ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

ANA CLÁUDIA DE OLIVEIRA PEDRO ANDRÊO

SOBRE OS AUTORES

Ana Cláudia de Oliveira Pedro Andrêo

Mestre em Engenharia Elétrica pela UFU-MG
Graduação em Engenharia Elétrica pela UFU-MG

Meu nome é Ana Cláudia de Oliveira Pedro Andrêo e sou Engenheira Eletricista pela Universidade Federal de Uberlândia.

Durante meu mestrado, atuei com a área de Processamento da Informação, mais especificamente Inteligência Artificial. O título de minha dissertação foi *Redes Neuro-Nebulosas Auto-Organizáveis*, na qual trabalhei com reconhecimento de sinais de voz utilizando técnicas de Redes Neurais Artificiais e Lógica Nebulosa.

Sou professora universitária há mais de dezenove anos, ministrando disciplinas na área de Sistemas Digitais, Arquitetura e Organização de Computadores, Inteligência Artificial, Instalações Elétricas, Matemática Fundamental, entre outras.

Também já fui coordenadora dos cursos de Ciência da Computação e Engenharia da Computação, mas atualmente estou atuando na área que mais gosto: trabalhando com os alunos.

Introdução

Olá, querido(a) aluno(a).

Seja muito bem-vindo(a) à nossa disciplina de Arquitetura e Organização de Computadores.

Na de informática, o aprendizado que você obtiver nesta disciplina será de fundamental importância para sua vida acadmica e profissional.

Em um curso superior, não basta somente saber fazer, e sim entender o que se faz e como solucionar problemas que possam surgir. Somente com o conhecimento da teoria aliada à prática é que é possível entender os conteúdos como um todo e efetuar julgamentos críticos.

Para que esses objetivos sejam alcançados, dividimos este material em quatro unidades.

Na Unidade 1, trabalharemos com a conversão de bases e aritmética computacional, englobando os sistemas de numeração utilizados na área da computação, e veremos como funcionam os diversos métodos para mudanças de base. Na aritmética binária, veremos as operações de adição, subtração, multiplicação e divisão, com valores inteiros tanto sem sinal quanto positivos e negativos. Finalizaremos essa unidade com os números fracionários, aprendendo o formato utilizado para sua representação.

Na Unidade 2, você aprenderá conceitos da lógica digital, principalmente as portas lógicas, que formam todos os circuitos que fazem o sistema computacional funcionar. Além disso, também veremos como desenhar esses circuitos e simplificá-los, para que possamos obter o melhor desempenho possível no sistema.

Na Unidade 3, focaremos o funcionamento do computador propriamente dito: seus conceitos, histórico e componentes. Nessa unidade, veremos também como avaliar o desempenho de um computador, os principais equipamentos que fazem parte dele e o funcionamento do "cérebro" do sistema computacional: o microprocessador. O estudo do microprocessador englobará suas funções básicas e como ele trabalha com as instruções.

A Unidade 4 trará um estudo sobre os subsistemas de memória, sua organização, operações que podem ser realizadas com ela, seu funcionamento e a relação entre velocidade/capacidade/custo. Também serão trabalhados os dispositivos de entrada e saída, além do armazenamento dos dados. Finalizamos essa unidade com conceitos sobre o funcionamento do sistema computacional mais moderno, englobando as arquiteturas RISC e os sistemas multiprocessados.

Todas as unidades apresentam exemplos resolvidos e exercícios propostos.

Aproveite bastante e bons estudos!

UNIDADE I

Conversão de Bases e Aritmética Computacional

Ana Cláudia de Oliveira Pedro Andrêo

O computador é uma máquina, formada de circuitos lógicos e que utiliza códigos totalmente diferentes dos que estamos acostumados em nosso dia a dia.

Esses circuitos lógicos realizam todas as suas operações utilizando sinais que podem ser representados somente de duas formas: 0 e 1s. Esses 0s e 1s são chamados bits, que configuram a menor quantidade de informação que se tem e fazem parte de um sistema de numeração chamado binário.

Nesta unidade, trabalharemos com os principais sistemas de numeração que fazem parte de um computador, com ênfase nesse sistema binário. Vamos estudar como é realizada a conversão de bases entre os sistemas decimal, octal, hexadecimal e binário, verificando o método mais apropriado para realizar cada uma delas.

Além disso, aprenderemos como realizar as principais operações aritméticas no sistema binário, além da representação de valores inteiros e fracionários nos padrões utilizados pelo sistema computacional.

Notação posicional, Sistemas numéricos e Conversões

A notação posicional é a forma mais utilizada para a representação dos números. Por meio dela, tem-se que cada um dos algarismos de uma base numérica assume valores diferentes dependendo de sua posição no número. Isso significa que é a posição do algarismo, que pode ser chamado também de dígito, que irá determinar seu valor. Para que se possa obter o valor total do número, deve-se somar os valores relativos de cada dígito.

Pode-se citar, como exemplo, em um sistema decimal que possui dez algarismos (0, 1, 2, 3, 4, 5, 6, 7, 8 e 9) e base dez:

$$2415 = 2.10^3 + 4.10^2 + 1.10^1 + 5.10^0 = 2000 + 400 + 10 + 5 = 2415$$

Utilizando-se a fórmula para calcular o valor do número, tem-se:

$$N = d_{n-1}.b^{n-1} + d_{n-2}.b^{n-2} + ... + d_1.b^1 + d_0.b^0$$

Lembre-se que:

- d indica cada algarismo ou dígito do número;
- n é o número de dígitos do número;
- n-1, n-2, 1, 0 indicam a posição do algarismo; e
- b é a base numérica.

