operacional é quem cuida da execução de um aplicativo. Para ser executado, o aplicativo pode estar em um arquivo binário que é reconhecido com um arquivo *executável* pelo sistema operacional ou pode ser executado sobre um *interpretador* de software, que lê o programa e o executa.

## LINGUAGENS DE PROGRAMAÇÃO

Software, ou programa de computador, é um conjunto de instruções passado ao computador para que este resolva um determinado problema. Esse conjunto de instruções precisa ser escrito em alguma *linguagem de programação* a fim de que o computador entenda as instruções.

Uma **linguagem de programação** se assemelha, em muitos fatores, às linguagens utilizadas pelas pessoas. Ela possui um conjunto de palavras válidas, uma gramática, um emissor e um receptor etc.

A linguagem de programação mais primitiva seria a **linguagem de máquina**, que basicamente consiste em utilizar as instruções do processador direto em código binário. Pelo que aprendemos sobre binários, dá pra perceber como não seria legal programar em linguagem de máquina, mas é o que era utilizado na primeira geração de programadores (MONTEIRO, 2002).

Depois, temos diferentes níveis de linguagens de programação. Dizemos que uma linguagem é de **baixo nível** se ela se aproxima mais da linguagem de máquina e de **alto nível** se é mais próxima da linguagem natural (linguagem que falamos), se afastando bastante da linguagem de máquina.



SAIBA MAIS



Existem centenas, talvez milhares, de linguagens de programação e suas derivadas, então, pode ter certeza de que você nunca vai aprender todas. O ranking Tiobe ([tiobe.com](http://tiobe.com)) lista quais são as linguagens mais populares no momento pelo mundo, vale a pena conhecer.

Uma outra boa ferramenta é o repl.it (esse é o nome e também a URL), que é um ambiente de desenvolvimento que roda direto no navegador e permite você usar 30 linguagens de programação diferentes. Você não vai usar para projetos grandes, mas é interessante para testar código e até mesmo conhecer outras linguagens.

Fonte: o autor.

## COMPILAÇÃO X INTERPRETAÇÃO

Não existe uma linguagem que seja melhor do que as outras, as linguagens costumam ser melhores em alguns aspectos e piores em outros. Existem vários aspectos que precisam ser analisados na hora de escolher uma linguagem, sendo o principal, o tipo de problema que queremos resolver.

Durante o processo de programação, existe uma parte onde o programador é o responsável e outra parte que fica por conta do computador. A parte do programador é, basicamente, a escrita do código-fonte, enquanto o computador é responsável por fazer esse código virar um programa executável.

O **código-fonte** é um arquivo em texto puro (*plain text*) que contém as instruções de acordo com a sintaxe de uma linguagem de programação. De acordo com a linguagem, esse código pode ser *compilado* ou *interpretado*.

Em linhas gerais, o processo de **compilação** consiste em transformar o código-fonte que você escreveu em um arquivo executável (aqueles arquivos *.exe*, caso você use Windows). Esse processo é utilizado por linguagens, como *C*, *Pascal*, *Java* e *Objective-C*. O processo de compilação está detalhado na imagem a seguir.

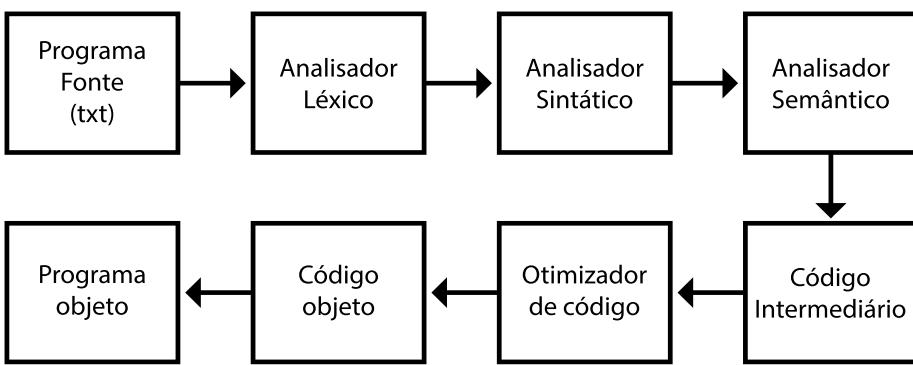


Figura: Processo de compilação de um programa

Fonte: o autor.

