



Interativa

Organização de Computadores

Autor: Prof. Alexandre Bozolan dos Santos

Colaboradores: Prof. Ricardo Sewaybriker
Profa. Christiane Mazur Doi

Professor conteudista: Alexandre Bozolan dos Santos

Doutor em Ciências, subárea de Engenharia Elétrica com ênfase em Telecomunicações, pela Escola de Engenharia de São Carlos da Universidade de São Paulo (EESC-USP), mestre em Engenharia Elétrica e Computação pela Universidade Presbiteriana Mackenzie (UPM), Engenheiro da Computação pelo Centro Universitário Braz Cubas, licenciado em Matemática pela Universidade Paulista (UNIP).

Atua como professor nos cursos de Tecnologia em Automação Industrial, Gestão em Análise de Sistemas e Gestão em Tecnologia da Informação na UNIP.

Possui mais de dez anos atuando como docente nos cursos de Engenharia Elétrica, Engenharia de Computação, Ciência da Computação e Sistemas de Informação. Atuou por mais de quinze anos como consultor tecnológico na área de redes de computadores e telecomunicações em diversas empresas do segmento de tecnologia da informação, além de pesquisador nas áreas de fotônica, trabalhando com *laser*, óptica não linear, nanomateriais, coloides, pontos quânticos, fibras ópticas microestruturadas, fibras ópticas especiais e sensores à fibra óptica.

Dados Internacionais de Catalogação na Publicação (CIP)

S237o Santos, Alexandre Bozolan dos.

Organização de Computadores / Alexandre Bozolan dos Santos.
– São Paulo: Editora Sol, 2021.

268 p., il.

Nota: este volume está publicado nos Cadernos de Estudos e Pesquisas da UNIP, Série Didática, ISSN 1517-9230.

1. Conversões. 2. Memória. 3. Barramento. I. Título.

CDU 681.3.02

U511.29 – 21

Prof. Dr. João Carlos Di Genio
Reitor

Prof. Fábio Romeu de Carvalho
Vice-Reitor de Planejamento, Administração e Finanças

Profa. Melânia Dalla Torre
Vice-Reitora de Unidades Universitárias

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Pós-Graduação e Pesquisa

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Graduação

Unip Interativa – EaD

Profa. Elisabete Brihy
Prof. Marcello Vannini
Prof. Dr. Luiz Felipe Scabar
Prof. Ivan Daliberto Frugoli

Material Didático – EaD

Comissão editorial:

Dra. Angélica L. Carlini (UNIP)
Dr. Ivan Dias da Motta (CESUMAR)
Dra. Kátia Mosorov Alonso (UFMT)

Apoio:

Profa. Cláudia Regina Baptista – EaD
Profa. Deise Alcantara Carreiro – Comissão de Qualificação e Avaliação de Cursos

Projeto gráfico:

Prof. Alexandre Ponzetto

Revisão:

Willians Calazans
Jaci Albuquerque
Lucas Ricardi

Sumário

Organização de Computadores

| | |
|--------------------|----|
| APRESENTAÇÃO | 11 |
| INTRODUÇÃO | 11 |

Unidade I

| | |
|--|----|
| 1 ORGANIZAÇÃO ESTRUTURADA DE COMPUTADORES | 13 |
| 1.1 Organização e arquitetura | 13 |
| 1.1.1 Diferenças básicas entre organização e arquitetura de computadores | 13 |
| 1.2 Função e estrutura | 14 |
| 1.2.1 Função | 14 |
| 1.2.2 Estrutura | 17 |
| 1.3 Máquinas multiníveis | 19 |
| 1.3.1 Máquinas multiníveis modernas | 19 |
| 1.3.2 Evolução das máquinas multiníveis | 21 |
| 1.4 Evolução dos computadores eletrônicos | 21 |
| 1.4.1 Primeira geração de computadores: relés e válvulas (1936-1953) | 22 |
| 1.4.2 Segunda geração de computadores: transistores (1954-1965) | 25 |
| 1.4.3 Terceira geração de computadores: circuitos integrados (1965-1980) | 26 |
| 1.4.4 Quarta geração de computadores: VLSI (1980-) | 27 |
| 1.5 Crescimento computacional e a lei de Moore | 28 |
| 1.6 Desenvolvimento dos sistemas operacionais | 29 |
| 2 SISTEMAS DE NUMERAÇÃO E A LÓGICA DIGITAL | 31 |
| 2.1 Conceito de base | 32 |
| 2.2 Conversão de bases numéricas | 32 |
| 2.2.1 Conversões de binário para decimal | 32 |
| 2.2.2 Conversões de decimal para binário | 34 |
| 2.2.3 Conversão de números hexadecimais em números decimais | 36 |
| 2.2.4 Conversão de números decimais em números hexadecimais | 37 |
| 2.2.5 Conversão de números hexadecimais em números binários | 38 |
| 2.2.6 Conversão de números binários em números hexadecimais | 39 |
| 2.3 Operações aritméticas em base binária | 39 |
| 2.3.1 Adição de números binários | 39 |
| 2.3.2 Subtração de números binários | 40 |
| 2.3.3 Multiplicação de números binários | 41 |
| 2.3.4 Divisão de números binários | 42 |
| 2.4 Circuitos lógicos digitais | 43 |
| 2.4.1 Variáveis booleanas e tabela-verdade | 44 |

| | |
|--|----|
| 2.4.2 Operação com porta lógica OU (OR)..... | 45 |
| 2.4.3 Operação com porta lógica E (AND) | 48 |
| 2.4.4 Operação com porta lógica NÃO (NOT) ou inversora..... | 50 |
| 2.4.5 Operação com porta lógica NÃO OU (NOR)..... | 51 |
| 2.4.6 Operação com porta lógica NÃO E (NAND) | 52 |
| 2.4.7 Operação com porta lógica OU EXCLUSIVO (XOR)..... | 53 |
| 2.4.8 Operação com porta lógica NÃO OU EXCLUSIVO (XNOR)..... | 54 |
| 2.4.9 Circuitos lógicos interconectados..... | 55 |

Unidade II

| | |
|---|-----|
| 3 ORGANIZAÇÃO DE SISTEMAS COMPUTACIONAIS..... | 61 |
| 3.1 Unidade central de processamento (CPU) | 61 |
| 3.1.1 Processo de fabricação da CPU | 61 |
| 3.1.2 <i>Chips</i> de CPU | 64 |
| 3.1.3 Intel Core i7..... | 66 |
| 3.1.4 Organização geral de um processador | 68 |
| 3.1.5 Microarquitetura de processadores..... | 68 |
| 3.1.6 Unidade lógica e aritmética (ULA) | 69 |
| 3.1.7 Unidade de controle (UC) | 70 |
| 3.2 Desempenho de operação do processador | 71 |
| 3.2.1 <i>Clock</i> (relógio) | 73 |
| 3.2.2 Taxa de execução de instruções por segundo..... | 75 |
| 3.3 Máquina de von Neumann | 76 |
| 3.3.1 Computador IAS..... | 76 |
| 3.3.2 Arquitetura Harvard..... | 79 |
| 3.4 Organização dos registradores..... | 80 |
| 3.4.1 Registradores de propósito geral ou "visíveis" ao usuário | 80 |
| 3.4.2 Registradores de controle e estado..... | 80 |
| 3.4.3 Organização de registradores em diferentes microprocessadores | 83 |
| 3.5 Arquitetura do processador Intel x86 e sua evolução | 85 |
| 4 CONJUNTO DE INSTRUÇÕES..... | 92 |
| 4.1 Ciclo de instrução..... | 92 |
| 4.2 Tipos de operandos..... | 93 |
| 4.2.1 Números..... | 93 |
| 4.2.2 Caracteres | 94 |
| 4.2.3 Dados lógicos | 95 |
| 4.3 Ciclo indireto..... | 95 |
| 4.4 Busca e execução de instruções..... | 96 |
| 4.5 Formatos de instrução | 97 |
| 4.5.1 Expansão de <i>opcodes</i> | 99 |
| 4.6 Instruções no Intel Core i7 | 101 |
| 4.7 Endereçamento de instruções..... | 102 |
| 4.7.1 Endereçamento imediato | 102 |
| 4.7.2 Endereçamento direto | 102 |

| | |
|--|-----|
| 4.7.3 Endereçamento indireto | 103 |
| 4.7.4 Endereçamento de registradores..... | 103 |
| 4.7.5 Endereçamento indireto por registradores | 104 |
| 4.7.6 Endereçamento por deslocamento..... | 104 |
| 4.7.7 Endereçamento de pilha | 106 |
| 4.8 Interrupções | 107 |
| 4.8.1 Interrupções e o ciclo de instrução..... | 109 |
| 4.8.2 Interrupções múltiplas..... | 113 |
| 4.9 Pipeline de instruções..... | 115 |
| 4.9.1 Desempenho do <i>pipeline</i> | 121 |
| 4.9.2 Hazards de <i>pipeline</i> | 126 |
| 4.9.3 Previsão de desvio em <i>pipeline</i> | 128 |
| 4.10 Linguagem de montagem | 132 |

Unidade III

| | |
|---|-----|
| 5 ORGANIZAÇÃO DAS MEMÓRIAS | 142 |
| 5.1 Características básicas da memória..... | 142 |
| 5.1.1 Acesso à memória..... | 144 |
| 5.1.2 Endereços de memória | 146 |
| 5.1.3 Hierarquia de memória..... | 147 |
| 5.2 Memória <i>cache</i> | 149 |
| 5.2.1 <i>Cache</i> de dados e instruções..... | 150 |
| 5.2.2 Endereço de <i>cache</i> | 151 |
| 5.2.3 <i>Caches</i> associativas..... | 152 |
| 5.2.4 <i>Caches</i> com mapeamento direto | 153 |
| 5.3 Memória somente de leitura | 154 |
| 5.3.1 ROM programada por máscara | 155 |
| 5.3.2 PROM..... | 156 |
| 5.3.3 EPROM..... | 156 |
| 5.3.4 EEPROM e <i>flash</i> | 157 |
| 5.4 Memória RAM..... | 158 |
| 5.4.1 Memória DRAM e SRAM..... | 159 |
| 5.4.2 Endereço, conteúdo, armazenamento e posição na memória RAM..... | 161 |
| 5.4.3 Operação de leitura..... | 164 |
| 5.4.4 Operação de escrita | 165 |
| 5.4.5 DRAM síncrona | 166 |
| 5.4.6 DRAM Rambus..... | 168 |
| 5.4.7 DDR-SDRAM..... | 169 |
| 6 MEMÓRIAS SECUNDÁRIAS | 171 |
| 6.1 Disco rígido | 172 |
| 6.1.1 Organização e funcionamento dos discos rígidos..... | 172 |
| 6.1.2 Propriedades de funcionamento dos discos rígidos..... | 175 |
| 6.1.3 Cálculo de espaçamento e armazenamento em discos rígidos..... | 176 |
| 6.1.4 Desempenho em discos rígidos..... | 177 |

| | |
|---------------------------------------|-----|
| 6.2 RAID em discos rígidos..... | 178 |
| 6.2.1 RAID 0..... | 178 |
| 6.2.2 RAID 1..... | 179 |
| 6.2.3 RAID 2..... | 179 |
| 6.2.4 RAID 3..... | 180 |
| 6.2.5 RAID 4..... | 180 |
| 6.2.6 RAID 5..... | 180 |
| 6.2.7 Memória virtual..... | 181 |
| 6.3 Drive de estado sólido (SSD)..... | 182 |
| 6.4 Discos ópticos..... | 185 |
| 6.4.1 CD-ROM..... | 185 |
| 6.4.2 DVD..... | 186 |
| 6.4.3 Blu-ray..... | 187 |
| 6.5 Disquetes..... | 188 |
| 6.6 Fitas magnéticas..... | 189 |

Unidade IV

| | |
|---|-----|
| 7 BARRAMENTOS, TIPOS DE TRANSMISSÃO E DISPOSITIVOS DE ENTRADA/SAÍDA (E/S) | 195 |
| 7.1 Barramentos do computador | 195 |
| 7.2 Largura de barramento | 196 |
| 7.3 Barramentos síncronos | 198 |
| 7.4 Barramentos assíncronos..... | 199 |
| 7.5 Barramento ISA..... | 200 |
| 7.6 Barramento PCI..... | 201 |
| 7.7 Barramento AGP..... | 202 |
| 7.8 Barramento PCI Express..... | 203 |
| 7.9 Barramento serial universal USB | 204 |
| 7.10 Tipos de transmissão | 205 |
| 7.10.1 Transmissão serial..... | 205 |
| 7.10.2 Transmissão paralela | 207 |
| 7.11 Módulos de E/S..... | 208 |
| 7.12 Interfaces e dispositivos de E/S..... | 210 |
| 7.12.1 Teclado..... | 211 |
| 7.12.2 Mouse | 212 |
| 7.12.3 Impressoras matriciais, jato de tinta e a laser..... | 213 |
| 7.12.4 Monitores de vídeo..... | 217 |
| 8 ARQUITETURA RISC, PROCESSADORES SUPERESCALARES, MULTITHREADING E MULTICORE..... | 219 |
| 8.1 Arquitetura RISC..... | 219 |
| 8.1.1 Arquitetura RISC da IBM..... | 220 |
| 8.1.2 Pipeline na arquitetura RISC..... | 222 |

| | |
|---|-----|
| 8.2 Processadores superescalares e <i>superpipeline</i> | 224 |
| 8.2.1 Comparativo entre processadores superescalares e <i>superpipeline</i> | 226 |
| 8.2.2 Limitações | 228 |
| 8.3 <i>Multithreading</i> | 229 |
| 8.4 <i>Multicore</i> | 233 |

APRESENTAÇÃO

O objetivo desta disciplina é oferecer ao aluno conhecimentos básicos e ao mesmo tempo fundamentais para uma boa compreensão do funcionamento dos computadores em geral e/ou dos sistemas embarcados, por exemplo os dispositivos móveis, que possuem funcionamento equivalente.

A evolução dos computadores nos tem possibilitado um elevado nível de acessibilidade e interação com as máquinas, o que facilitou sua aplicação em diversas áreas da ciência e tecnologia, tornando nosso cotidiano muito mais prático e versátil. Devido a essa rápida evolução dos computadores, se faz necessário estudarmos um pouco mais essa máquina, além de poder conhecer toda sua trajetória evolutiva até chegarmos às máquinas atuais.


Há aproximadamente setenta anos, desde que os computadores eletromecânicos foram desenvolvidos, os rumos da história tecnológica do homem mudaram para sempre. Logicamente que, em seu início, os computadores ainda não tinham o pleno alcance de usabilidade como nos dias de hoje. Na verdade, os computadores não possuíam a capacidade de processamento sequer comparada a um celular mais simples atualmente. Hoje em dia, computadores estão em toda a parte, sejam eles de grande ou pequeno porte, em cima de uma mesa ou no bolso da calça, no pulso, nas televisões modernas, nas geladeiras etc., realizando todos os tipos de tarefas a que foram programados. Entre algumas dessas tarefas, podemos exemplificar as tarefas simples, que vão desde escrever/ler um *e-mail*, manipular fotografias, construir tabelas e gráficos, até tarefas mais complexas, como simular colisão de partículas subatômicas, efetuar cálculos meteorológicos, controlar voos, toda uma malha ferroviária, robôs em uma linha de produção etc.

INTRODUÇÃO

O novo profissional do século XXI deverá estar adaptado às diferentes funcionalidades apresentadas por um computador, inclusive saber utilizar essas novas tecnologias para sugerir outras novas que ainda não chegaram a nossas residências. Os profissionais da área de tecnologia da informação deverão se apropriar dos conhecimentos de funcionamento interno dos computadores para que assim possam também contribuir nesse desenvolvimento das novas tecnologias. Por fim, um conhecimento aprofundado da organização de computadores oferece ao profissional/aluno um pleno entendimento de como a relação *hardware/software* é implementada, e poderá, assim, embasar todas as suas implementações futuras na área de tecnologia.