Na área de computação, utilizamos quatro bases numéricas ou sistemas numéricos: decimal (que serve de interface com o usuário), base binária (essa é a utilizada pelo computador) e as bases octal e hexadecimal (usadas para interfacear o sistema numérico decimal e o sistema binário). A necessidade de se utilizar essa interface entre os sistemas utilizados pelos seres humanos e o sistema computacional dá-se pela dificuldade em escrever os valores em binários, pois eles, muitas vezes, possuem um número de bits elevado.

Sistema decimal

Como já foi dito anteriormente, o sistema decimal possui dez algarismos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. Seu nome vem do valor de sua base, que é 10.

Esse sistema é o utilizado em nosso dia a dia.

São exemplos de valores no sistema decimal: 321, 9745, 17, 3, entre outros.

Sistema binário

O sistema binário é aquele utilizado pelo sistema computacional, pois a informação é representada, dentro de um sistema digital, por sinais elétricos. Esses sinais podem ser recebidos de equipamentos externos ou gerados e armazenados dentro do sistema. O armazenamento pode ser feito, também, utilizando-se campos magnéticos e sinais óticos.

Os sinais elétricos são representados por diferentes níveis de tensão, devendo existir uma faixa de tolerância entre eles para que se possa garantir a precisão da informação.

Por isso é que não se pode utilizar o sistema decimal para representação interna. Imagine só: teríamos que ter dez níveis diferentes de tensão, com uma margem de segurança entre eles. Como ficaria a tensão que teria que ser utilizada?

Assim, o sistema binário utiliza somente dois dígitos: 0 e 1 e sua base é 2.

São exemplos de valores no sistema binário: 1011, 01110, 1111111, 10, 1, 0.

Sistema octal

O sistema octal possui 8 algarismos: 0, 1, 2, 3, 4, 5, 6 e 7 e a base utilizada para sua representação é 8.

São exemplos de valores no sistema octal: 7, 54, 23, 425.

Sistema hexadecimal

Como se pôde reparar nos exemplos de valores binários, a representação dos números fica razoavelmente extensa e, em alguns casos, excessivamente grande. Assim, há necessidade de uma interface entre os valores decimais e os binários, para que se possa diminuir os erros na hora dos programadores inserirem dados no sistema.

O sistema hexadecimal é o utilizado para essa interface. Como ele possui mais algarismos numéricos, no caso 16, que é a sua base, o tamanho dos números fica menor.

Mas como representar 16 algarismos diferentes se só temos 10 conhecidos? Nesse caso, passamos a utilizar, também, as primeiras letras do alfabeto.

Assim, tem-se como dígitos do sistema hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F. O A corresponde a dez em decimal, o B a onze, o C a doze e assim por diante, até chegarmos a F, que corresponde a quinze.

São exemplos de valores no sistema hexadecimal: BA5, 34E2, FF, A, 2.

¶ Reflita

Quando é utilizado o sistema hexadecimal, podemos reconhecer isso rapidamente, pois aparecem letras na composição dos valores. Mas isso não é tão fácil nos outros sistemas. Conforme estudamos, há algarismos que se repetem em todos os sistemas. Assim, muito cuidado ao utilizá-los. Há sempre a necessidade de que se indique qual é a base que está sendo utilizada.

Conversão entre bases

A conversão entre as bases numéricas pode ser feita de quatro maneiras diferentes: pelo método polinomial, método das subtrações, método das divisões e método da substituição direta. Cada um desses métodos é adequado para a conversão de determinadas bases.

Método polinomial

Como cada número pode ser representado por um polinômio em uma certa base, tudo o que se deve fazer para transformar um número de uma base para outra é interpretar esse número como um polinômio utilizando-se a aritmética da base de destino (WEBER, 2012).

Esse método utiliza a fórmula da notação posicional vista no início dessa unidade, considerando-se a base de origem e a base de destino.

$$N = d_{n-1}.b^{n-1} + d_{n-2}.b^{n-2} + ... + d_1.b^1 + d_0.b^0$$

Nesse caso, b é a base de origem e N o valor do número, já na base de destino.

Exemplos:

Utilizando o método polinomial, converta os valores iniciais para a base indicada:

a. $1011_2 = (\ )_{10} = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 8 + 0 + 2 + 1 = 11_{10}$

b. $AB3_{16} = (\ )_{10} = A.16^2 + B.16^1 + 3.16^0 = 10.256 + 11.16 + 3.0 = 2560 + 176 + 3 = 2739_{10}$

c. $137_8 = (\ )_{10} = 1.8^2 + 3.8^1 + 7.8^0 = 1.16 + 3.8 + 7.1 = 64 + 24 + 7 = 95_{10}$

Lembre-se: o método polinomial é mais utilizado para a conversão de valores para a base 10!

Método das subtrações

O método das subtrações também é baseado na fórmula da notação posicional. Para realizá-lo, deve-se ir subtraindo os maiores produtos possíveis na nova base até que sejam obtidos todos os dígitos.

Note-se que o resultado das diversas subtrações sempre deve ser positivo (ou zero). Se a subtração não for possível, isso indica que o coeficiente **xi** é zero (WEBER, 2012).

Exemplos:

Utilizando o método das subtrações, converta os valores iniciais para a base indicada:

$$95_{10} = (\ )_2$$

O maior produto utilizando a potência de 2, menor do que o valor a ser convertido, que se tem é $64 = 2^6$.