O processo de **interpretação** é similar, com a diferença que ele não gera um arquivo executável final, o código-fonte é interpretado toda vez que o sistema precisa ser executado. Pode parecer estranho e não-usual, mas é o processo utilizado por linguagens como *Python*, *JavaScript*, *PHP*, *Lua* e *Ruby*.

**SAIBA MAIS**



O processo de compilação de um programa é feito de acordo com a arquitetura onde o programa deve rodar. Programas que serão executados em Windows 32 e 64 bits, por exemplo, devem ser compilados para o Windows 32 e também compilados para Windows 64, o que resulta em arquivos executáveis diferentes.

A linguagem Java foi criada com o propósito de ser multiplataforma. Para isso, ela possui o conceito de máquina virtual. Os programas em Java, ainda, são compilados, mas não para um executável final. A compilação gera um arquivo que será executado pela máquina virtual. O mesmo código compilado pode ser executado em qualquer arquitetura, desde que possua a máquina virtual Java instalada.

Fonte: o autor.



## LICENÇAS DE SOFTWARE

No início da computação, o que realmente tinha valor era o hardware, o software era apenas “algo necessário” para o hardware funcionar. O preço dos computadores era definido, basicamente, pelo maquinário.

Esse conceito começou a mudar na década de 1980 com Bill Gates e a Microsoft, que foi criada, exclusivamente, para trabalhar com software e vender as suas soluções. Sua principal concorrente, a Apple, sempre trabalhou com hardware e software juntos, então, não estavam muito preocupados com isso.

A Microsoft conseguiu licenciar seus sistemas para a IBM, o que a fez crescer muito no início, e foi quem popularizou muito as licenças de software (aqueelas que você nunca lê e clica no botão “eu concordo com os termos da licença”).

O próprio UNIX era um software proprietário. Richard Stallman, que começou o movimento software livre, certa vez disse, como antes não existia essa noção de “propriedade” de software, as pessoas utilizavam, copiavam para quem precisava e só.

Estudaremos, aqui, algumas classificações de software em termos de suas licenças.

### SOFTWARE PROPRIETÁRIO

Iniciamos pelo software proprietário por já termos comentado sobre ele. Basicamente, é a forma mais comum: o criador do software é o seu “dono” e define os termos de como o software pode ser utilizado ou revendido.

Cada software pode ter os seus próprios termos de licença, por isso, é importante ler os termos com atenção.

## FREWARE/SHAREWARE

*Frewares* e *sharewares* são parecidos. Ambos são softwares proprietários, mas distribuídos de forma gratuita. A diferença é que o *freeware* (não confunda *freeware* com *free software*) é simplesmente distribuído de forma gratuita, enquanto o *shareware* é uma versão de demonstração, às vezes com limitação de funcionalidades e outras vezes com limitação de tempo de uso, que podem ser liberados mediante a compra da licença.

## SOFTWARE LIVRE E CÓDIGO ABERTO

Programas distribuídos como software livre ou código aberto não são programas sem licença, mas possuem uma licença que garante a qualquer um algumas liberdades, dentre elas, a de ter acesso ao código-fonte do programa.

Licenças de software livre e código aberto são bem parecidas, mas não são a mesma coisa. Um **software livre** garante o que são chamadas de “4 liberdades básicas”:

0. A liberdade de executar o software para qualquer uso.
1. A liberdade de estudar o funcionamento de um programa e de adaptá-lo às próprias necessidades.
2. A liberdade de redistribuir cópias do software.
3. A liberdade de melhorar o programa e de tornar as modificações públicas de modo que a comunidade inteira beneficie da melhoria.

As liberdades #1 e #3 exigem que haja acesso ao código-fonte. Na declaração original são descritas assim, começando em zero (Projeto GNU, 2018).

O **código aberto** é muito parecido, mas com o foco mais comercial e menos na filosofia. A ideia é basicamente a mesma, a *Open Source Initiative*, que mantém o movimento código aberto, garantindo que é a mesma coisa, apenas com o foco um pouco diferente e o código *open source* não restringe a licença de outros softwares distribuídos juntos.

Como os termos são muito parecidos, também, é comum encontrar o termo **FLOSS** (*Free/Libre and Open Source Software*), que é simplesmente uma junção de todas as definições em torno de software livre e *open source*, para que englobe os diferentes pontos de vista.



## DOMÍNIO PÚBLICO

Um software em domínio público é, simplesmente, um software que não possui autoria. Com isso, pode ser usado por qualquer um e para quaisquer fins. Na prática, o autor abre mão de qualquer forma de licença e autoria.