Neste livro-texto, são abordados, primeiramente, a organização e a arquitetura básica dos computadores, além de sua evolução a partir do desenvolvimento do primeiro computador eletromecânico. Também são apresentados os sistemas de numeração e métodos de conversão em binário e hexadecimal, utilizados em sistemas computacionais, além do conceito de portas lógicas digitais, fundamentais para realizar o processamento e o armazenamento de dados.

São apresentados os temas referentes à organização dos sistemas computacionais no que se refere à arquitetura de um processador, e o conjunto de instruções que são executadas internamente em um computador.



São explanados os tipos e características das memórias primárias e secundárias de um computador, responsáveis por operacionalizar todo o armazenamento dos dados e instruções na máquina.

Finalmente, são tratados temas como os meios de transmissão (barramentos) de comunicação de dados e instruções nos computadores. Também são abordados meios de comunicação entre dispositivos de entrada e saída, além de métodos de processamento paralelo e máquinas com vários núcleos de processadores.

Esperamos que você tenha um ótimo aproveitamento deste material e se sinta motivado a ler e conhecer mais sobre a arquitetura de computadores.

Boa leitura!

Unidade I

1 ORGANIZAÇÃO ESTRUTURADA DE COMPUTADORES

Um computador pode ser definido como uma máquina constituída de partes eletrônicas e/ou eletromecânicas capazes de manipular e fornecer de forma sistemática informações para uma gama variada de aplicações. Computadores realizam o processamento de dados, que consiste em uma série de atividades ordenadas e que são realizadas a fim de produzir um arranjo determinado de dados a partir de outros manipulados anteriormente.

Um dado ou informação podem ser tratados como sinônimos, de forma que o dado também pode ser definido como a matéria-prima ou entrada original a ser processada. Já a informação é definida como o resultado ou saída do processamento desse dado (MONTEIRO, 2019). Todos os dados introduzidos em um computador ou mesmo a informação resultante do processamento de dados precisam ser interpretados pela máquina, para que ela possa processá-los de forma correta.

Em se tratando de informação computacional armazenável, a menor unidade de informação é o algarismo binário ou dígito binário, popularmente conhecido como *bit*, que se trata de uma contração das palavras inglesas *binary digit*. Os dígitos binários possuem somente dois valores: 0 e 1, representados geralmente em termos de volts. O *byte* (conjunto de 8 bits), implementado inicialmente pela IBM, foi a primeira padronização de ordenação para um grupo de *bits*, e é utilizado atualmente por todos os fabricantes de computadores.



Lembrete

Computadores realizam o processamento de dados, que consiste em uma série de atividades ordenadas e que são realizadas a fim de produzir um arranjo determinado de dados a partir de outros manipulados anteriormente.

1.1 Organização e arquitetura

1.1.1 Diferenças básicas entre organização e arquitetura de computadores

Fazem parte dos atributos de uma arquitetura de computadores o conjunto de instruções que uma máquina necessita para poder operar sob comandos, o número de *bits* utilizados para que seja realizada a representação de uma gama diversa de dados que um computador pode operar, por exemplo números ou caracteres, mecanismos de entrada e saída de dados e instruções, além de técnicas de endereçamento de memórias internas ou externas (STALLINGS, 2010).

Alguns atributos presentes em sistemas computacionais podem incluir *hardwares* acessíveis (visíveis) a um programador, por exemplo sinais de controle, interfaces entre o computador e seus periféricos e estrutura de memória utilizada no sistema.

Historicamente, não há uma distinção bem definida entre a arquitetura e a organização de computadores. Alguns fabricantes de computadores podem oferecer uma família de computadores (grupo de computadores tendo como base o uso de microprocessadores com características em comum) utilizando a mesma arquitetura, mas com diferenças em sua organização. Em uma classe de computadores denominada microcomputadores (computador de pequeno porte cujos dados e instruções são operados por um microprocessador), o relacionamento entre a arquitetura e organização de computadores é de fato praticamente indistinguível.



Observação

Arquitetura de computadores são atributos acessíveis a um programador de computador e que possuem impacto direto sobre a execução lógica em um programa de computador.

Organização de computadores são unidades operacionais pertencentes à arquitetura e suas interconexões.

1.2 Função e estrutura

A natureza baseada na hierarquia dos sistemas computacionais complexos, em que cada nível de hierarquia consiste em componentes de entrada e saída que se inter-relacionam com os níveis mais baixos de sua estrutura, é essencial em um projeto e descrição de um computador moderno. Dessa forma, cada nível hierárquico dependerá somente de uma caracterização simplificada e resumida do sistema local, ficando os demais níveis caracterizados separadamente, de modo que, quando um computador é projetado, essas características são analisadas em relação à sua estrutura e função.

1.2.1 Função

O funcionamento de um computador, apesar de parecer complexo, pode ser representado através de um diagrama contendo funções básicas, que são processamento de dados, armazenamento de dados, movimentação de dados e controle. A figura a seguir mostra uma visão funcional de um computador, com cada uma das funções citadas.

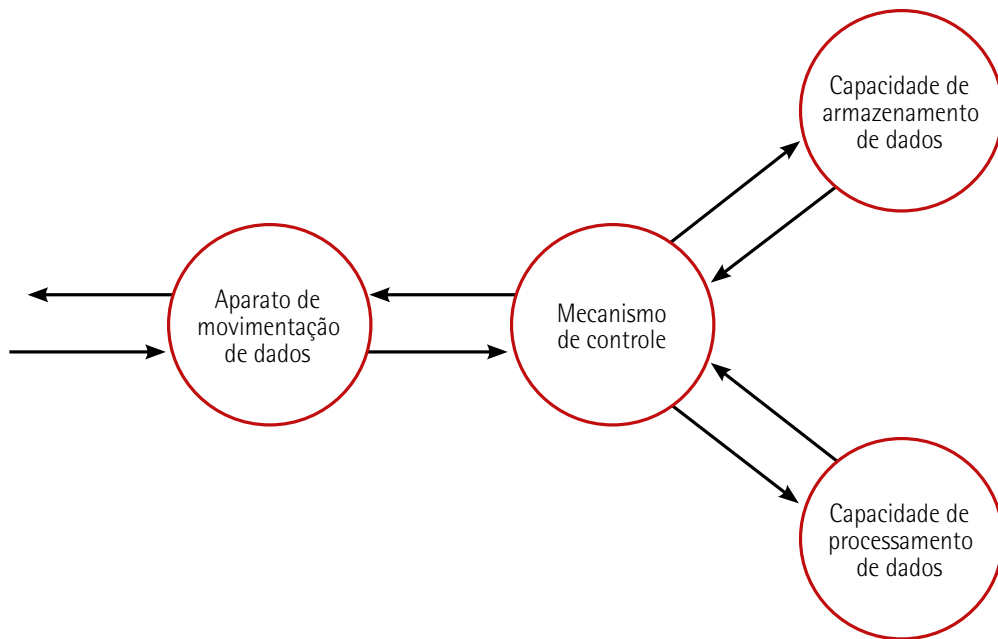


Figura 1 – Ambiente operacional

Na figura anterior, observa-se que o sistema necessita de uma interface física (aparato de movimentação de dados), constituído por dispositivos periféricos, que servirão de receptores/transmissores para os dados após a inserção de dados no sistema ou seu processamento. Na sequência, o mecanismo de controle indica qual tipo de operação o dado requererá, seja o processamento de dados, seja o armazenamento dos dados. Após o processamento de dados, estes são armazenados temporária ou definitivamente, e uma cópia desses dados é devolvida para o operador no ambiente externo à máquina pelos periféricos do computador, como monitor, impressora etc.

Processamento de dados

O processamento de dados é o processo que consiste na extração de informações de forma ordenada, resultando em uma combinação de novos resultados a serem utilizados por um sistema computacional. O processamento de dados é subdividido em três etapas: entrada, processamento e saída. Os dados processados geralmente estão contidos no próprio sistema de armazenamento, sendo necessária a movimentação desses dados para que seja realizado o processamento, como observado na figura a seguir.

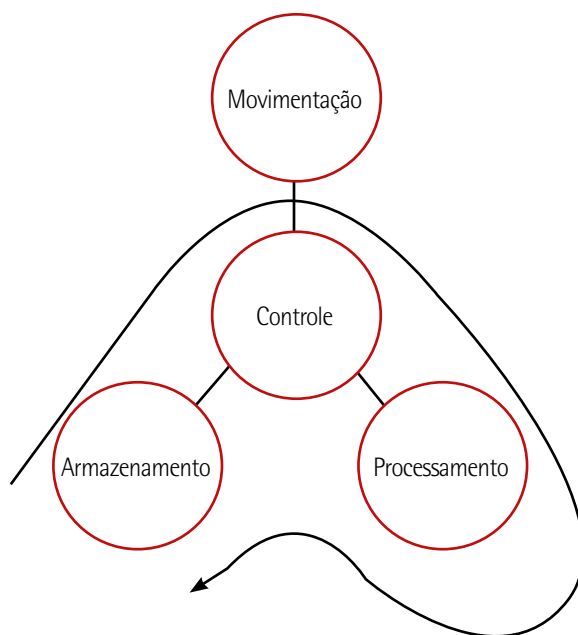


Figura 2 – Processamento de dados

Armazenamento de dados

Um computador deve ser capaz de armazenar dados mesmo que temporariamente, tornando o armazenamento de dados e instruções dinâmico. Dessa forma, os dados transferidos do ambiente externo podem ser armazenados no computador para a realização do processo de leitura/escrita, como mostra a figura a seguir.

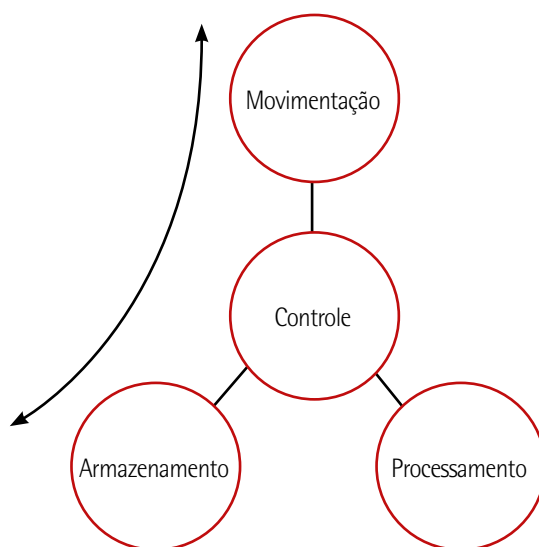


Figura 3 – Armazenamento de dados

Movimentação de dados

Um computador deve ser capaz de movimentar dados gerados internamente para dispositivos externos. Dessa forma, ele será capaz de receber dados externamente e processá-los internamente. Os dispositivos responsáveis pela interconexão com seu exterior (entrada/saída – E/S) são conhecidos como periféricos. Um computador pode funcionar como um dispositivo para movimentação de dados transferindo-os de um periférico para outro, como ilustra a figura a seguir.

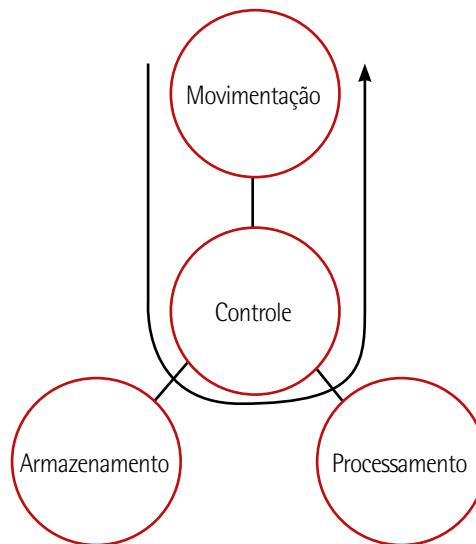


Figura 4 – Movimentação de dados

Controle

Controle diz respeito a operações realizadas pelo processador, fornecendo instruções para a realização das três funções descritas anteriormente (processamento de dados, armazenamento de dados e movimentação de dados). O gerenciamento dessas funções é realizado pela unidade de controle (UC), gerenciando os recursos computacionais através de instruções, além de coordenar o desempenho e funcionalidades do processador.

1.2.2 Estrutura

Um computador pode estar estruturado internamente a partir de quatro componentes principais responsáveis pelo seu pleno funcionamento, classificados como segue (STALLINGS, 2010):

- **Unidade central de processamento (UCP ou *central processing unit* – CPU):** constituída pela unidade lógica e aritmética (ULA) e pela unidade de controle (UC), tem como função controlar toda a operação do computador e realizar suas funções de processamento de dados e instruções, além de gerenciar o armazenamento de dados.
- **Memória principal:** tem a função de armazenar os dados de forma dinâmica e temporária.

- **Entrada e saída (E/S ou *input/output* – I/O):** tem como função mover os dados entre o computador e seu ambiente externo ou de um ambiente externo para seu processamento interior.
- **Interconexão do sistema:** é formada por mecanismos que oferecem um meio de comunicação entre a CPU, a memória principal e os dispositivos de E/S. Um exemplo comum de interconexão do sistema é por meio de um barramento de dados e instruções, que consiste em uma série de fios de cobre condutores aos quais todos os outros componentes se conectam para troca de dados. A figura a seguir ilustra uma visão ampliada dos dispositivos contidos na estrutura interna de um computador moderno.

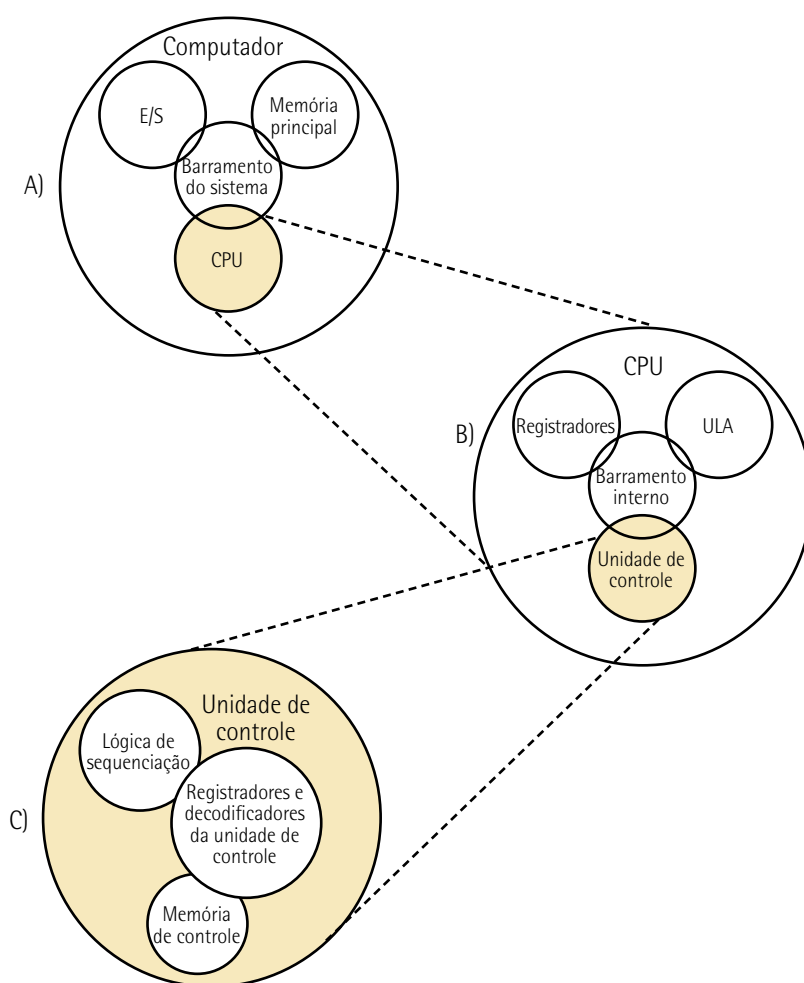


Figura 5 – Estrutura interna organizacional de um computador moderno

Conforme pode ser observado na figura anterior, a estrutura interna organizacional de um computador moderno é constituída principalmente por barramentos que fazem o elo entre os diferentes dispositivos e níveis de um computador. Na figura 5A, é possível observar o nível de dispositivos internos do computador, como CPU, memória principal e dispositivos de E/S; na figura 5B, é possível ter uma visão ampliada da CPU, constituída pelos registradores, ULA, UC e o barramento interno, que faz a interconexão entre esses dispositivos; na figura 5C, um nível mais abaixo mostra uma visão ampliada da

UC, constituída por registradores e decodificadores da UC, memória de controle e dispositivos lógicos, que realizam a sequenciação de instruções.