- Assim: $95 - 1.2^6 = 95 - 64 = 31$
- O próximo maior produto é **32**: $31 - 0.2^5 = 31 - 0 = 31$
- O próximo maior produto é **16**: $31 - 1.2^4 = 31 - 16 = 15$
- O próximo maior produto é **8**: $15 - 1.2^3 = 15 - 8 = 7$
- O próximo maior produto é **4**: $7 - 1.2^2 = 7 - 4 = 3$
- O próximo maior produto é **2**: $3 - 1.2^1 = 3 - 2 = 1$
- O último produto é **1**: $1 - 1.2^0 = 1 - 1 = 0$

Assim, o valor em binário é 1011111 (*basta ir "pegando" os valores multiplicativos das potências de 2*).

Uma maneira mais fácil de realizar essa conversão é montar uma tabela com os valores das potências de 2 e ir completando-a:

$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
1 (95-64 = 31)	0	1 (31 - 16 = 15)	1 (15 - 8 = 7)	1 (7 - 4 = 3)	1 (3 - 2 = 1)	1 (1 - 1 = 0)

Nesse caso, você deverá ir subtraindo os valores de cada coluna da tabela.

Para montagem dessa tabela, é importante lembrar os valores das potências de 2.

Dica:

Você pode começar pelo 1 e ir dobrando os valores das colunas, caminhando da direita para a esquerda.

Lembre-se

O método das subtrações é mais utilizado para a conversão da base 10 para a base 2, por só termos os dígitos 0 e 1!

Método das divisões

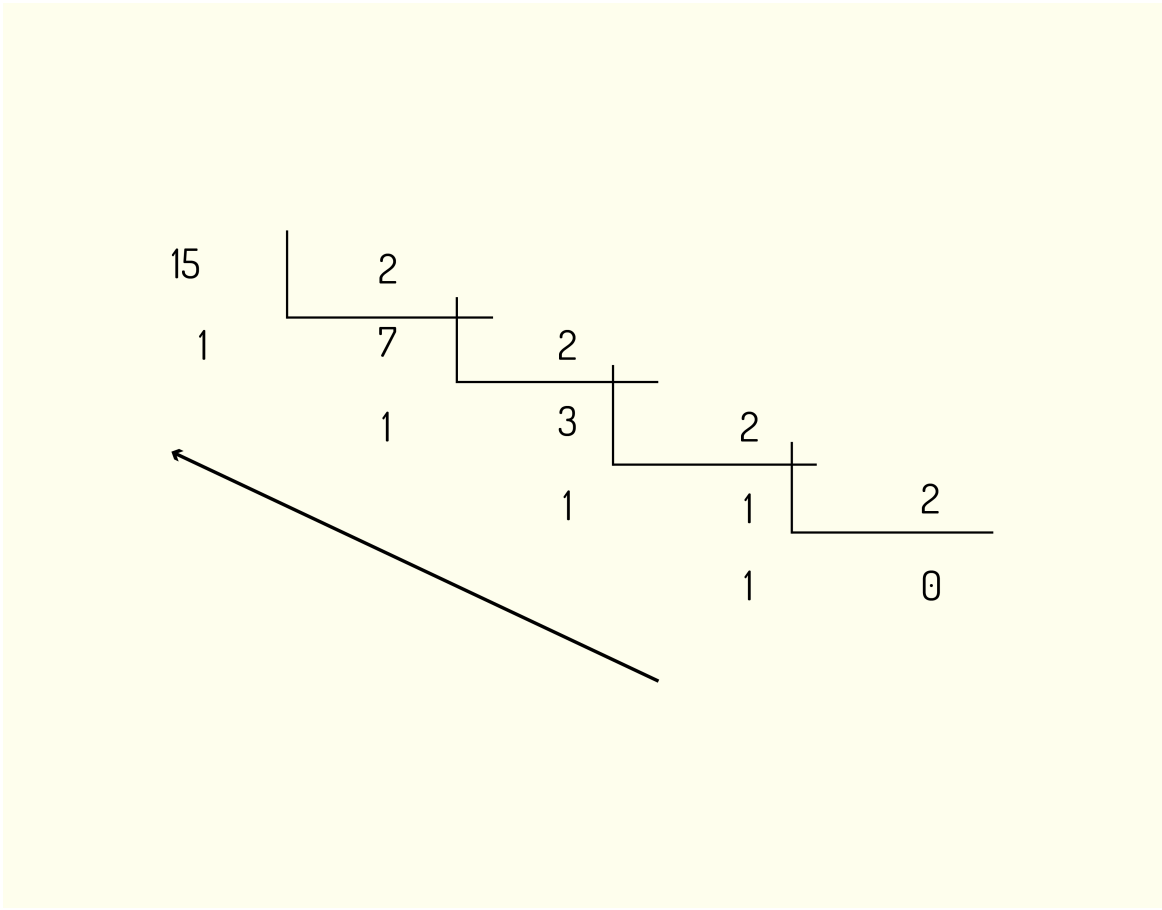
Para se fazer a conversão utilizando esse método, basta fazer divisões sucessivas do número que será convertido pela base para a qual se deseja converter, até que o quociente seja zero.

Esse método é utilizado quando se deseja converter da base decimal para qualquer outra base. O valor convertido é encontrado utilizando-se os restos da divisão, computados de baixo para cima.