Era a forma mais comum de software nas décadas de 1970 e 1980, os softwares simplesmente “existiam”.

Hoje em dia, as licenças de software desempenham um papel muito importante, assim como os termos de uso e política de privacidade, que costumamos ver gerando alguns problemas e escândalos, mas que seriam um ótimo assunto para um outro momento.

## CONSIDERAÇÕES FINAIS

Assim, terminamos essa etapa em nossa jornada de conhecimento. Espero ter ajudado e despertado em você pelo menos uma pequena curiosidade, uma pequena chama, para que você continue sempre estudando e se atualizando.

Nossa área está em constante mudança, em constante evolução. Muita gente quer começar a estudar pelas últimas tecnologias e linguagens de programação. Você fez bem e veio aprender as bases, isso fortalece muito as estruturas do nosso conhecimento.

Nós vimos, nesta última unidade, como funcionam os dispositivos de entrada e saída, como se dá a comunicação deles com o processador e com a memória. O tempo todo em que estamos utilizando um computador estamos utilizando entrada e saída.

Depois, vimos para que servem os sistemas operacionais e quais são as suas funções, vimos que ele é o responsável por fazer os programas funcionarem e faz toda a parte de gerenciamento de hardware, fornecendo aos programas as funções que eles precisam.

Então, passamos aos softwares aplicativos, como são executados e como desenvolvê-los, com uma noção inicial de compilação e interpretação. Para quem está começando, logo vai aprender, nas aulas de algoritmos e de programação, sobre todo o processo, na prática.

Por fim, fechamos com as diferentes licenças de software, pois é importante para quem desenvolve e para quem usa saber como funcionam as licenças e permissões de uso de software. O mercado de software está bem aberto e é importante saber como funciona para conhecer, também, qual estratégia de licenciamento utilizar.

Vários dos assuntos vistos aqui você vai rever e se aprofundar em futuras disciplinas. A intenção é que você aprenda bem os fundamentos, para que eles abram as portas por toda a carga de conhecimento que vem pela frente.

Continue firme, continue estudando e se aprimorando, e seja o melhor que você puder em tudo o que você fizer!

## ATIVIDADES



1. Além da velocidade do processador e a velocidade de acesso à memória principal, as entradas e saídas de um dispositivo podem impactar no desempenho do sistema. Como as requisições de entrada e saída não possuem um momento certo para acontecer, há um mecanismo utilizado pelos processadores para tratar essas requisições.

Considerando o texto anterior, assinale a afirmativa com o nome dado a esse mecanismo de tratamento das entradas e saídas em um processador.

- a) Eventos.
- b) Requisições.
- c) Interrupções.
- d) Módulo de E/S.
- e) Barramento de E/S.

2. Existem diferentes interfaces de comunicação utilizadas para entrada e saída de dados. Todas elas são projetadas para transmitir bits de dados, mas a forma de transmissão pode variar. Um dos tipos de interface transmite conjuntos de bits, geralmente 8 bits, de forma simultânea e eles devem chegar no destino, também, de forma simultânea.

Considerando o texto anterior, assinale o tipo de interface referenciada pelo texto.

- a) Interface USB.
- b) Interface COM.
- c) Interface serial.
- d) Interface paralela.
- e) Interface dinâmica.

## ATIVIDADES



3. "Chamamos de dispositivos de entrada e saída aos dispositivos encarregados de incorporar e extrair informação de um computador. Eles se enquadram dentro da denominada Arquitetura de Von Neumann, que explica as principais partes de um computador. (...) estes periféricos operam através de interrupções que fazem com que os processos executados sejam suspensos temporariamente. Assim, os dispositivos de entrada e saída enviam interrupções ao processador."

**Conceitos. Dispositivos de Entrada e Saída** - Conceito, o que é, significado. [On-line]. Disponível em: <<https://conceitos.com/dispositivos-de-entrada-e-saida/>>. Acesso em: 23 ago. 2018.

Considerando o texto, associe os dispositivos indicando se o determinado dispositivo é de entrada ou saída, coloque 1 para entrada e 2 para saída.