Observação

A UC realiza todo o controle operacional da CPU, a ULA realiza a função de processamento de dados e os registradores oferecem armazenamento interno na CPU para alocação de dados e instruções.

1.3 Máquinas multiníveis

1.3.1 Máquinas multiníveis modernas

Os computadores modernos, ao longo de seu desenvolvimento, possuíram diversos estágios em que foram introduzidos diferentes níveis de abstração e organização para tratamento de dados e instruções. Na atualidade, são aceitos basicamente seis níveis, que vão do nível 0, também conhecido como nível lógico, até o nível 5, que é o nível mais alto, onde estão as linguagens de programação orientadas a problemas (TANBEMBAUM; AUSTIN, 2013). A figura a seguir mostra como são separados os multiníveis de computadores contemporâneos.

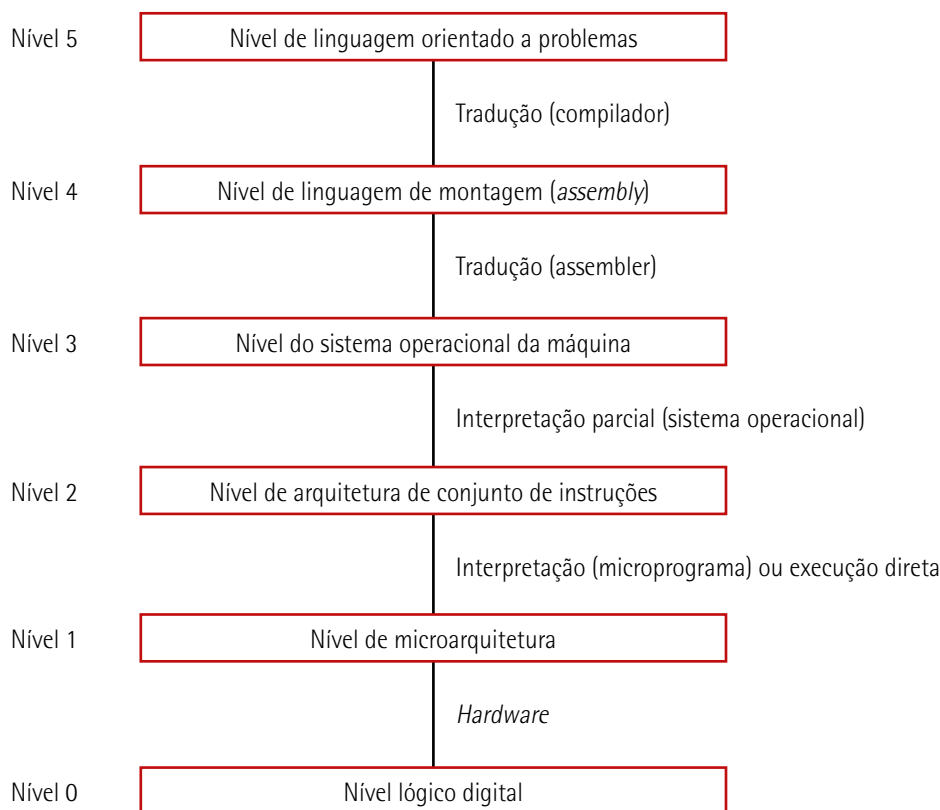


Figura 6 – Computador moderno separado em níveis

O nível mais baixo na estrutura (nível 0 ou nível lógico digital) é onde está contido todo o *hardware* da máquina. Nesse nível podem ser encontrados os dispositivos mais fundamentais para a operação de um computador, que são as portas lógicas ou *logic gates*. As portas lógicas são responsáveis principalmente pelo comportamento digital binário de um computador, tratando os dados apresentados a partir de estados eletrônicos 0 (desligado) e 1 (ligado). As portas lógicas são constituídas por elementos analógicos como transistores, que, combinados, resultam em um comportamento digital. Cada porta lógica é constituída por alguns transistores, e a combinação de várias portas lógicas forma circuitos digitais que podem operar sobre dados binários constituídos por 8, 16, 32, 64, 128 bits ou até mais.

O nível seguinte é o nível 1, conhecido como nível de microarquitetura, e ainda está na categoria de nível de *hardware*. No nível 1 também se encontra uma coleção de registradores, que são dispositivos de memória interna e estão contidos dentro da CPU, e são basicamente constituídos pelas portas lógicas. Nesse nível também se encontra a ULA, que é responsável pela organização lógica e aritmética (soma e subtração) do processador. Os registradores também servem como base para o transporte de dados e instruções para dentro e fora da ULA, a fim de que eles sejam processados. Geralmente, em computadores modernos a operação do fluxo dos dados é controlada por um programa denominado microprograma.

O nível 2 também é conhecido como nível de arquitetura do conjunto de instruções de máquina ou nível *instruction set architecture* (ISA). As empresas que fabricam e vendem computadores geralmente entregam junto com o computador um manual de referência da máquina com seus princípios práticos de funcionamento; nesse manual se encontra justamente o nível ISA e como as instruções são executadas e interpretadas pela máquina.

O seguinte é o nível 3, que é o nível de máquina do sistema operacional e que também costuma ser conhecido como nível híbrido, pois a maioria de suas instruções operacionais está contida no nível ISA, deixando claro que há uma mescla entre os níveis mais baixos, dificultando sua separação exata. Os acréscimos feitos nesse nível implicam a introdução de um interpretador, que realiza o controle dos *hardwares* contidos abaixo (nível 0 e nível 1), além de interpretar o conjunto de instruções produzidas no nível 2.

O nível 4 foi desenvolvido especialmente para que programadores de sistemas, ou seja, profissionais especializados no projeto e execução, possam ter acesso direto ao *hardware* das camadas abaixo, utilizando uma linguagem de baixo nível, como a linguagem Assembly.

O último nível (5) é conhecido como nível de linguagem orientada a objeto e trata dos *softwares* utilizados por programadores de aplicações, também conhecidas popularmente como linguagens de alto nível. Há uma gama extensa dessas linguagens, sendo algumas das mais conhecidas C, C++, C#, Java, Python, PHP, Perl etc., que possuem como função a compilação (interpretação) de instruções obtidas em alto nível (nível de usuário) convertendo-as em instruções de máquinas que possam ser interpretadas pelos níveis mais baixos (4, 3, 2, 1 e 0).

1.3.2 Evolução das máquinas multiníveis

Embora os seis níveis adotados atualmente como responsáveis pelo pleno funcionamento do processamento de dados de um computador, anteriormente a ideia de níveis sofreu algumas mutações necessárias para que essa separação fosse plenamente prática. Os programas escritos em linguagem de máquina (nível 1) podiam inicialmente ser executados diretamente pelos circuitos digitais encontrados no nível 0 (*hardware*), sem a necessidade de algum interpretador ou tradutor para realizar esse intercâmbio de dados. Entretanto, com o passar das décadas, os níveis mais elevados, que constituem basicamente do uso de *software*, foram cada vez mais responsáveis por fazer o intercâmbio das informações entre o *hardware* e o *software*, a fim de otimizar o processamento e armazenamento das informações.

Os primeiros computadores digitais desenvolvidos na década de 1940, por exemplo, tinham apenas dois níveis: nível 0 e nível 1, onde se realizava toda a programação em nível lógico digital. Durante a década de 1950, foi introduzido mais um nível nos projetos de computadores com o intuito de facilitar a inserção e interpretação de dados, baseada no uso de microprogramação, que executava programas do nível ISA através de interpretação. Os três níveis perduraram até a década de 1970, quando os demais níveis foram elaborados e aperfeiçoados, principalmente através do desenvolvimento dos sistemas operacionais (nível 3).

1.4 Evolução dos computadores eletrônicos

A evolução dos computadores é caracterizada pelo constante aumento na capacidade de processamento, diminuição de tamanho dos componentes eletrônicos, aumento na capacidade de memória e melhorias na comunicação e armazenamento de dispositivos de E/S.



Lembrete

A CPU é constituída por diferentes partes distintas em unidade de controle, responsável pela busca e decodificação de instruções e pela unidade lógica e aritmética, que efetua operações lógicas (AND, OR, NOT, NAND, NOR, XOR, XNOR), assim como operações aritméticas como adição e subtração.

Um dos fatores que contribui para o aumento do desempenho dos processadores é justamente o encolhimento nos componentes como os transistores, o que resulta na redução da distância entre componentes, possibilitando um real aumento de velocidade na comunicação de dados. Outro fator que vem contribuindo para a melhoria no desempenho dos processadores inclui o intenso uso do processamento paralelo com técnicas de *pipeline*, *superpipeline*, superescalares ou mesmo execução de predição de instruções e execução especulativa, que tentam prever a execução de instruções futuras.

Além dessas técnicas, um projeto deve conter um equilíbrio de desempenho entre os componentes, a fim de que o ganho final seja equilibrado e nenhum dispositivo envolvido no processamento de dados sofra algum tipo de atraso.

A evolução dos computadores modernos, iniciada nas décadas de 1930 e 1940, sintetizada no quadro a seguir, possibilita a visualização do desempenho (operações por segundo) em função da quantidade de meios necessários para realizar o processamento (relé, válvula, transistor).

Quadro 1 – Gerações dos computadores modernos

| Geração | Datas aproximadas | Tecnologia | Velocidade (operações por segundo) |
|---------|-------------------|-----------------------------------|------------------------------------|
| 1 | 1946-1957 | Válvula | 40.000 |
| 2 | 1958-1964 | Transistor | 200.000 |
| 3 | 1965-1971 | Integração em pequena escala | 1.000.000 |
| 4 | 1972-1977 | Integração em grande escala | 10.000.000 |
| 5 | 1978-1991 | Integração em escala muito grande | 100.000.000 |
| 6 | 1991-? | Integração em escala ultragrande | 1.000.000.000 |

Adaptado de: Stallings (2010, p. 20).

1.4.1 Primeira geração de computadores: relés e válvulas (1936-1953)

No início da década de 1930, o cientista alemão Konrad Zuse (1910-1995) desenvolveu um computador que ficou conhecido como Z1 e que foi concebido para ser uma continuação do projeto inicial do computador de Babbage (1791-1871). O Z1 era um computador programável e se baseava no uso de relés (figura a seguir) eletromecânicos para a realização de cálculos numéricos, em vez de engrenagens à manivela como proposto por Babbage.



Figura 7 – Relé eletromecânico

Além de possuir memória principal, o Z1 também continha uma ULA e uma UC. O Z1 foi construído na sala de estar da casa dos pais de Zuse durante o período da Segunda Guerra Mundial (1939-1945), e, embora possa parecer amador, era um computador bastante avançado para a época. Ele já utilizava um sistema de numeração na base binária (base 2), com uma programação limitada e instruções inseridas através de cartões perfurados (figura a seguir) utilizados para escrever as instruções e/ou dados.



Figura 8 – Cartões perfurados

O Z1 foi constituído basicamente por um somador/subtrator de 22 bits de ponto flutuante e uma unidade lógica de controle que possibilitava a realização de tarefas mais complexas, como multiplicação (por sucessivas somas) e divisão (por sucessivas subtrações). O único componente totalmente elétrico do Z1 era um motor, que servia para gerar um sinal de *clock* para sincronizar frequências de alguns *hertz*s. Esse computador eletromecânico representa um marco histórico, pois é considerado a primeira máquina binária programável do mundo.

Embora o Z1 não contivesse válvulas, ele foi considerado pertencente à primeira geração dos computadores, que também estavam sendo desenvolvidos tanto na Europa quanto no outro lado do oceano, mais precisamente nos Estados Unidos. Em 1946, os cientistas John Mauchly (1907-1980) e John Presper Eckert (1919-1995) desenvolveram o Electronic Numerical Integrator and Computer (Eniac – computador integrador numérico eletrônico), que foi conhecido como o primeiro computador totalmente eletrônico de uso geral, visualizado na figura a seguir.

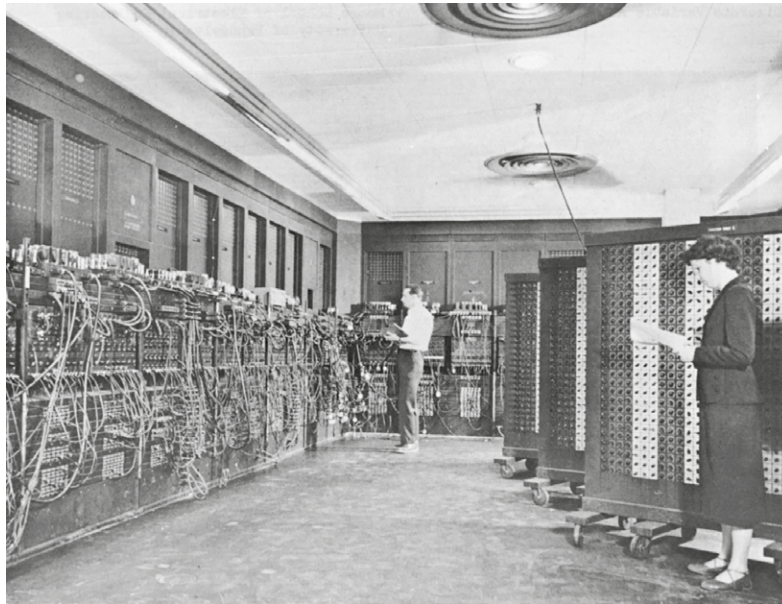


Figura 9 – Computador Eniac

Essa máquina ocupava um espaço de 1.800 metros quadrados, pesava aproximadamente 30 toneladas e consumia 174 kilowatts de energia elétrica. Diferentemente do Z1, o Eniac utilizava 17.468 válvulas (figura a seguir) para a realização das operações computacionais, além de memória com capacidade de cerca de 1.000 bits de informação, também armazenados em cartões perfurados (NULL; LOBUR, 2010).



Figura 10 – Válvula termiônica

Assim como o Z1, o Eniac foi desenvolvido para fins militares, principalmente no cálculo de trajetórias balísticas de seus artefatos. Antes disso, milhares de "computadores" humanos foram utilizados para esses cálculos aritméticos. Porém, com o desenvolvimento de máquinas capazes de reduzir o tempo de cálculo que antes era de 20 horas para apenas 30 segundos, o emprego de humanos nessa tarefa se tornou obsoleto, de forma que essa tendência perdura até os dias atuais.

1.4.2 Segunda geração de computadores: transistores (1954-1965)

O uso de válvulas durante a primeira geração de computadores não era prático do ponto de vista de engenharia. As válvulas superaqueciam e queimavam com muita facilidade, mais rápido do que poderiam ser repostas, de modo que os computadores passavam mais tempo em manutenção do que funcionando, tornando sua praticidade quase inviável. Diante desse problema, os cientistas John Bardeen (1908-1991), Walter Brattain (1902-1987) e William Shockley (1910-1989) foram os responsáveis pela invenção de um dos componentes eletrônicos mais revolucionários da história, o transistor (figura a seguir).