Exemplos:

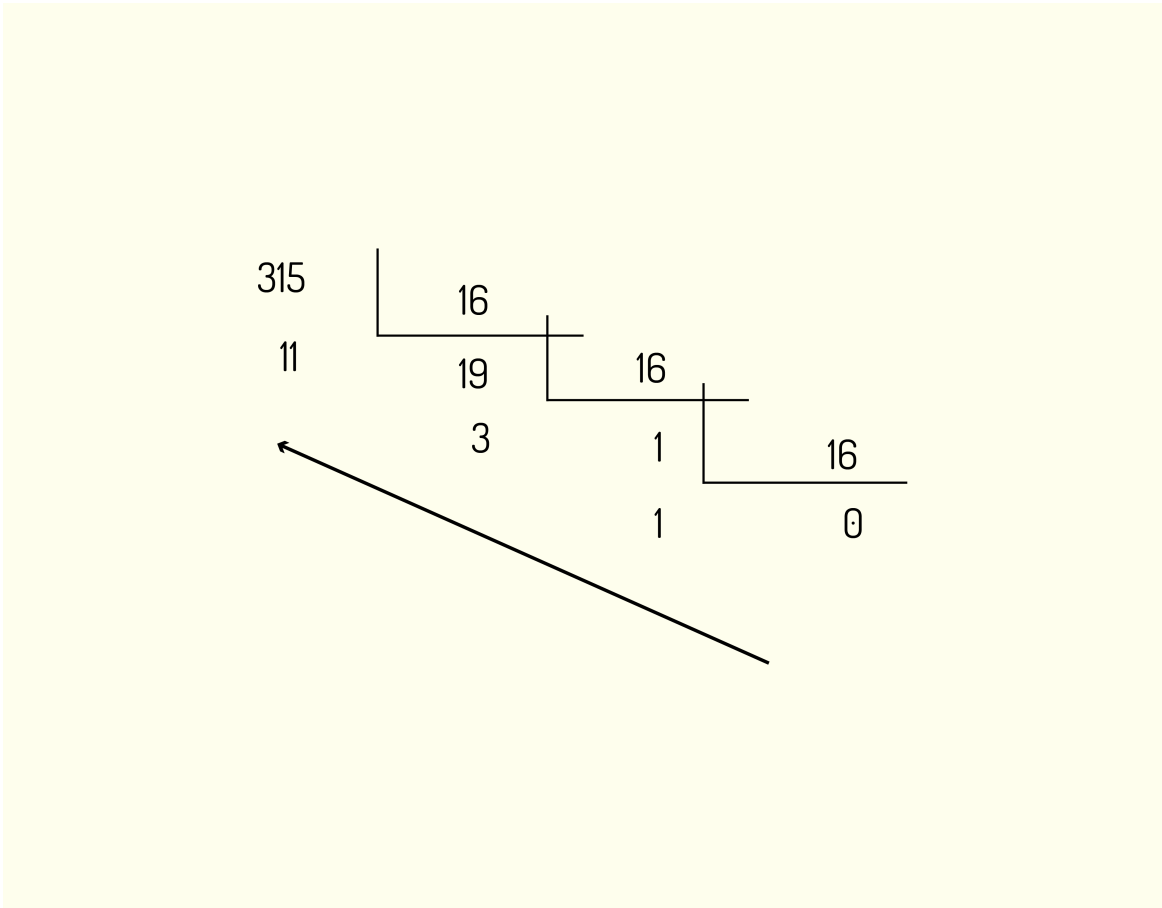
Utilizando o método das divisões, converta os valores iniciais para a base indicada:

a. $15_{10} = (\ )2$



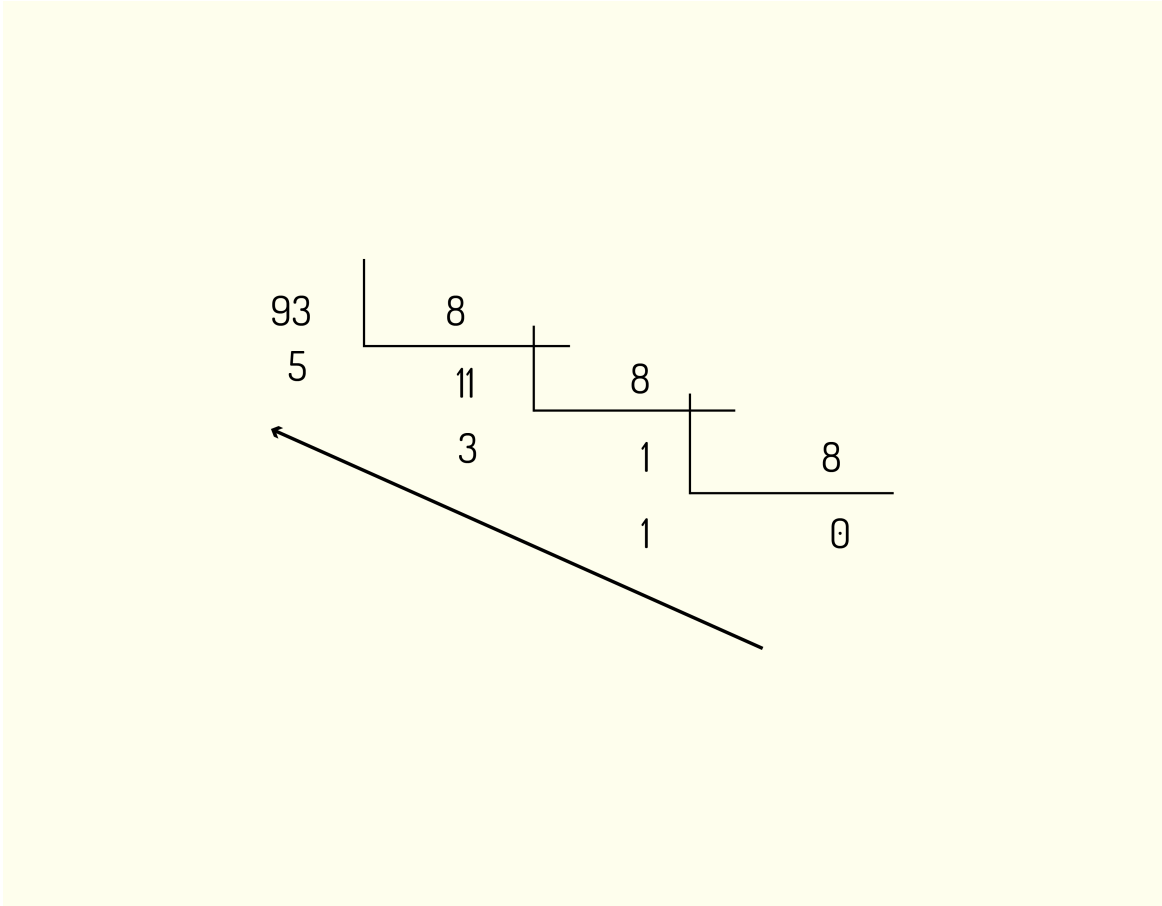
Assim, 15_{10} = 11112

b. 315_{10} = ( )16



Assim, 315_{10} = 13B16

c. 93_{10} = ( )8



Assim, $93_{10} = 1358$

¶ Reflita

Sempre que um número for ímpar na base decimal, também terminará em 1 na base 2 e, se for par, terminará em 0. Isso ocorre porque a única potência que produz um número ímpar (1) é o da última posição, ou seja, 20.

Método da substituição direta

O método da substituição direta é utilizado quando se quer converter da base binária para octal e vice-versa e da binária para hexadecimal, e vice-versa.

Para o seu uso, deve-se ter em mente os quadros de correspondência entre os valores, apresentadas a seguir.

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

1FIGURA 1.7 - Correspondência entre os valores binário e octal FONTE: A autora.

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

1FIGURA 2.7 - Correspondência entre os valores binário e hexadecimal FONTE: A autora.

Lembre-se: quando o valor inicial estiver na base binária, é sempre importante realizar a divisão dos dígitos em grupos de 3, quando se for converter para a base octal, e em grupos de 4, quando se for converter para a base hexadecimal, completando o último grupo com bits 0 à esquerda, se faltar algum.

Exemplos:

Utilizando o método da substituição, converta os valores iniciais para a base indicada:

a. $1101011_2 = (\ )_8$

Conforme vimos anteriormente, primeiro é necessário agrupar em conjuntos de 3 dígitos: 001 101 011 (*como o grupo mais à esquerda possui somente um dígito, completamos com 2 zeros*). Agora, é só utilizar o Quadro 1.3.

Assim, $1101011_2 = 153_8$

b. $110101111111_2 = (\ )_{16}$

Agrupando em conjuntos de 4 dígitos: 0110 1011 1111 (*nesse caso, é preciso completar com somente um 0*). Utilizando o Quadro 1.4.

Assim, $110101111111_2 = 6BF_{16}$

BINÁRIO	OCTAL
001	1
101	5
011	3

1FIGURA 3.7 - Conversão de binário para octal FONTE: A autora.

BINÁRIO	HEXADECIMAL
0110	6
1011	B
1111	F

1FIGURA 4.7 - Conversão de binário para hexadecimal FONTE: A autora.

Aritmética no Sistema Binário

As operações nos sistemas computacionais funcionam da mesma maneira que no sistema decimal. O que muda é a base. Nesse caso, o "vai um" ou o "pede emprestado" passa a ser a base que indica o sistema.

Como já vimos anteriormente, o sistema binário utiliza a base 2 para representação do número e possui apenas dois dígitos: 0 e 1.

Operação de soma ou adição

Você já parou para pensar como é feita a soma no sistema decimal? Depois de tanto tempo realizando essa operação, ela passou a ser automática para nós e nos esquecemos do significado do que estamos fazendo.

Vamos analisar um cálculo em decimal, somando os valores 19 e 15:

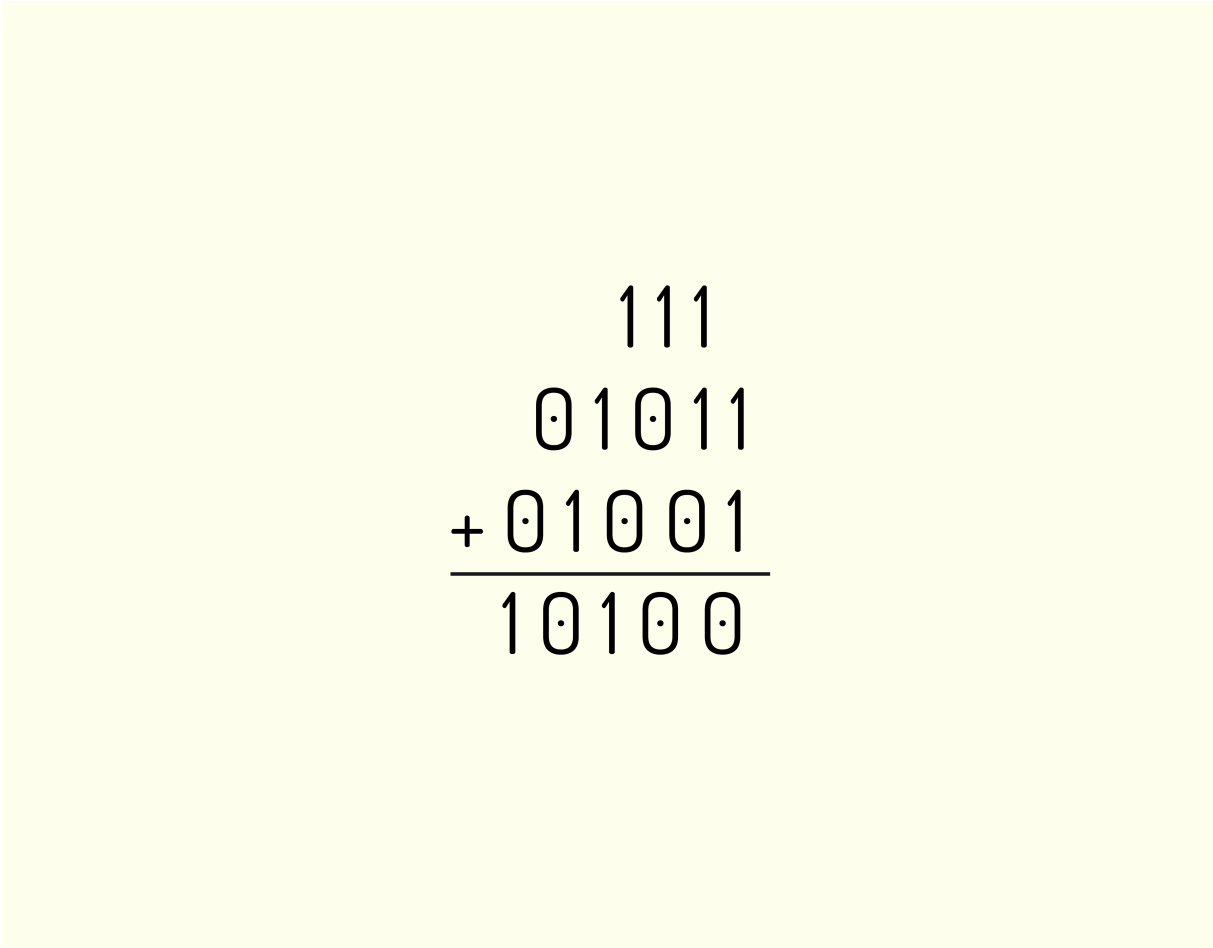
$$\begin{array}{r} 1 \\ 19 \\ +\,15 \\ \hline 34 \end{array}$$