1. Entrada

2. Saída

(  ) Monitor    (  ) Mouse    (  ) Teclado    (  ) Impressora    (  ) Scanner

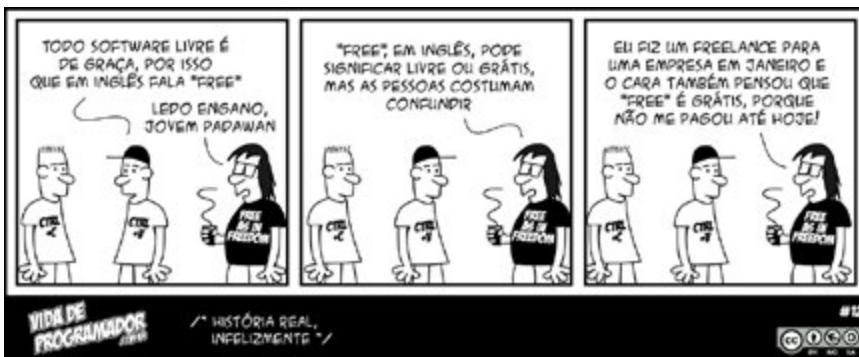
4. Um computador nunca sabe quando será gerado um evento de entrada ou saída. Um dispositivo de entrada (por exemplo, mouse ou teclado) pode ser acionado, ou não, a qualquer momento. O processador, constantemente, fica realizando operações de cálculos e trocas de dados, ele não pode perder tempo verificando a todo instante se há entrada ou saída. Para tratar a entrada e saída de dados, os processadores podem se utilizar de interrupções ou do *polling*.

Explique o que são interrupções e *polling*, deixando clara a diferença entre ambos.

## ATIVIDADES



5. Observe a tira:



Fonte: <[vidadeprogramador.com.br](http://vidadeprogramador.com.br)>.

Considerando a tira anterior, assinale a alternativa correta.

- Um software livre é um programa que é distribuído gratuitamente.
- Um software livre é um programa de computador que foi disponibilizado sem nenhum tipo de licença.
- Um software livre é um programa que foi disponibilizado gratuitamente, ou não, que contém o código-fonte e cuja licença permite a alteração do código-fonte, assim, como a redistribuição.
- Um software livre é um programa que foi disponibilizado gratuitamente, ou não, que contém o código-fonte e cuja licença permite o estudo do código-fonte, desde que não seja modificado.
- Um software livre é um programa que foi disponibilizado gratuitamente, ou não, que contém o código-fonte e cuja licença permite a alteração do código-fonte, desde que não seja redistribuído.

## ATIVIDADES



6. O desenvolvimento de sistemas para computadores modernos não é feito mais em linguagem de máquina, ou seja, usando diretamente as instruções do processador, a menos que seja necessário para algum fim específico. Para isso, utiliza-se linguagens de programação que possuem suas instruções que serão convertidas, em algum momento, para a linguagem de máquina, sem que o programador precise conhecer a linguagem de máquina especificamente.

Considerando o texto, assinale a afirmativa que contém qual é a camada que fica entre o programa desenvolvido e o hardware, realizando essa ponte entre as instruções.

- a) Software.
- b) Compilador.
- c) Montador assembly.
- d) Sistema operacional.
- e) Camada de aplicação.



## 5 linguagens de programação promissoras para se estudar

Que tal aproveitar esse início de ano e traçar uma meta de estudos envolvendo as linguagens mais promissoras para esse ano?

Se você é iniciante, é uma ótima maneira de começar para poder estar apto ao que o mercado está pedindo. Para você que já é da área, nunca é demais estudar e se aprimorar nas linguagens que você já conhece ou se aventurar em uma nova, pois sabemos que em nossa área o estudo é constante... então vamos começar?

### JavaScript

O JavaScript é uma linguagem que está praticamente em tudo! Esse já é um bom motivo para o qual você não estará perdendo tempo estudando ele. Vai chegar uma hora que você precisará saber nem que seja um pouco. Além disso, ele funciona tanto do lado do cliente como do servidor, assim você também poderá criar aplicativos e até executá-los em dispositivos IoT (Internet das coisas).

O JavaScript é uma linguagem de programação universal e continuará sendo. Caso você já domine ele, pode dar uma olhada no TypeScript, um superset de JavaScript.

### Python

O Python sempre foi uma linguagem popular para se aprender a programar, justamente por ter uma sintaxe bem clara e fácil de se aprender.

Além do Python ser uma ótima linguagem de programação para iniciantes, ela também se aplica para programadores mais experientes, pois é uma linguagem vastamente utilizada no mercado de trabalho.