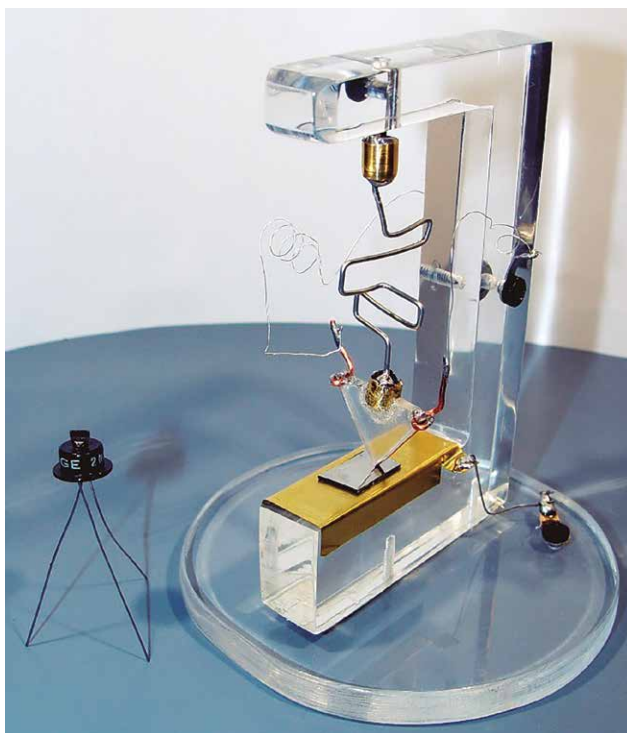


Figura 11 – Protótipo do primeiro transistor

O transistor é um componente eletrônico utilizado como amplificador ou interruptor de sinais elétricos, entre outras funções. A derivação da palavra *transistor* tem origem do inglês, *transfer resistor*, ou resistor de transferência, e opera basicamente permitindo ou impedindo a passagem de corrente elétrica. O transistor não revolucionou apenas a indústria dos computadores, mas também foi responsável pelo desenvolvimento de dispositivos como o rádio, a televisão, calculadoras de bolso, máquinas de lavar, geladeiras, micro-ondas, aviões etc. A utilização de transistores em eletroeletrônicos se tornou muito mais vantajosa, pois, além de o custo de fabricação ser menor, o consumo de energia elétrica era irrisório, principalmente se comparado às antigas válvulas termiônicas.

Atualmente, os transistores são utilizados em larga escala na indústria devido ao fato de que os fabricantes conseguem a cada ano diminuir cada vez mais seu tamanho, facilitando o aumento da quantidade desses componentes em um processamento, o que por consequência irá aumentar sua capacidade de realizar tarefas necessárias para seu funcionamento. Por exemplo, nos processadores atuais pode haver mais de um bilhão de transistores em uma única placa microprocessada, formando circuitos capazes de realizar cálculos extremamente complexos.



Observação

Os transistores modernos possuem apenas alguns nanômetros de diâmetro (1 nanômetro equivale a 1 milímetro dividido por 1 milhão), sendo que na 10ª geração de processadores Intel eles possuem apenas 10 nanômetros.

O primeiro computador transistorizado foi uma máquina de 16 bits construída nos laboratórios do Massachusetts Institute of Technology (MIT), denominado TX-0 (Transistorized Experimental Computer – computador transistorizado experimental).



Saiba mais

Leia mais sobre transistores em processadores em:

NOVO processador tem 100 milhões de transistores. *Inovação Tecnológica*, 4 dez. 2012. Disponível em: <https://bit.ly/2OqqMVr>. Acesso em: 1º fev. 2021.

1.4.3 Terceira geração de computadores: circuitos integrados (1965–1980)

O grande avanço no uso dos computadores ocorreu a partir da terceira geração, principalmente devido à construção do circuito integrado (CI) ou *microchip*, desenvolvido pelos cientistas Jack Kilby (1923-2005) e Robert Noyce (1927-1990), de forma independente. O CI nada mais é do que o empacotamento de vários transistores conectados, formando uma rede onde é possível controlar o comportamento da corrente elétrica de forma lógica, ou seja, a partir de alguns transistores são formadas as portas lógicas básicas (AND, OR, NOT, XOR, XNOR) responsáveis pelo processamento de dados e instruções no computador.

Um dos computadores de uso empresarial de maior sucesso desenvolvido na terceira geração foi o Mainframe IBM 7094 (figura a seguir), utilizado para processamento de aritmética binária de alto desempenho, baseado em registradores de 36 bits e processador robusto para tarefas de E/S.



Figura 12 – Mainframe IBM 7094 nos laboratórios da NASA

1.4.4 Quarta geração de computadores: VLSI (1980-)

Como observado, na terceira geração vários transistores foram integrados em um *chip* e, à medida que as técnicas de fabricação avançam, um número maior de transistores pode ser agrupado, possibilitando uma variedade de escalas de integração de circuitos eletrônicos.

Existem diversas categorias de integração de circuitos como o *small scale integration* (SSI – integração em pequena escala), em que é possível compactar cerca de 10 a 100 componentes por *chip*; o *medium scale integration* (MSI – integração em média escala), capaz de conter de 100 a 1.000 componentes por *chip*; o *large scale integration* (LSI – integração em larga escala), em que é possível acoplar de 1.000 a 10.000 componentes; e, por fim, o *very large scale integration* (VLSI – integração em escala muito larga), em que existem mais de 10.000 componentes por *chip* e que marca o início da quarta geração dos computadores (NULL; LOBUR, 2010). A tecnologia VLSI possibilitou o desenvolvimento do microcomputador ou *personal computer* (PC – computador pessoal), dispositivo suficientemente pequeno, barato (se comparado ao *mainframe*) e de fácil fabricação, tornando seu acesso facilitado ao público em geral, como o da figura a seguir, desenvolvido pela IBM em 1981.



Figura 13 – IBM PC modelo 5150



Saiba mais

Conheça mais sobre a evolução dos computadores modernos:

Disponível em: computerhistory.org. Acesso em: 8 abr. 2021.

1.5 Crescimento computacional e a lei de Moore

Atualmente, a indústria está em pleno avanço graças à evolução da capacidade de desempenho dos *chips* dos computadores e sistemas embarcados. O que tornou possível a melhora no desempenho foi a miniaturização dos transistores, que, em menor tamanho e maior quantidade, foram capazes de preencher a mesma área destinada aos cálculos lógicos e aritméticos que um processador da geração anterior havia deixado.



Observação

O fundador da Intel, o famoso cientista Gordon Moore (1929-), publicou em 1965 na *Electronic Magazine* um artigo científico impactante, em que, baseado em sua pesquisa, verificou que cada nova geração de *chips* de memória estava sendo lançada três anos após a geração anterior.

E se cada geração tivesse quatro vezes mais memória do que sua antecessora, o número de transistores do chip também cresceria a uma taxa constante, o que permitiu observar que esse crescimento continuaria estável pelas próximas décadas (TANENBAUM; AUSTIN, 2013). Essa pesquisa ficou conhecida como **lei de Moore**, e na atualidade costuma ser empregada para referenciar que o número de transistores aumenta a cada 18 meses, o que equivale a um aumento de 60% do número de transistores a cada ano, conforme pode ser observado na figura a seguir.

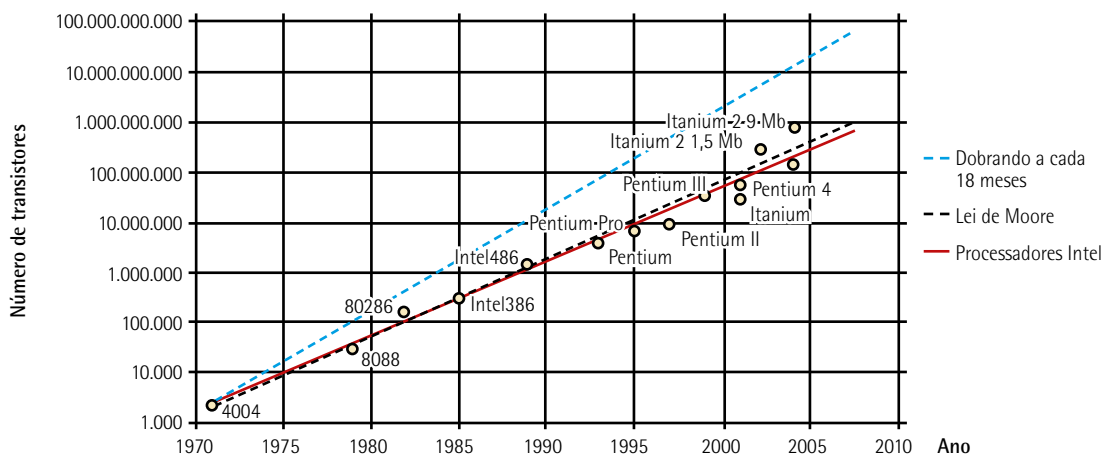


Figura 14 – Aumento do número de transistores em função do ano

A lei de Moore não se trata obviamente de uma lei real, mas apenas uma observação empírica sobre o quão rápido os cientistas e engenheiros avançam nessa área de desenvolvimento de processadores. Alguns cientistas afirmam esperar que a lei de Moore continue válida por pelo menos mais uma década ou mesmo mais tempo. Já outros cientistas acreditam que a dissipação de energia, fuga de corrente elétrica e outros efeitos indesejados comecem a aparecer, causando problemas mais sérios nos processadores.

Outro fator muito importante na constante diminuição dos transistores é que a espessura desses dispositivos consistirá em um nível atômico, impossibilitando, com o passar do tempo, maiores diminuições de tamanho, exigindo assim blocos de montagem subatômicos. Contudo, novas pesquisas com transistores estão sendo realizadas, incluindo os avanços em computação quântica, nanotubos de carbono e o grafeno, trazendo um novo patamar para o poder computacional atual.



Saiba mais

Conheça mais sobre computação quântica e informação quântica:

inctiq.if.ufrj.br

1.6 Desenvolvimento dos sistemas operacionais

Um sistema operacional é definido como um programa capaz de gerenciar todo o *hardware* de um computador (SILBERCHATZ; GALVIN; GAGNE, 2015). Um sistema operacional também pode ser definido como um ambiente de *software* que facilita, no âmbito do usuário, a inserção/extração de dados e instruções de um ambiente para outro (*software/hardware*).

A necessidade da criação dos sistemas operacionais surgiu quando o usuário precisava operar um computador nos anos 1960, e o único acesso para a máquina era através do *hardware*, além do que o volume de dados e instruções crescia de forma exponencial e era necessário um *software* para realizar esse gerenciamento de dados. Assim, durante a segunda geração de computadores (1955-1965), surgiram os primeiros *softwares* para gerenciamento de arquivos em lote (*batch*). Nesse período, computadores (*mainframes*) começaram a processar um volume muito grande de dados que eram inseridos através do uso de cartões perfurados. Os programas de computadores eram escritos nesses cartões perfurados e depois inseridos um a um para leitura e processamento. Porém, a inserção de dados manualmente e unitária de cada cartão perfurado se mostrou ineficaz. Logo, se mostrou necessário o uso de um lote de cartões perfurados para serem lidos e processados pelo computador, otimizando o tempo de execução das tarefas. A partir disso, notou-se a necessidade do desenvolvimento de algum *software* que realizasse o controle dessa tarefa, surgindo os primeiros sistemas operacionais em lote.

A figura a seguir mostra uma sequência de como o processo era realizado nesse período.

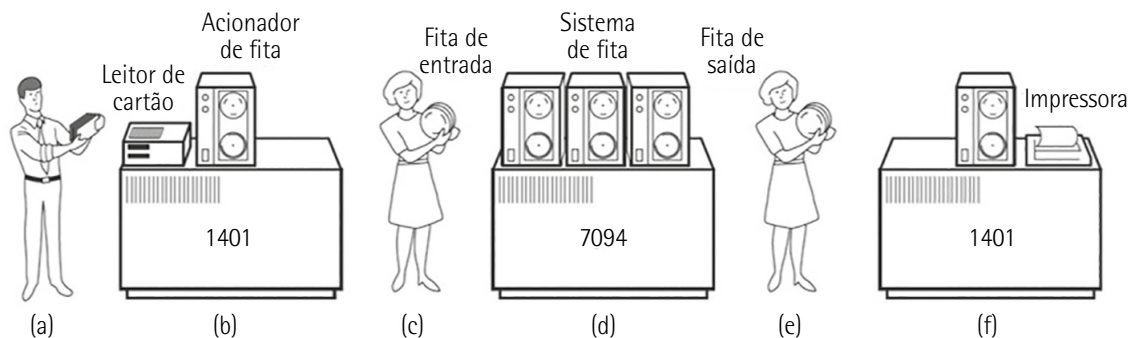


Figura 15 – Funcionamento de um sistema operacional em lote

No início do processo (figura 15A), um usuário é responsável pela inserção de um lote de cartões perfurados em um leitor de cartões que fazem parte do sistema de *mainframe* IBM 1401 (figura 15B). O processo continua com o *mainframe* 1401 realizando a operação de conversão dos dados contidos nos cartões para um sistema de fita magnética. Ao término da transferência, os dados gravados em fitas de entrada (figura 15C) agora são inseridos em outro *mainframe* (IBM 7094) (figura 15D), que de fato realiza o processamento do programa escrito inicialmente nos cartões perfurados. Após o término do processamento, os dados são armazenados em outro conjunto de fita magnética de saída (figura 15E), que serão lidos por outro *mainframe* 1401, sendo por fim realizada a impressão dos dados resultantes na saída (figura 15F).

Os primeiros sistemas operacionais de que se tem notícia baseavam-se na linguagem de programação Fortran e tinham como função a organização das instruções e dos dados de leitura, para que as tarefas (*jobs*) fossem realizadas em uma ordem correta. A figura a seguir ilustra em qual sequência as instruções eram realizadas em um lote de cartões perfurados.

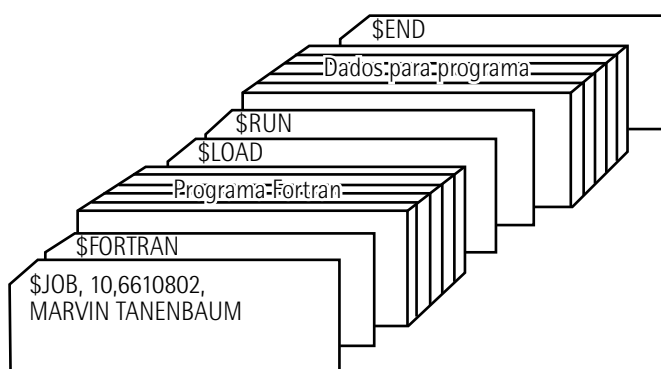


Figura 16 – Conjunto estruturado de cartões perfurados em lote

Inicialmente, um cartão denominado \$JOB contém informações como o tempo máximo para o processamento e os dados do programador. Na sequência, encontra-se o cartão \$FORTRAN, que determina ao sistema operacional o acionamento do compilador Fortran a partir de informações contidas em uma fita magnética. A próxima leitura era justamente do programa Fortran desenvolvido pelo programador contendo os dados a ser processados pelo sistema. O cartão \$LOAD ordenava ao sistema

operacional o carregamento do programa-objeto recém-compilado. Na sequência, o próximo cartão a ser lido era o \$RUN, que indicava ao sistema operacional que naquele momento deveria ser executado o programa constituído pelo conjunto de dados já ordenados nos cartões. Por fim, o cartão \$END indicava a conclusão das tarefas. Um dos sistemas operacionais mais populares a executar as tarefas de leitura e processamento de dados e instruções nesse período (1960-1970) foi o Fortran Monitor System (FMS) e o sistema operacional da IBM, IBSYS (TANENBAUM, 2010).