Quando somamos 9 + 5, o valor obtido é 14, que excede o último algarismo que temos na representação em decimal (9). Assim, aparece o vai um para o dígito seguinte. O vai um nada mais é que o valor que excede esse dígito, ou seja, a base. Na realidade, o vai um, nesse caso, vale 10. Se você se lembrar da representação em notação posicional, vai perceber porque o dígito passa a ser 1 ao invés de 10.

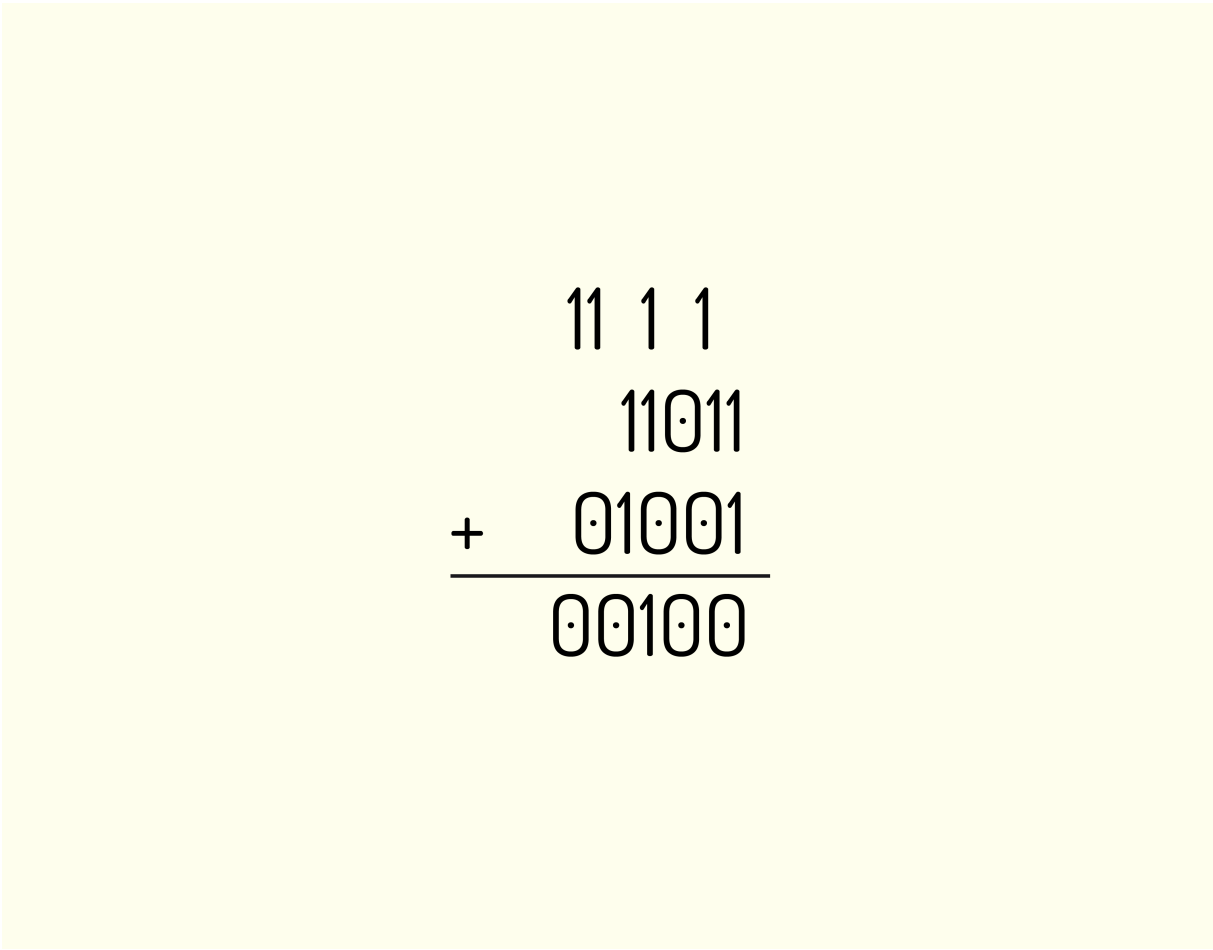
Lembre-se: na notação posicional, o algarismo tem valores diferentes dependendo da posição que ocupa no número e a posição seguinte, caminhando da direita para a esquerda, é acrescida do valor da base.