O Python é uma linguagem que está em expansão principalmente na área de Machine Learning (com R), além de poder ser utilizada com vários propósitos. É possível desenvolver desde scripts até aplicações web e mobile.

### C#

O C# é uma das linguagens de programação mais modernas e populares hoje em dia, e esse ano não será diferente. Você pode utilizá-la para desenvolver para plataformas web, dispositivos móveis e aplicações desktop.

Com a praticidade dessa linguagem, você pode, de forma relativamente fácil, desenvolver desde projetos mais simples até projetos complexos e multiplataforma.

Além disso, o C# é usado em Unity, então se você deseja se aprofundar na área de games será realmente valioso, para que assim você possa se aventurar nas experiências de AR (Realidade Aumentada) e VR (Realidade Virtual).



## Go

O Go é uma linguagem que vem subindo rapidamente sua popularidade. É uma linguagem simples, mas muito poderosa. Ela foi criada pela Google com o objetivo de ser uma linguagem de programação ideal e você poderá se surpreender do tanto que essa linguagem pode atender às suas expectativas.

Se você já tiver uma base de conhecimentos em programação, se sentirá mais confortável em aprender Go.

## Kotlin com Android

O Kotlin é uma linguagem recente, onde você pode desenvolver aplicativos para Android com ela. Agora, a Kotlin é a linguagem oficial suportada na plataforma Android, se tornando uma substituição gradual ao Java.

Com o apoio do Google e JetBrains, não é surpresa que o Kotlin tenha tido um aumento tão grande em popularidade. Então se você deseja criar aplicativos para Android, já pode começar a estudar o Kotlin.

## Concluindo

Vale lembrar que essas são as linguagens que julgamos promissoras para este ano. Se você gosta e/ou precisa aprender alguma outra não há problema nenhum, o importante é o estudo constante.

Fonte: Guedes (2018, on-line)<sup>2</sup>.

# MATERIAL COMPLEMENTAR



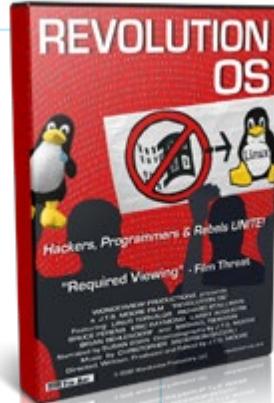
FILME

## Revolutions OS

Ano: 2001

**Sinopse:** com a participação de algumas das pessoas mais influentes da história como Richard Stallman, Eric Raymond e Linus Torvalds, o documentário começa no início da década de 80 com o projeto GNU, passando pela definição de software livre e código aberto, o papel do Apache na adoção inicial do Linux, a liberação do código-fonte do navegador Netscape para combater o monopólio do Internet Explorer da Microsoft, o IPO da Red Hat e a adoção do Linux e do software aberto de forma geral como modelo viável de negócios.

**Comentário:** vale muito a pena assistir esse documentário porque é feito com depoimento das próprias pessoas que fazem parte da história da computação e, mais especificamente, de software.



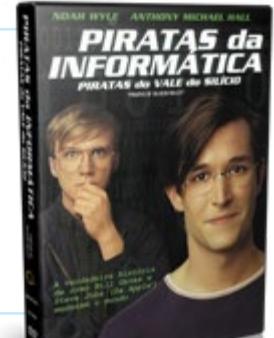
FILME

## Piratas da Informática (Pirates of Silicon Valley)

Ano: 1999

**Sinopse:** a ascensão da Apple e da Microsoft, as duas maiores empresas de informática do planeta. Em busca da liderança do mercado Steve Jobs (Noah Wyle) e Bill Gates (Anthony Michael Hall), fundadores das empresas, enfrentam-se em uma guerra de bastidores.

**Comentário:** um filme ótimo para entender o começo da era dos PCs, as parcerias, as intriga e as personalidades marcantes de Bill Gates e Steve Jobs.



# MATERIAL COMPLEMENTAR



FILME

## Sujeito a Termos e Condições (Terms and Conditions May Apply)

Ano: 2013

**Sinopse:** o que acontece quando o usuário clica em "aceitar" um contrato de termo de uso na internet? O documentário busca revelar o que vem sendo feito pelas corporações e os governos que possuem acesso ilimitado às informações dos usuários por meio de bancos de dados em computadores e celulares, permitidos de serem acessados ao se clicar "aceitar" em um termo de uso.