Saiba mais

Para conhecer um pouco mais sobre os sistemas operacionais, leia:

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Fundamentos de sistemas operacionais*. 9. ed. Rio de Janeiro, LTC, 2015. Capítulos 1 e 2.

2 SISTEMAS DE NUMERAÇÃO E A LÓGICA DIGITAL



Observação

O sistema de numeração decimal foi desenvolvido na Índia por volta de 600 a 300 a.C., sendo transcrito para a língua árabe pelo povo persa em meados de 900 d.C. e introduzido na Europa pelos povos bárbaros por volta do século X.

A evolução da grafia na representação dos números teve uma lenta modificação através dos anos, desde sua origem até os números que se utilizam no cotidiano atual, como mostra a figura a seguir.

| | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|---|---|
| Hindu 300 a.C. | - | = | ≡ | 𐌒 | 𐌔 | 𐌖 | 𐌘 | 𐌚 | 𐌜 | |
| Hindu 500 d.C. | 𐌐 | 𐌑 | 𐌒 | 𐌓 | 𐌔 | 𐌕 | 𐌖 | 𐌗 | 𐌘 | 𐌙 |
| Árabe 900 d.C. | 1 | 𐌐 | 𐌑 | 𐌒 | 𐌓 | 𐌔 | 𐌕 | 𐌖 | 𐌗 | 𐌘 |
| Árabe (Espanha) 1000 d.C. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
| Italiano 1400 d.C. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
| Atual | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |

Figura 17 – Representação dos números decimais através dos séculos

Os números decimais também podem ser representados através de um sistema posicional em que sua formação é definida pelo fato de que cada componente do algarismo pode ser diferente de acordo com a posição do número (MONTEIRO, 2019). O valor absoluto de um número é modificado pelo peso ou fator, que varia de acordo com a posição do algarismo, de modo crescente, iniciando-se da direita para a esquerda. Por exemplo, a representação do número em decimal 4244 é constituída de quatro algarismos, em que três deles possuem o valor absoluto 4, entretanto, devido à sua posição, eles possuem diferentes valores:

$$4244_{10} = 4000 + 200 + 40 + 4$$

$$4000 = 4 \times 10^3$$

$$200 = 2 \times 10^2$$

$$40 = 4 \times 10^1$$

$$4 = 4 \times 10^0$$

O peso modifica o valor do algarismo de acordo com sua posição, pois cada fator representa uma parcela da potência de 10 a partir do número 0, como mostrado no exemplo anterior, iniciando-se da direita. A numeração posicional se baseia nesse tipo de conceito e pode, a partir disso, realizar modificações nos números de acordo com a base utilizada.

2.1 Conceito de base

A necessidade da separação em bases numerais advém da ideia de grupamento de valores, permitindo a contagem e as operações aritméticas em qualquer base, utilizando uma simbologia.

O sistema decimal se baseia em dez símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. Números binários ou na base 2 possuem apenas dois símbolos, 0 e 1, enquanto o sistema octal (base 8) possui oito símbolos, 0, 1, 2, 3, 4, 5, 6 e 7.

Em sistemas de numeração maiores que 10, é necessária a criação de símbolos adicionais para a representação dos algarismos que não constam da base decimal. A base hexadecimal, por exemplo, utiliza, além dos números de 0 a 9, as letras do alfabeto em maiúsculo A, B, C, D, E e F para indicar os seis algarismos restantes após o número 9, completando o conjunto de algarismos com 16 símbolos da base hexadecimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

2.2 Conversão de bases numéricas

Ora entendido como os algarismos possuem várias maneiras de representação, é necessário adquirir o conhecimento de como se pode chegar a diferentes bases partindo de uma ou várias bases numéricas.

2.2.1 Conversões de binário para decimal

Cada dígito binário (*bit*) possui um certo peso, de acordo com sua posição relativa, que pode ser *less significant bit* (LSB – *bit* menos significativo) e *most significant bit* (MSB – *bit* mais significativo).

Assim, qualquer número binário poderá ser convertido em um número decimal equivalente somente somando-se os pesos das posições em que o número binário contiver o *bit* 1.

Exemplo de aplicação

Exemplo 1

A partir do número binário 11001110_2 , qual é seu equivalente em base decimal?

Resolução

Como a base em que o número se encontra é base 2, então deve-se organizar os números relativos à base 2 com a potência crescente da esquerda para a direita (LSB para MSB), iniciando-se com a potência zero: $2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0$. Esses valores, após efetuar a operação de potenciação, serão: 128, 64, 32, 16, 8, 4, 2 e 1. Representando isso em uma tabela, tem-se:

Tabela 1 – Conversão de base 2 para base 10 do número 11001110

| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

Agora, como mencionado, basta verificar na tabela quais valores em decimal (segunda linha da tabela anterior) com o valor em binário que se deseja converter (terceira linha da tabela anterior), somando-se os valores decimais encontrados como segue: $128 + 64 + 8 + 4 + 2 = 206$. Lembrando que nessa técnica de conversão deve-se associar somente os *bits* 1 com os valores em decimal, como foi exemplificado. Assim, a conversão de 11001110 em binário resulta em 206 decimal.

Exemplo 2

A partir do número binário 00110111_2 , qual é seu equivalente em base decimal?

Resolução

Tabela 2 – Conversão de base 2 para base 10 do número 00110111

| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

Assim: $32 + 16 + 4 + 2 + 1 = 55$.

Outro método bastante eficaz para realizar a conversão da base binária para a base decimal é o *double-dabble*.

Exemplo de aplicação

Dado o número em binário 11011_2 , encontre seu equivalente na base decimal.

Resolução

Primeiro, escreva de modo inverso o primeiro *bit* 1 mais à esquerda do valor e multiplique-o por 2. O resultado será $1 \times 2 = 2$. Em seguida, some o resultado (2) com o próximo valor em binário que se deseja converter (1). O resultado será $2 + 1 = 3$. Multiplique o resultado (3) por 2, obtendo-se $3 \times 2 = 6$. Some o resultado (6) com o próximo *bit* do número que se deseja converter (0), ou seja, $6 + 0 = 6$. Realize essas operações até que não haja mais nenhum *bit* a ser convertido.

Dado a ser convertido: 1 1 0 1 1

Resultado: $1 \times 2 = 2$

+1

$3 \times 2 = 6$

+0

$6 \times 2 = 12$

+1

$13 \times 2 = 26$

+1

27

Logo, o número 11011 em binário representa o número 27 na base decimal.

2.2.2 Conversões de decimal para binário

No sistema decimal, a conversão é realizada para a base binária utilizando o método de divisões sucessivas do número em decimal pela base em que se deseja converter o valor, no caso base 2, até que o valor não possa mais ser dividido por 2.

Exemplo de aplicação

Exemplo 1

Encontre o valor binário para o número em base decimal 25.

Resolução

$$\frac{25}{2} = 12 + \text{o resto } 1 \text{ (LSB)}$$

$$\frac{12}{2} = 6 + \text{o resto } 0$$

$$\frac{6}{2} = 3 + \text{o resto } 0$$

$$\frac{3}{2} = 1 + \text{o resto } 1$$

$$\frac{1}{2} = 0 + \text{o resto } 1 \text{ (MSB)}$$

Observe que nessa técnica o resultado em binário é obtido escrevendo-se o primeiro resto na posição designada por LSB e o último *bit* na posição do MSB. Assim, reconstituindo os valores dos restos encontrados do último para o primeiro, tem-se 11001.

Exemplo 2

Encontre o valor em binário para o número em decimal 37.

$$\frac{37}{2} = 18 + \text{o resto } 1 \text{ (LSB)}$$

$$\frac{18}{2} = 9 + \text{o resto } 0$$

$$\frac{9}{2} = 4 + \text{o resto } 1$$

$$\frac{4}{2} = 2 + \text{o resto } 0$$

$$\frac{2}{2} = 1 + \text{o resto } 0$$

$$\frac{1}{2} = 0 + \text{o resto } 1 \text{ (MSB)}$$

A técnica é a mesma utilizada no exemplo anterior. Assim, reconstituindo os valores dos restos encontrados do último (MSB) para o primeiro (LSB), tem-se 100101.

Outra técnica de conversão envolve subdividir o número que se deseja converter como uma soma de potências na base 2, de forma que os *bits* 0 e 1 serão atribuídos para que o valor em decimal correto se encaixe.

Exemplo de aplicação

Converta o número 76 na base decimal para seu respectivo valor em base binária.

Resolução

$76 = 64 + 0 + 0 + 8 + 4 + 0 + 0$ ou $26 + 0 + 0 + 23 + 22 + 0 + 0$. Atribui-se o *bit* 1 quando há um valor diferente de zero e atribui-se o *bit* 0 quando não há valor a ser atribuído em potência. Dessa forma, tem-se 1001100 em binário.

2.2.3 Conversão de números hexadecimais em números decimais

Diferentemente da base decimal, a base hexadecimal utiliza 16 símbolos como possíveis dígitos de conversão. Os dígitos estão entrepostos entre 0 e 9, além das letras do alfabeto A, B, C, D, E e F. As posições relativas aos dígitos recebem um peso cada uma em forma de potência de 16, por exemplo: 16^4 , 16^3 , 16^2 , 16^1 , 16^0 , e assim por diante. A tabela a seguir mostra algumas das relações de conversão entre valores em hexadecimal, decimal e binário. Um valor em hexadecimal poderá ser convertido em um equivalente decimal de acordo com um peso associado à potência de 16. O *bit* menos significativo (LSB) possui um peso $16^0 = 1$, seguido pela próxima posição, que possui o peso $16^1 = 16$, e assim por diante.



Observação

Os valores em hexadecimais são muito utilizados em linguagens de baixo nível como Assembly, além de estarem presentes em protocolos de comunicação de redes de computadores, como o protocolo IP versão 6 (IPv6).

Tabela 3 – Conversão de base 16 para base 10 e base 2

| Hexadecimal | Decimal | Binário |
|-------------|---------|---------|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| A | 10 | 1010 |
| B | 11 | 1011 |

| Hexadecimal | Decimal | Binário |
|-------------|---------|---------|
| C | 12 | 1100 |
| D | 13 | 1101 |
| E | 14 | 1110 |
| F | 15 | 1111 |

Adaptada de: Tanenbaum e Austin (2013).

Exemplo de aplicação

Converta o número hexadecimal 2AF em seu equivalente em base decimal.

Resolução

Inicialmente, coloca-se cada um dos componentes do número em hexadecimal multiplicando a potência de 16 associada e somando-se com o próximo número, começando a numeração da potenciação da esquerda para a direita: $(2 \times 16^2) + (10 \times 16^1) + (15 \times 16^0)$. Note que, no lugar de utilizar as letras A e F, utiliza-se seu equivalente em decimal, como escrito na tabela anterior. Em seguida, encontram-se os valores de potência e substituem-se na expressão, e por fim encontra-se o valor na base decimal: $(2 \times 256) + (10 \times 16) + (15 \times 1) = (512) + (160) + (15) = 687$.

2.2.4 Conversão de números decimais em números hexadecimais

A conversão de números na base 10 para a base 16 ocorrerá de forma semelhante à conversão de decimal para binário, ou seja, utilizando sucessivas divisões, porém agora utilizando a base 16. O exemplo a seguir ilustra o modo de conversão.

Exemplo de aplicação

Exemplo 1

Converta o número decimal 423 em seu equivalente na base hexadecimal.

Resolução

$$\frac{423}{16} = 26 + \text{o resto } 7 \text{ (LSB)}$$

$$\frac{26}{16} = 1 + \text{o resto } 10$$

$$\frac{1}{16} = 0 + \text{o resto } 1 \text{ (MSB)}$$

No exemplo, inicialmente se divide o número 423 em decimal pela base a qual se deseja encontrar o valor, no caso base 16. O valor encontrado na divisão é reutilizado em uma próxima divisão por 16, e seu resto será utilizado como o *bit* menos significativo (LSB). Segue-se com a divisão de 26 pela base 16, encontrando o valor 1, que é utilizado na próxima divisão, e o resto 10, que é utilizado na composição do valor em hexadecimal. Por fim, divide-se o valor 1 pela base 16, não sendo mais possível encontrar um resultado para essa divisão, e mantendo seu resto (1) na composição do número em hexadecimal resultante. Note que esse último valor a ser encontrado será considerado o mais importante, pois se trata do *bit* mais significativo (MSB). Logo, o valor final composto é: 1A7 na base hexadecimal, lembrando que o valor 10 deve ser automaticamente convertido para seu equivalente em hexadecimal, no caso a letra A.

Exemplo 2

Converta o número decimal 157 em seu equivalente na base hexadecimal.

Resolução

$$\frac{157}{16} = 9 + \text{o resto } 13 \text{ (LSB)}$$

$$\frac{9}{16} = 0 + \text{o resto } 9 \text{ (MSB)}$$

Como no exemplo anterior, realizam-se sucessivas divisões até que o número não possa mais ser dividido; então, considera-se seu resto para compor o número em hexadecimal: 9D.

2.2.5 Conversão de números hexadecimais em números binários

A conversão da base hexadecimal para base binária é conhecida como método compacto, e é relativamente simples. Nessa conversão, basta associar cada dígito em hexadecimal em seu equivalente binário de 4 bits encontrados na tabela anterior.

Exemplo de aplicação

Converta o número hexadecimal 9F2 em seu equivalente binário.

Resolução

Primeiramente, separam-se os números em hexadecimal individualmente e indica-se seu equivalente com 4 bits na tabela anterior:

$$9 \rightarrow 1001$$

$$F \rightarrow 1111$$

$$2 \rightarrow 0010$$

Depois, iniciando-se a composição do número da direita para a esquerda, ou seja, do MSB para o LSB, tem-se o número em binário: 100111110010.

2.2.6 Conversão de números binários em números hexadecimais

O processo de conversão da base binária para hexadecimal é basicamente o inverso do apresentado na conversão de base hexadecimal para binária, ou seja, é um processo simples, apenas utilizando a tabela anterior para compor cada equivalente na base necessária.

Exemplo de aplicação

Converta o número binário 1110100110 em seu equivalente hexadecimal.

Resolução

Para realizar essa conversão, é preciso primeiro separar os *bits* em agrupamentos de quatro, sempre da direita para a esquerda, da seguinte forma: 11 1010 0110.

Note que, ao agrupar os valores em grupos de 4 bits, os primeiros não possuirão *bits* necessários para compor o valor de forma correta; nessas situações, deve-se preencher os valores faltantes sempre com o zero à esquerda. Reescrevendo, tem-se: 0011 1010 0110. A seguir, deve-se verificar na tabela anterior o número equivalente em hexadecimal, resultando em: 3A6 em hexadecimal.

2.3 Operações aritméticas em base binária

Sistemas de computadores modernos se baseiam em operações capazes de armazenar, interpretar e manipular dados codificados em formato de números binários. Muitos sistemas possuem a capacidade de operar, de forma aritmética, os valores binários a fim de realizar as quatro operações básicas da matemática: adição, subtração, multiplicação e divisão.