No sistema binário, a adição funciona da mesma maneira, mas o vai um aqui passa a ser 2, que é a base.

Assim, temos, ao somar os valores binários 01011 e 01001:



Nesse caso, não tivemos problema com a posição do bit de vai um mais à esquerda, pois havia lugar para ele ser colocado. Mas veja o exemplo seguinte:



Nesse caso, ocorreu vai um para fora do número e não temos lugar para colocá-lo no número de bits escolhido para representação (5 bits). Quando isso ocorre, dizemos que aconteceu um *overflow* ou estouro de bits, ou seja, o número de bits escolhido para representar os valores não foi suficiente para realizar o cálculo.

Isso, nos sistemas computacionais, é um problema muito sério, pois o resultado final ocasionou um erro e ele está incorreto.

Assim, é muito importante compreender o conceito de *overflow* e o número de bits necessários para representação dos valores.

Se repetirmos o mesmo cálculo, mas agora utilizando 8 bits para representação, o erro não ocorre mais:

```
   11 1 1
  000 11 0 11
+ 00001001
 ----------
  00100100
```

Quando somamos 2 bits em binário, podemos obter os seguintes valores:

0 + 0 = 0

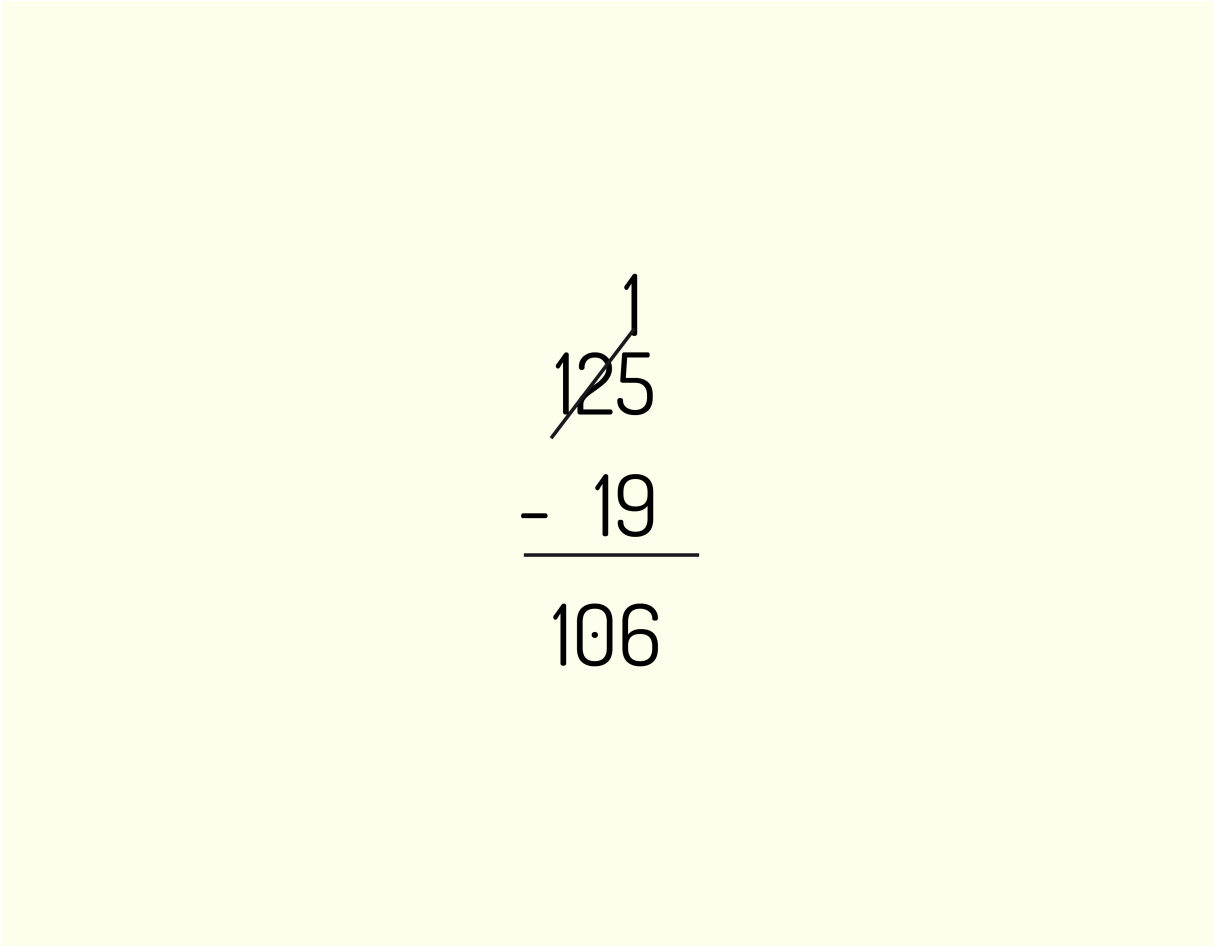
0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 e vai um