**Comentário:** em 2018 essa discussão de reacendeu com força após o escândalo do vazamento de dados do Facebook para a Cambridge Analytica e outras. A gente realmente não sabe o que estão fazendo com os nossos dados.



## REFERÊNCIAS

- CARTER, N. **Teoria e problemas de arquitetura de computadores**. Porto Alegre: Bookman, 2003.
- MONTEIRO, M. A. **Introdução à organização de computadores**. 4. ed. Rio de Janeiro: LTC, 2001. 498 p.
- MUHAMMAD, H.; DETSCH, A. **Uma nova proposta para a árvore de diretórios UNIX. III WSL - Workshop em Software Livre**, Porto Alegre, 2002.
- NULL, L.; LOBUR, J. **Princípios básicos de arquitetura e organização de computadores**. 2. ed. Porto Alegre: Bookman, 2011. 822 p.
- STALLINGS, W. **Arquitetura e organização de computadores**. 8. ed. São Paulo: Pearson Prentice Hall, 2010.
- TANENBAUM, A. S. **Organização estruturada de computadores**. 6. ed. São Paulo: Pearson Prentice Hall, 2013.
- WEBER, R. F. **Arquitetura de computadores pessoais**. 2. ed. Porto Alegre: Sagra Luzzatto, 2000. 272 p.
- WEBER, R. F. **Fundamentos de arquitetura de computadores**. 4. ed. Porto Alegre: Bookman, 2012. 424 p.

## REFERÊNCIAS ON-LINE

<sup>1</sup>Em:<<https://www.devmedia.com.br/como-funcionam-os-dispositivos-de-entra-da-e-saida/28275>>. Acesso em: 03 set. 2018.

<sup>2</sup>Em: <<https://www.treinaweb.com.br/blog/5-linguagens-de-programacao-promis-soras-para-se-estudar-em-2018/>>. Acesso em: 23 ago. 2018.



## **GABARITO**

1. C.
2. D.
3. 2, 1, 1, 2, 1.
4. As interrupções são um mecanismo usado pelos processadores onde um dispositivo de entrada, quando acionado, dispara um comando de interrupção para o processador, para que a entrada seja tratada no momento em que ocorre. O polling é uma alternativa na qual o processador fica verificando, sistematicamente, as entradas e saídas para ver se há solicitações a serem tratadas. É como se fosse uma trava de interrupções, na qual o processador fica verificando o tempo todo se há entrada ou saída.
5. C.
6. D.



# CONCLUSÃO

Meus parabéns por chegar até aqui, ao final de nossa jornada! E seria, talvez, um duplo parabéns: parabéns pelo conhecimento adquirido e parabéns pela perseverança de aguentar essas minhas linhas infindáveis.

Gostaria de dizer que agora você está pronto para qualquer desafio que vier pela frente, mas eu mesmo já disse algumas vezes neste livro que nossa área exige um estudo constante e que muitas coisas novas surgem a todo momento. Você pode ver a sua vida de estudos na área de T.I. como a aventura da Caverna do Dragão (referência vinda de um professor velho): sempre que você acha que vai acabar, não acaba! E você tem que continuar se esforçando e continuar aprendendo, lidando, também, com os problemas gerados pelos bugs (representados pelo unicórnio Uni).

Fico muito feliz de termos chegado até aqui. Aprendemos com as lendas do passado, quando estudamos a história da computação. Aprendemos que algumas dessas lendas nem são tão do passado assim, mas, ainda, estão conosco, escrevendo artigos, escrevendo tweets, palestrando em eventos...

Aprendemos a converter nossa informação em números e nossos números em binários, para guardar de uma forma mais coerente com a forma de operar dos computadores. Aprendemos a criar pequenos circuitos digitais, que já nos possibilitam a entender melhor os computadores e a criar pequenos componentes de automação para diversos usos.

Entendemos melhor como funcionam os processadores, a memória e os sistemas de entrada e saída, o que já vai nos possibilitar desenvolver sistemas bem mais gentis e respeitosos com os recursos computacionais.

E aprendemos bastante, também, sobre o software, desde os sistemas operacionais até os softwares aplicativos que você usa ou pode vir a desenvolver.

Você poderá se aprofundar mais nesses assuntos em outros estudos e outras disciplinas. Espero apenas duas coisas: que você tenha um futuro brilhante, seja qual for, e que a gente possa se encontrar pelo caminho.

Sucesso!



# **ANOTAÇÕES**



# **ANOTAÇÕES**



# ANOTAÇÕES