2.3.1 Adição de números binários

A operação de soma utilizando dois ou mais números binários na base 2 é realizada de forma muito semelhante à soma em números decimais (base 10), porém lembrando que somente dois algarismos estão disponíveis na operação (0 e 1). Em binário, tem-se as seguintes situações:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ e "vai 1" para o próximo numerador, também conhecido como "transporte".}$$

Exemplos:

$$\begin{array}{r} \text{a) } 111 \\ +101 \\ \hline 1100 \end{array}$$

No exemplo (a), iniciando-se da coluna da direita para a esquerda, primeiro somam-se $1 + 1 = 0$ e "vai 1" para o próximo numerador à esquerda. Na coluna seguinte à esquerda, somam-se $1 + 0$ e também "vai 1" da operação anterior; logo, $1 + 1 + 0 = 0$ e "vai 1" para o próximo numerador à esquerda. Na terceira e última coluna à esquerda, somam-se o "vai 1" + $1 + 1 = 1$, e "vai 1" para o próximo numerador à esquerda, que nesse caso está vazio. Logo após esse "vai" chegar ao numerador, ele irá "cair", pois não há valores com quem somar, resultando em 1100.

$$\begin{array}{r} \text{b) } 1101 \\ +0100 \\ \hline 10001 \end{array}$$

No exemplo (b), também iniciando-se da coluna da direita para a esquerda, primeiro somam-se $1 + 0 = 1$. Na coluna à esquerda seguinte, somam-se $0 + 0 = 0$. Na terceira coluna à esquerda somam-se $1 + 1 = 0$ e "vai 1" ao denominador da próxima coluna à esquerda. Por fim, na quarta coluna à esquerda, somam-se o "vai 1" + $1 + 0 = 0$ e "vai 1" para o próximo numerador, que descenderá assim que chegar, pois não há mais números a serem somados, resultando no valor 10001. Da mesma maneira que nas operações em decimal, a soma é realizada pelos algarismos, e quando se somam 1 com 1, obtém-se a resultante 0 e sobra o valor 1, que vai ser retomado aos algarismos da parcela seguinte à esquerda ("vai 1"). Se os dois valores somados forem iguais a 1, ainda assim haverá o valor "vai 1" anterior que ainda será acrescentado; dessa forma, tem-se 1 com outro "vai 1" para o próximo algarismo à esquerda. Exemplo: $1 + 1 + 1 = 1$ e "vai 1".

2.3.2 Subtração de números binários

A subtração em números binários ocorre de forma semelhante ao sistema decimal, com: minuendo – subtraendo = diferença; entretanto, de forma um pouco mais complicada, pois se utilizam apenas dois algarismos para realizar as operações (0 e 1). As regras para realizar a subtração de binários são:

$$\begin{array}{l} 0 - 0 = 0 \\ 0 - 1 = 1 \text{ e "vai 1" para o próximo denominador.} \\ 1 - 0 = 1 \\ 1 - 1 = 0 \end{array}$$

Exemplos:

$$\begin{array}{r} \text{a) } 111 \\ -101 \\ \hline 010 \end{array}$$

No exemplo (a), iniciando-se da primeira coluna à esquerda, subtraem-se $1 - 1 = 0$. Na próxima coluna, subtraem-se $1 - 0 = 1$. E na última coluna à esquerda, subtraem-se $1 - 1 = 0$. Note que nesse exemplo não houve o "vai 1".

$$\begin{array}{r} \text{b) } 1100 \\ -0101 \\ \hline 0111 \end{array}$$

No exemplo (b), inicia-se a operação de subtração com $0 - 1 = 1$, e "vai 1" para o próximo denominador à esquerda. Logo em seguida, na próxima coluna à esquerda subtraem-se $0 - 1 - 0 = 1$. Nesse ponto, fica mais complexo o entendimento, pois, olhando para a expressão, tem-se apenas $0 - 0$, mas não se deve esquecer do "vai 1" da operação anterior, que foi incluído não no numerador, e sim acima do denominador, de forma que se tem: $0 - 1$ (referente ao "vai 1"), que resulta em 1 e novamente "vai 1". O 1 resultante dessa operação é subtraído de 0, ou seja, $1 - 0 = 1$. Na coluna seguinte à esquerda, tem-se $1 - 1 - 1 = 1$, ou seja, $1 - \text{"vai 1"}$, que veio da operação anterior, resultando em 0. Daí, esse 0 é subtraído de 1, ou seja, $0 - 1 = 1$, e "vai 1" para o próximo denominador à esquerda. Na última coluna à esquerda, tem-se $1 - 1 - 0 = 0$, em que $1 - \text{vai 1} = 0$, e finalmente $0 - 0 = 0$.

2.3.3 Multiplicação de números binários

O procedimento para a realização da multiplicação em números binários também é semelhante ao utilizado na base decimal, e possui as seguintes regras:

$$\begin{array}{l} 0 \times 0 = 0 \\ 0 \times 1 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \end{array}$$

Exemplos:

$$\begin{array}{r} \text{a) } 111 \\ \times 10 \\ \hline 000 \\ 111+ \\ \hline 1110 \end{array}$$

No exemplo (a), como na aritmética elementar, primeiro multiplica-se 0 por cada um dos valores do numerador, nesse caso o resultado foi 000. Depois, multiplica-se 1 por todos os valores individuais do numerador, também da esquerda para a direita, resultando em 111. Os valores 000 e 111 são somados na sequência, não se esquecendo de inserir o sinal + entre os valores do numerador e denominador, como mostra o exemplo. O resultado é 1110.

$$\begin{array}{r} \text{b) } 100 \\ \times 11 \\ \hline 100 \\ 100+ \\ \hline 1100 \end{array}$$

No exemplo (b), inicialmente multiplica-se 1 por todos os valores individuais do numerador (100) começando da esquerda para a direita, resultando no valor 100. Depois, multiplica-se novamente 1 por todos os valores do numerador, também resultando em 100. Esses dois valores (100 + 100) são somados, não se esquecendo da inserção do sinal de +, resultando no valor 1100.

2.3.4 Divisão de números binários

Na divisão de números binários, as seguintes considerações devem ser utilizadas:

$$\begin{array}{l} 0 \div 0 = 0 \\ 0 \div 1 = 0 \\ 1 \div 0 = \text{não existe solução.} \\ 1 \div 1 = 1 \end{array}$$

A forma para realizar a divisão pode seguir dois procedimentos, ou seja, pelo método comum, realiza-se dividendo / divisor = quociente e resto. Também é possível realizar a divisão por meio de sucessivas subtrações, o que se torna um processo mais simplificado para aplicações em circuitos digitais. Dessa forma, o quociente será obtido de acordo com a quantidade de vezes que o divisor poderá ser subtraído do dividendo, até que se obtenha o quociente igual a zero.

Exemplos:

$$\begin{array}{r} \text{a) } 100 \quad | \quad 10 \\ \quad 00 \quad 10 \end{array}$$

Primeiro escolhe-se o valor do quociente que, multiplicado pelo divisor (10), chegará ao valor mais próximo ou ao valor exato do dividendo. Como só é possível a escolha de dois algarismos (0 ou 1) para inserir no quociente, então se escolhe 1. Na sequência, multiplica-se 1 pelo divisor 10, resultando também no valor 10, que é colocado logo abaixo do dividendo. O dividendo é separado em seus dois primeiros termos (10) e subtraído pela resultante do quociente multiplicado pelo divisor que também foi 10, resultando em 0. Para finalizar a operação, o valor 0 à direita, restante do valor original 100 do dividendo, será multiplicado por 0 inserido no quociente e multiplicado pelo divisor, resultando num quociente 10.

$$\begin{array}{r} \text{b) } 1001 \quad | \quad 11 \\ \quad 11 \quad 11 \\ \quad \quad 0 \end{array}$$

De forma semelhante, no exemplo (b), inicia-se a operação escolhendo um número em binário para realizar a multiplicação do quociente pelo divisor. O número escolhido por razões óbvias é o 1, que será então multiplicado pelo divisor 11, resultando também em 11, que será subtraído dos três primeiros números do dividendo, escolhidos da direita para a esquerda (100). O resultado da subtração entre $100 - 11$ é igual a 001. Na sequência, desce o número 1 restante do dividendo, tornando agora o número 11. Esse resto 11 será o novo dividendo, então se insere o algarismo 1 no quociente para multiplicar com o divisor 11, resultando também em 11, e o resto será igual a 0.



Saiba mais

Para conhecer um pouco mais sobre conversão de bases numéricas, leia:

MONTEIRO, M. A. *Introdução à organização de computadores*. 5. ed. Rio de Janeiro: LTC, 2019. Capítulo 3.

2.4 Circuitos lógicos digitais

Em 1854, o matemático britânico George Boole escreveu um trabalho científico intitulado *An investigation of the laws of thought* (Uma investigação das leis do pensamento), em que era descrito como as tarefas poderiam ser tomadas com base em decisões lógicas como "verdadeiro" ou "falso" (TOCCI; WIDMER; MOSS, 2011). Sua teoria ficou conhecida até os dias atuais como **Teoria Booleana** e emprega um sistema baseado em símbolos ou operadores para descrever logicamente as tomadas de decisão. Da mesma forma que se utilizam símbolos algébricos, por exemplo x e y , assim também a Teoria Booleana se baseia no uso de símbolos para representar uma expressão lógica.



Observação

Cabe ressaltar que em álgebra não há um limite no uso de símbolos para expressar algum resultado, e na lógica booleana é comum utilizar dois valores possíveis, como: verdadeiro ou falso, aberto ou fechado, ligado ou desligado, aceso ou apagado, alto ou baixo etc.

Assim, fica também evidente que se tornaria um tanto quanto cansativo escrever expressões lógicas utilizando esses termos, embora corretos, e convencionou-se o uso de letras do alfabeto em maiúsculo, como: A, B, C etc. A lógica booleana não é somente utilizada como um instrumento de análise ou mesmo simplificação de sistemas lógicos, ela também é utilizada em projetos de circuitos digitais que necessitam de uma relação de entrada/saída de dados. Circuitos lógicos são utilizados para compor ou expor resultados de uma tabela-verdade, símbolos esquemáticos, diagramas de tempo e em linguagens de programação.

2.4.1 Variáveis booleanas e tabela-verdade

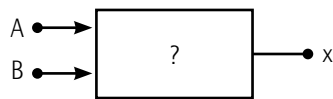
A principal diferença entre a álgebra convencional e a álgebra booleana é que, na booleana, as variáveis somente assumem dois valores possíveis, ou, no caso, 2 bits possíveis, 0 e 1.

Essas variáveis booleanas também são utilizadas na representação de níveis de tensão elétrica presente na conexão de circuitos digitais, muito utilizados em computação. Nesse tipo de lógica pode-se assumir um valor booleano 0 para representar alguma tensão baixa na faixa de 0 a 0,8 volts, enquanto se pode assumir o valor 1 para representar alguma tensão elétrica dentro da faixa de 2 a 5 volts. Dessa forma, os valores 0 e 1 não representam efetivamente números, mas sim um estado do nível de tensão elétrica que pode ser variável, e será chamado de nível lógico.

O sistema lógico da álgebra booleana se baseia em apenas três operações básicas: OU (OR), E (AND) e NÃO (NOT). Essas operações são conhecidas como operações lógicas, e os circuitos baseados nessas operações são denominados circuitos lógicos digitais.

As portas lógicas individuais são constituídas por diversos transistores, que produzem um comportamento lógico na saída que irá representar, justamente, as operações AND, OR ou NOT ou de suas variantes.

Também faz parte da constituição do projeto de circuitos lógicos uma tabela-verdade, que nada mais é do que uma técnica para descrever a saída de um circuito lógico dependente de suas entradas. A figura a seguir mostra uma tabela-verdade considerando um circuito lógico com duas entradas (A, B) e relaciona sua saída (X) com as possíveis combinações de acordo com o operador simbolizado pela incógnita (?).



| Entradas | | Saída |
|----------|---|-------|
| A | B | x |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Figura 18 – Tabela-verdade com duas entradas e sua respectiva porta lógica

A primeira linha da tabela-verdade da figura anterior resulta que, quando A e B possuírem o nível lógico 0, sua saída x será nível 1. Na segunda linha da tabela-verdade, quando a entrada A possuir nível lógico 0 e a entrada B possuir nível lógico 1, a saída x será 0. Na terceira linha, enquanto a entrada A é 1 e a outra entrada B é 0, a saída x será 1. Finalmente, na quarta linha, quando a entrada A é 1 e a entrada B também é 1, sua saída x será 0. A quantidade de linhas e colunas em uma tabela-verdade é determinada pela quantidade de entradas da expressão lógica. Por exemplo, a figura anterior possui duas entradas (A, B), logo, possuirá duas colunas; enquanto para determinar a quantidade de linhas basta elevar a quantidade de entradas pela base 2, ou seja, 2^2 , resultando em quatro linhas.

Note que esse exemplo é fictício e não foram levadas em consideração as operações lógicas AND, OR ou NOT para que o resultado tenha algum sentido real. A figura a seguir mostra outro exemplo de uma tabela-verdade, agora baseada em quatro entradas (A, B, C e D) e uma saída x, seguindo a mesma lógica da figura anterior. Note que o número de colunas corresponde ao número de entradas, no caso, quatro, e para saber a quantidade de linhas dessa tabela é só realizar a operação 2^4 , que resulta em 16 linhas.

| A | B | C | D | x |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Figura 19 – Tabela-verdade com quatro entradas



Lembrete

A lógica booleana também é utilizada em projetos de circuitos digitais que necessitam de uma relação de entrada/saída de dados.

2.4.2 Operação com porta lógica OU (OR)

A operação lógica OR é definida a partir da expressão booleana $x = A + B$, em que, nessa expressão, o sinal de + não representa uma adição convencional algébrica, mas sim a própria operação OR. A operação OR é descrita pela tabela-verdade de duas entradas (A, B) e uma saída (x) da figura a seguir.

OR

| A | B | $x = A + B$ |
|---|---|-------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Figura 20 – Tabela-verdade da operação lógica OR

Note que a expressão resultante de cada linha das operações OR $A + B$ resultará em um valor x , de acordo com a combinação das entradas. Em álgebra booleana, a soma de $1 + 1 = 1$, e não 2, pois em lógica booleana não é possível representar um número maior do que 1. A representação para o *bit* 1 é o nível (estado) ALTO, enquanto o *bit* 0 representa o nível (estado) BAIXO. Assim, de acordo com a tabela-verdade, lê-se:

$$\begin{aligned} 0 + 0 &= 0 & (0 \text{ OR } 0) \\ 0 + 1 &= 1 & (0 \text{ OR } 1) \\ 1 + 0 &= 1 & (1 \text{ OR } 0) \\ 1 + 1 &= 1 & (1 \text{ OR } 1) \end{aligned}$$

Em termos de lógica booleana, também é possível dizer que na operação OR, quando a entrada A estiver em nível BAIXO e B também estiver em nível BAIXO, a resultante x estará em nível BAIXO. Quando a entrada A estiver em nível BAIXO e a entrada B estiver em nível ALTO, a saída x resultante será igual ao nível ALTO. Circuitos digitais baseados em portas lógicas OR possuem, geralmente, duas ou mais entradas, cuja saída será igual à combinação das entradas através da operação OR, como mostra a figura a seguir.

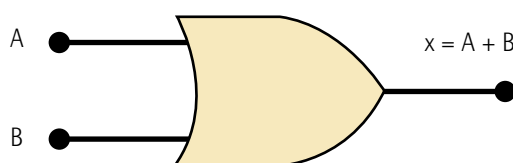


Figura 21 – Porta lógica OR

Em uma porta lógica com três entradas, como no exemplo da figura a seguir, a saída sempre será 1 quando ao menos uma das entradas ou ambas forem 1, e a saída será 0 somente quando todas as entradas forem 0.

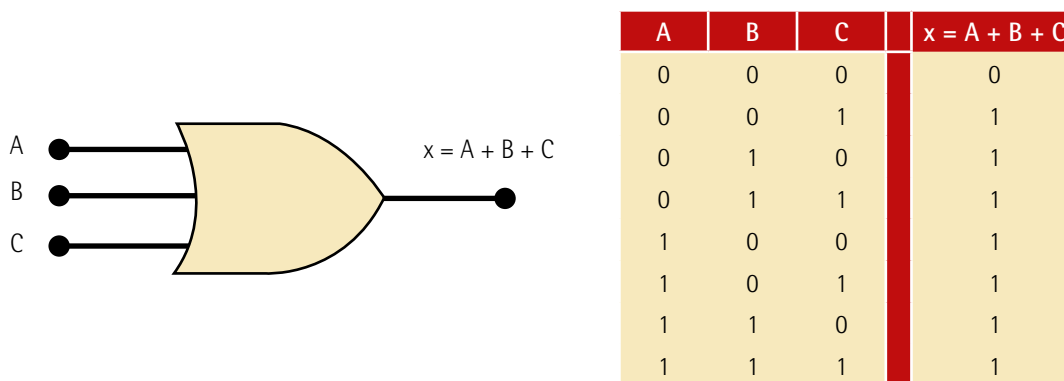


Figura 22 – Porta lógica OR de três entradas e sua respectiva tabela-verdade

A partir da explanação da porta lógica OR, é possível resolver o seguinte exemplo.

Exemplo de aplicação

No diagrama de tempo da figura a seguir, a entrada A tem seu início em nível BAIXO no instante t_0 e muda para o estado ALTO no instante t_1 , voltando para o estado BAIXO somente no instante t_3 , e assim alternando de forma sucessiva. A entrada B também se inicia em estado BAIXO no instante de tempo t_0 e assim permanece até o instante t_2 , alternando seu formato sucessivamente.

A partir dessas entradas, determine qual será sua saída, baseando-se no uso de uma porta lógica OR.

Note que na figura foram adicionados os *bits* 0 e 1 para representar as variações entre os níveis ALTO e BAIXO.

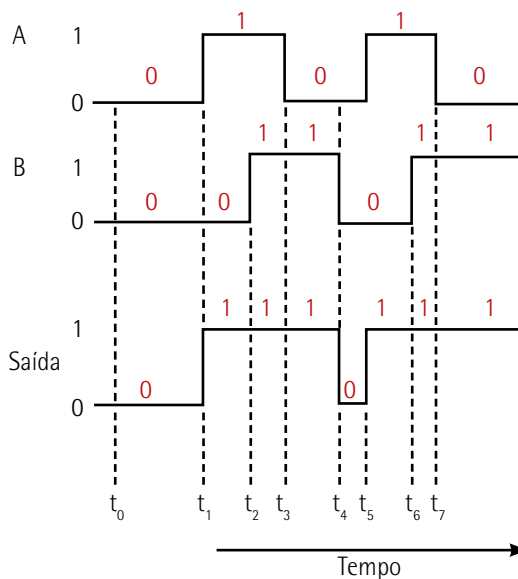


Figura 23 – Ondas digitais A e B e sua respectiva combinação de saída em função do tempo considerando o uso de uma porta OR

Resolução

Note que na figura anterior a saída da porta OR sempre terá nível ALTO para qualquer entrada também em nível ALTO. Por exemplo, nos instantes de tempo entre t1 e t2, t2 e t3, t3 e t4, t5 e t6, t6 e t7, as saídas terão nível ALTO, ou *bit* 1. Somente entre os tempos t4 e t5 é que a saída terá nível BAIXO ou *bit* 0.

2.4.3 Operação com porta lógica E (AND)

A operação lógica AND é definida a partir da expressão booleana $x = A \cdot B$, em que o sinal " " não representa uma multiplicação convencional algébrica, mas sim a operação AND. A operação AND é descrita pela tabela-verdade de duas entradas (A, B) e uma saída (x) na figura a seguir.

| AND | | | |
|-----|---|--|-----------------|
| A | B | | $x = A \cdot B$ |
| 0 | 0 | | 0 |
| 0 | 1 | | 0 |
| 1 | 0 | | 0 |
| 1 | 1 | | 1 |

Figura 24 – Tabela-verdade da operação lógica AND

Note que a expressão resultante de cada linha das operações AND $A \cdot B$ resultará em um certo valor x na saída, de acordo com a combinação das entradas. A representação para operação AND pode ser lida como:

$0 \cdot 0 = 0$ (0 AND 0)
 $0 \cdot 1 = 0$ (0 AND 1)
 $1 \cdot 0 = 0$ (1 AND 0)
 $1 \cdot 1 = 1$ (1 AND 1)

A contrário da operação OR, na operação AND somente quando as entradas estiverem todas em nível lógico ALTO (*bit* 1), a saída também estará em nível ALTO. Caso contrário, em qualquer combinação que possua alguma das entradas em nível BAIXO (*bit* 0), a saída também estará em nível BAIXO. Circuitos digitais baseados em portas lógicas AND possuem entre duas ou mais entradas, cuja saída será igual à combinação das entradas através da operação AND, como mostra a figura a seguir.

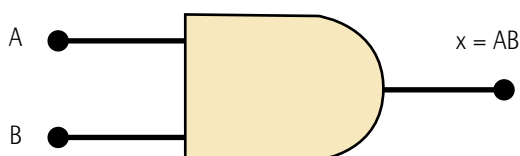


Figura 25 – Porta lógica AND

Em suma, a saída sempre será 1 quando todas as entradas também forem iguais ao *bit* 1, e 0 no caso contrário, como mostra a tabela-verdade da figura a seguir, para uma porta lógica AND com três entradas.

| A | B | C | $x = ABC$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

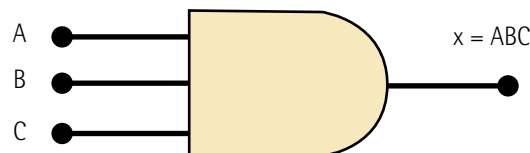


Figura 26 – Tabela-verdade referente a uma porta lógica AND com três entradas

A partir da explanação da porta lógica AND, também é possível determinar a saída x referente às duas entradas A e B, como mostra o diagrama de tempo da figura a seguir. Inicialmente, a entrada A está em nível BAIXO no instante t_0 e muda para o estado ALTO no instante t_1 , voltando para o estado BAIXO no instante t_3 , e assim sucessivamente. Já a entrada B está em nível BAIXO no instante t_0 , e assim permanece até o instante de tempo t_2 , quando muda para o estado ALTO, e assim varia até o final no instante t_7 .

Exemplo de aplicação

A partir do diagrama de ondas das entradas A e B, determine qual será sua saída, baseando-se no uso de uma porta lógica AND. Note que na figura foram adicionados os *bits* 0 e 1 para representar as variações entre os níveis ALTO e BAIXO.

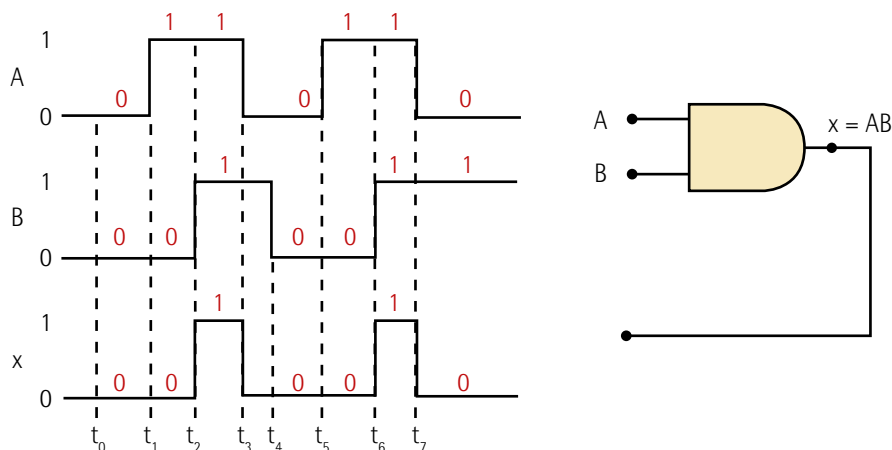


Figura 27 – Ondas digitais A e B e sua respectiva combinação de saída em função do tempo considerando o uso de uma porta AND

Resolução

Na figura anterior, a saída da porta AND de duas entradas somente estará em nível ALTO quando ambas as entradas estiverem também em nível ALTO, e a saída terá nível BAIXO para qualquer outra combinação. Por exemplo, somente nos instantes de tempo entre t_2 e t_3 , e t_6 e t_7 as saídas terão nível ALTO, ou *bit* 1, e todos os demais intervalos de tempo estarão em nível BAIXO, ou *bit* 0.

2.4.4 Operação com porta lógica NÃO (NOT) ou inversora

A operação lógica do tipo NOT, também denominada operação de complemento, inversão ou negação, é totalmente diferente das operações OR ou AND, devido ao fato de que ela pode ser realizada sobre apenas uma variável de entrada. A simbologia para a inversão de uma entrada é dada por: $x = \bar{A}$, em que \bar{A} (A barrado ou complemento de A) representa a operação inversa de A. Atribuindo-se *bits*, pode-se representar da seguinte forma:

$$A = 1 \text{ então } \bar{A} = 0$$

$$A = 0 \text{ então } \bar{A} = 1$$

Um circuito inversor possui um formato muito simples de ser representado, visto que é composto de apenas uma entrada e possui em sua representação um pequeno círculo em sua saída para denotar a operação de inversão, como mostra a figura a seguir.

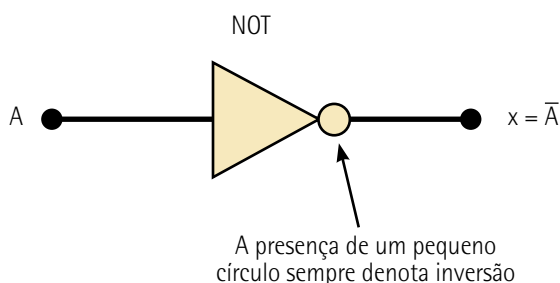


Figura 28 – Porta lógica NOT

A operação NOT é descrita pela tabela-verdade de uma entrada (A) e uma saída (x) da figura a seguir.

| NOT | |
|-----|---------------|
| A | $x = \bar{A}$ |
| 0 | 1 |
| 1 | 0 |

Figura 29 – Tabela-verdade da operação lógica NOT

A tabela-verdade da figura anterior só possuirá duas linhas, pois a tabela só contém uma entrada (A); logo, $2^1 = 2$, e suas respectivas saídas são exatamente o inverso dos valores do *bit* na entrada. A representação em formato de onda para uma porta NOT é mostrada na figura a seguir e indica o sinal de saída (x) com estado (nível) exatamente o oposto do estado de entrada (A).

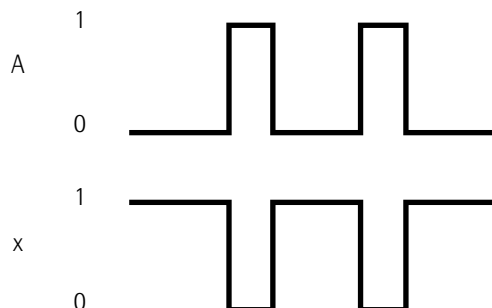


Figura 30 – Ondas digitais de entrada (A) e saída (x) em função do tempo considerando o uso de uma porta NOT

2.4.5 Operação com porta lógica NÃO OU (NOR)

A operação lógica NOR é equivalente às operações OR e NOT combinadas e é representada pela porta OR seguida de uma inversora, como mostra a figura a seguir.

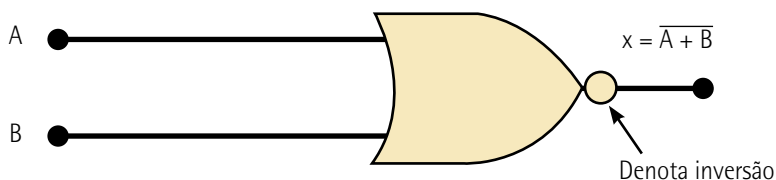


Figura 31 – Porta lógica NOR

Note que a expressão de saída para uma combinação de duas entradas (A, B) é dada por $x = \overline{A + B}$. A tabela-verdade da porta NOR possuirá a saída exatamente contrária à tabela-verdade da porta OR devido ao uso de uma inversora, como mostra a figura a seguir.

| A | B | OR | | NOR | |
|---|---|----|---------|-----|--------------------|
| | | | $A + B$ | | $\overline{A + B}$ |
| 0 | 0 | | 0 | | 1 |
| 0 | 1 | | 1 | | 0 |
| 1 | 0 | | 1 | | 0 |
| 1 | 1 | | 1 | | 0 |

Figura 32 – Tabela-verdade da operação lógica NOR

A saída $x = \overline{A + B}$ somente estará em nível lógico ALTO quando todas as entradas (A, B etc.) estiverem em nível lógico BAIXO, e em qualquer outra combinação de entradas a saída sempre estará em nível lógico BAIXO. A figura a seguir mostra como é uma combinação com duas entradas (A, B) e sua respectiva saída ao se utilizar uma porta lógica NOR na representação de ondas digitais. Note que, conforme há uma variação em função do tempo nas entradas, a saída também se modifica.

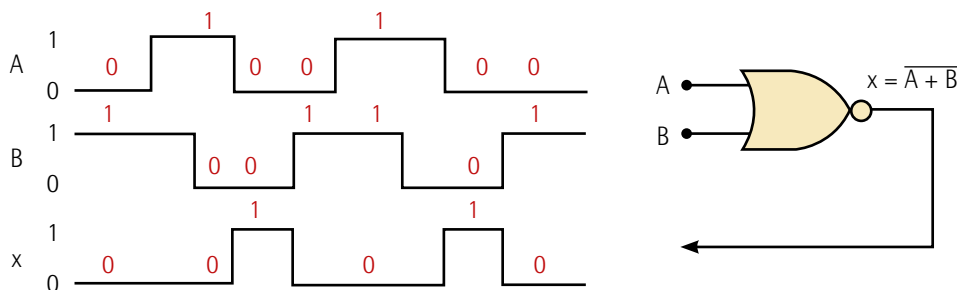


Figura 33 – Ondas digitais A e B e sua respectiva combinação de saída em função do tempo considerando o uso de uma porta NOR

Na figura anterior, a saída da porta NOR de duas entradas (A, B) somente estará em nível ALTO quando ambas as entradas estiverem em nível BAIXO, e a saída terá nível BAIXO para qualquer outra combinação.

2.4.6 Operação com porta lógica NÃO E (NAND)

A operação lógica NAND é equivalente às operações AND e NOT combinadas e é representada pela porta AND seguida de uma inversora, como mostra a figura a seguir.

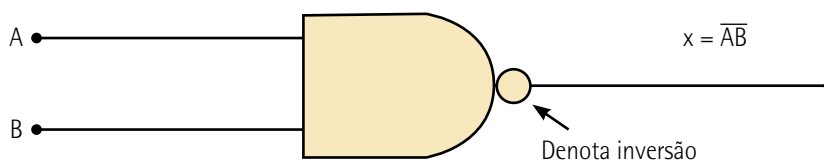


Figura 34 – Porta lógica NAND

Note que a expressão de saída para uma combinação de duas entradas (A, B) é dada por $x = \overline{A \cdot B}$. A tabela-verdade da porta NAND possuirá a saída exatamente contrária à tabela-verdade da porta AND, devido ao uso de uma inversora, como mostra a figura a seguir.

| | | AND | | NAND | |
|---|---|-----|----|------|----|
| A | B | | AB | | AB |
| 0 | 0 | | 0 | | 1 |
| 0 | 1 | | 0 | | 1 |
| 1 | 0 | | 0 | | 1 |
| 1 | 1 | | 1 | | 0 |

Figura 35 – Tabela-verdade da operação lógica NAND

A saída $x = \overline{A \cdot B}$ estará em nível lógico BAIXO somente quando todas as entradas (A, B etc.) estiverem em nível lógico ALTO, e em qualquer outra combinação de entradas a saída sempre estará em nível lógico ALTO. A figura a seguir mostra como se comporta uma combinação com duas entradas (A, B) e sua respectiva saída ao se utilizar uma porta lógica NAND na representação de ondas digitais.

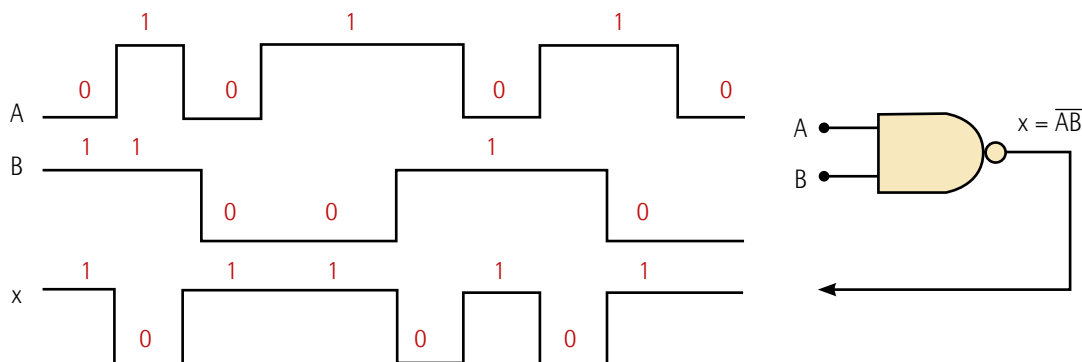


Figura 36 – Ondas digitais A e B e sua respectiva combinação de saída em função do tempo considerando o uso de uma porta NAND

Na figura anterior, a saída da porta NAND de duas entradas (A, B) somente estará em nível BAIXO quando ambas as entradas estiverem em nível ALTO, e a saída terá nível BAIXO para qualquer outra combinação.

2.4.7 Operação com porta lógica OU EXCLUSIVO (XOR)

A operação XOR, também conhecida como OU exclusivo (exclusive OR), é uma variação da porta lógica OR e é simbolizada por: $x = A \oplus B$, que corresponde à expressão lógica dada por: $x = \overline{A}B + A\overline{B}$. A figura a seguir mostra uma típica porta lógica XOR com duas entradas (A, B).

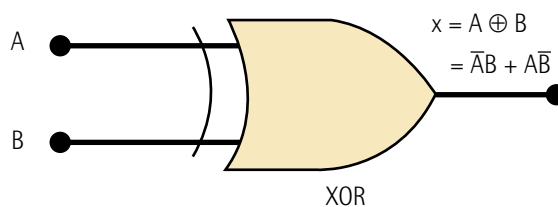


Figura 37 – Porta lógica XOR

A composição da tabela-verdade da porta XOR considerando duas entradas (A, B) é mostrada na figura a seguir.

| A | B | XOR $A \oplus B$ |
|---|---|---------------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Figura 38 – Tabela-verdade da operação lógica XOR

Note que para o caso da função lógica XOR a saída x terá nível BAIXO (*bit* 0) somente quando as entradas contiverem *bits* idênticos, por exemplo 0 e 0 ou 1 e 1, e saída com nível lógico ALTO para qualquer outra combinação com *bits* diferentes (0 e 1 ou 1 e 0).

2.4.8 Operação com porta lógica NÃO OU EXCLUSIVO (XNOR)

A operação XNOR é uma associação da porta lógica XOR adicionada a uma negação NOT e é simbolizada por: $x = \overline{A \oplus B}$, que corresponde à expressão lógica dada por: $x = \overline{AB} + \overline{AB}$. A figura a seguir mostra uma típica porta lógica XNOR com duas entradas (A, B).

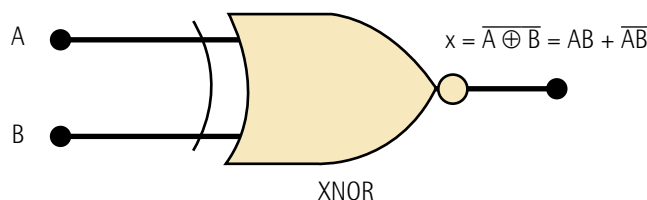


Figura 39 – Porta lógica XNOR

A tabela-verdade de uma porta XNOR, considerando duas entradas (A, B), é mostrada na figura a seguir.

| A | B | XNOR $A \oplus B$ |
|---|---|----------------------|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Figura 40 – Tabela-verdade da operação lógica XNOR

Note que, para o caso da função lógica XNOR, a saída x terá nível BAIXO (*bit* 0) somente quando as entradas contiverem *bits* diferentes, ou seja, quando contiver na entrada *bits* idênticos como 0 e 0 ou 1 e 1, a saída possuirá nível lógico ALTO.

2.4.9 Circuitos lógicos interconectados

Os circuitos lógicos podem ser compostos por várias portas digitais e representar seu comportamento de saída através das operações booleanas apresentadas anteriormente. Por exemplo, considere o circuito da figura a seguir, que possui três entradas (A, B, C) e uma saída x.

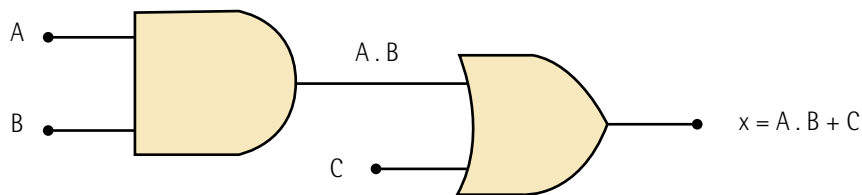


Figura 41 – Circuito lógico composto por duas portas digitais

A expressão lógica resultante na saída é descrita inicialmente através da operação AND entre as entradas A e B, resultando em $A \cdot B$, que, na sequência, é novamente combinada através de uma operação OR com uma nova entrada C, resultando na operação $x = (A \cdot B) + C$. Em um novo exemplo, descrito na figura a seguir, as entradas A e B se combinam em uma operação OR formando a primeira saída $A + B$, que é novamente recombina por uma nova entrada C, resultando na operação de saída $x = (A + B) \cdot C$.

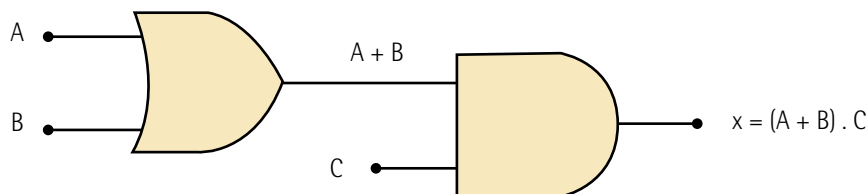


Figura 42 – Circuito lógico composto por duas portas digitais

A partir dos exemplos das figuras anteriores, é possível propor a solução para encontrar a expressão resultante em um circuito com mais componentes lógicos, como o da figura a seguir.

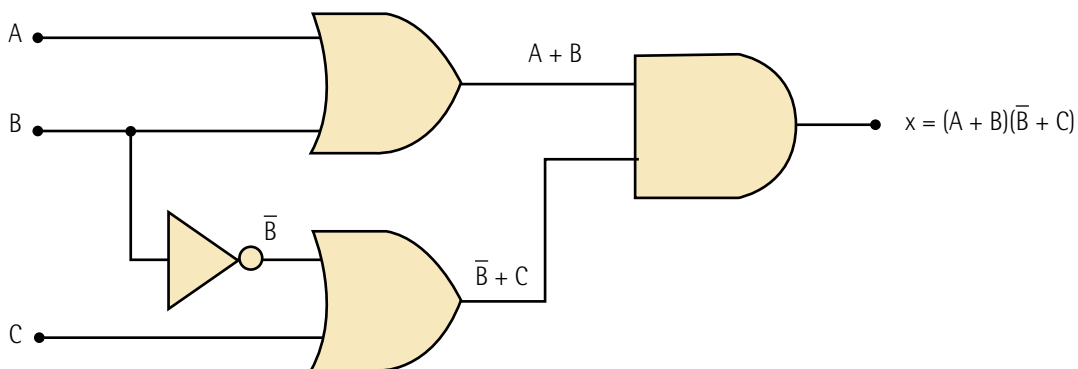


Figura 43 – Circuito lógico composto por quatro portas digitais

A solução para esse exemplo segue como nos casos anteriores, ou seja, deve-se inicialmente olhar o circuito e começar a localizar suas conexões sempre da esquerda para a direita. Inicialmente, há uma

combinação envolvendo a operação OR entre as entradas A e B, formando a expressão lógica $A + B$. Essa expressão serve de entrada para realizar uma combinação AND, e não poderá ser finalizada enquanto as entradas envolvidas na parte inferior da figura não estiverem completas. Na parte de baixo da figura, há primeiramente uma inversão lógica da entrada B, tornando-a \bar{B} , que, na sequência, é combinada em uma operação OR com a entrada C, formando a expressão de saída dessa operação $\bar{B} + C$. Após as operações na parte de cima da figura e na parte de baixo serem finalizadas em separado, o resultado das duas operações é combinado com a última porta lógica envolvida no processo (AND), resultando na expressão booleana: $x = (A + B) \cdot (\bar{B} + C)$.



Saiba mais

Para conhecer um pouco mais sobre portas lógicas, leia:

IDOETA, I. V.; CAPUANO, F. G. *Elementos de eletrônica digital*. 40. ed. São Paulo: Érica, 2011. Capítulo 2.



Resumo

Nesta unidade, foi possível entender como se deu o desenvolvimento dos primeiros computadores eletrônicos nas universidades e em departamentos de pesquisas em vários locais no mundo inteiro.

Vimos que um dos grandes impulsores para a criação dos computadores foi a grande dificuldade em realizar cálculos de forma rápida e prática, que somente era possível utilizando-se uma quantidade muito grande de pessoas (os computadores humanos).

Outro grande motivador ao desenvolvimento dos primeiros computadores foi seu uso para fins militares, a fim de calcular rotas balísticas.

Por fim, pudemos verificar que por trás do funcionamento dos computadores modernos são utilizados circuitos eletrônicos, formados por portas lógicas, baseadas em uma lógica binária que obedece à lógica booleana.



Exercícios

Questão 1. Um programador está trabalhando na manutenção de um programa que armazena informações em um arquivo com um formato proprietário e específico. Para investigar um problema com o funcionamento da aplicação, o programador resolve exibir o conteúdo do arquivo gravado no disco com um programa do sistema operacional GNU Linux chamado de "od". Com esse programa, é possível exibir o conteúdo de um arquivo de formato arbitrário de diversas formas. O programador opta por utilizar um formato de saída hexadecimal e obtém o seguinte resultado:

d2 04 00 00 d3 04 00 00

Após investigar o código do programa, foi identificado que esse arquivo armazena duas variáveis inteiras, sem sinais, com 4 bytes em cada uma delas. Considerando ainda a forma como esse computador armazena essas variáveis, o programador descobriu que os dois valores armazenados são (na notação hexadecimal):

000004d2

000004d3

Com base no cenário apresentado, avalie as afirmativas.

I – O valor da primeira variável, na notação em base binária e sem considerar nenhuma representação de sinal, é 10011010010.

II – O valor da primeira variável, na notação em base decimal, é 1234.

III – O valor da segunda variável, na notação em base binária e sem considerar nenhuma representação de sinal, é 10011010011.

IV – O valor da segunda variável, na notação em base decimal, é 1235.

É correto o que se afirma em:

A) I, apenas.

B) II, apenas.

C) III, apenas.

D) IV, apenas.

E) I, II, III e IV.

Resposta correta: alternativa E.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: podemos converter o número em hexadecimal para a base decimal e, depois, convertê-lo para a base binária. A conversão de hexadecimal para decimal pode ser feita conforme mostrado a seguir.

$$4 \times 16^2 + D \times 16^1 + 2 \times 16^0 = 4 \times 16^2 + 13 \times 16^1 + 2 \times 16^0 = 4 \times 256 + 13 \times 16 + 2$$

$$4 \times 16^2 + D \times 16^1 + 2 \times 16^0 = 4 \times 16^2 + 13 \times 16^1 + 2 \times 16^0 = 1024 + 208 + 2 = 1234$$

A conversão de decimal em binário pode ser facilmente feita se lembrarmos o que segue.

$$1234 = 1024 + 128 + 64 + 16 + 2 = 2^{10} + 2^7 + 2^6 + 2^4 + 2^1$$

Em binário, isso pode ser escrito conforme segue.

$$10000000000 + 10000000 + 1000000 + 10000 + 10 = 10011010010$$

II – Afirmativa correta.

Justificativa: podemos aproveitar o cálculo anterior, em que fizemos o que segue.

$$4 \times 16^2 + D \times 16^1 + 2 \times 16^0 = 4 \times 16^2 + 13 \times 16^1 + 2 \times 16^0 = 4 \times 256 + 13 \times 16 + 2$$

$$4 \times 16^2 + D \times 16^1 + 2 \times 16^0 = 4 \times 16^2 + 13 \times 16^1 + 2 \times 16^0 = 1024 + 208 + 2 = 1234$$

III – Afirmativa correta.

Justificativa: sabemos que o número em base hexadecimal é apenas uma unidade maior do que o valor calculado anteriormente. Assim, podemos simplesmente adicionar um em binário ao valor anterior. Com isso, obtemos o que segue.

$$10011010010 + 1 = 10011010011$$

IV – Afirmativa correta.

Justificativa: como sabemos que o valor é apenas uma unidade maior do que o valor calculado nas justificativas das afirmativas I e II, podemos calcular o valor decimal diretamente, fazendo $1234 + 1 = 1235$.

Questão 2. Um circuito digital bastante utilizado na computação e na eletrônica digital é o "meio somador" (em inglês, chamado de *half adder*). Ele é empregado para fazer a soma de 2 bits. É possível construir esse circuito com o uso de duas portas lógicas: uma porta XOR (OU exclusivo) e uma porta lógica AND (E lógico), como mostra a figura a seguir.

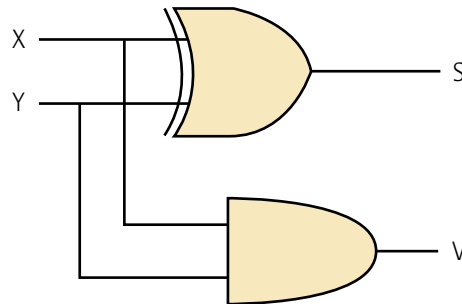


Figura 44 – Circuito digital de um meio somador

No circuito da figura, temos o seguinte:

- X e Y são os *bits* de entrada;
- S é a saída, que corresponde à soma dos *bits* de entrada;
- V é o sinal de "vai um" (em inglês, *carry out*) da operação de soma dos *bits*.

Com base nesse circuito e nos seus conhecimentos, avalie as afirmativas.

I – Caso X seja igual a um e Y também seja igual a um, a saída S será igual a zero e o valor de V será igual a um.

II – Como o valor de V é o resultado de uma operação lógica AND (E), tanto o valor de V quanto o valor de S devem ser iguais a zero nos dois seguintes casos: se X for zero e Y for igual a um (primeiro caso) ou se X for igual a um e Y for igual a zero (segundo caso).

III – Se X for igual a zero e Y for igual a zero, tanto S quanto V devem ser iguais a zero.

É correto o que se afirma em:

- A) I, apenas.
- B) II, apenas.
- C) III, apenas.
- D) I e III, apenas.
- E) I, II e III.

Resposta correta: alternativa D.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: sabemos que o valor de S é dado por:

$$S = X \text{ XOR } Y = XY' + X'Y = 10 + 01 = 0 + 0 = 0$$

Devemos lembrar que aqui o símbolo "+" representa a porta lógica OU e o símbolo de apóstrofo representa a negação (inversão do valor lógico).

O valor de V é $V = XY = 11 = 1$.

II – Afirmativa incorreta.

Justificativa: se X for igual a um e Y for igual a zero, ou se X for igual a zero e Y for igual a um, o valor de S será igual a um, pois:

$$S = X \text{ XOR } Y = 1 \text{ XOR } 0 = 0 \text{ XOR } 1 = 1$$

O valor de V será igual a zero, pois:

$$V = XY = 10 = 01 = 0$$

III – Afirmativa correta.

Justificativa: basta fazermos o seguinte:

$$S = X \text{ XOR } Y = 0 \text{ XOR } 0 = 0 \text{ e } V = XY = 00 = 0$$