Operação de subtração

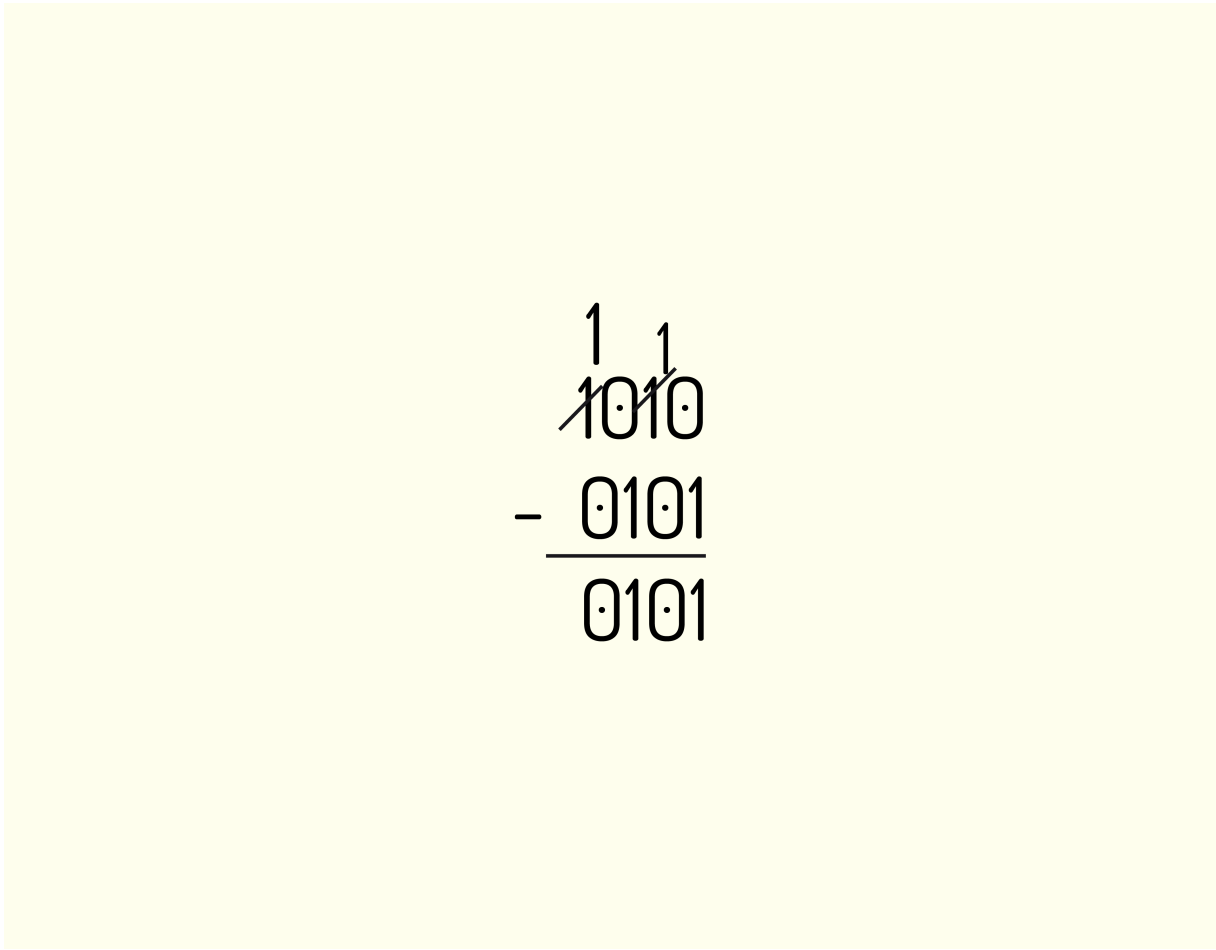
Vamos observar o funcionamento de uma operação de subtração no sistema decimal:



Conforme podemos observar, não é possível realizar a operação 5 - 9. Assim, há a necessidade de pedir emprestado para o dígito mais à esquerda. Quando colocamos o 1 acompanhando o 5, temos 15 - 9 = 6. Ou seja, passamos a ter, para esse dígito, 10 + 5 = 15.

Assim, quando utilizamos o pede emprestado, o que é acrescentado ao algarismo é a base, que, nesse caso, é 10.

Utilizando o mesmo raciocínio para efetuar a subtração no sistema decimal, temos, para os valores 1010 e 0101, em binário:



Não é possível fazer 0 - 1, portanto tivemos que pedir emprestado ao bit mais à esquerda. Nesse caso, a representação desse pede emprestado é a mesma do sistema decimal, mas o valor é totalmente diferente. Como quando pedimos emprestado, o que "vem" é a base, temos:

$$2 + 0 = 2$$

Assim, o cálculo fica: 2 - 1 = 1.

¶ Fique por dentro

Representação do vai um e do pede emprestado#

Apesar da representação do vai um e do pede emprestado ser a mesma em todos os sistemas numéricos, seu valor depende de qual representação está sendo utilizada.

Para o sistema binário, eles valem 2, para o sistema octal, valem 8 e, no sistema hexadecimal, seu valor é 16.

O cálculo é feito da mesma maneira, mas há a necessidade de se ficar atento(a) à base utilizada.

Quando subtraímos 2 bits em binário, podemos obter os seguintes valores:

0 - 0 = 0

0 - 1 = 1*(mas existe a necessidade de pedir emprestado ao bit anterior)*

1 - 0 = 1

1 - 1 = 0

Operação de multiplicação

Também na multiplicação, a operação é semelhante ao que ocorre no sistema decimal, mas temos os seguintes valores:

0. 0 = 0

0. 1 = 0

1. 0 = 0

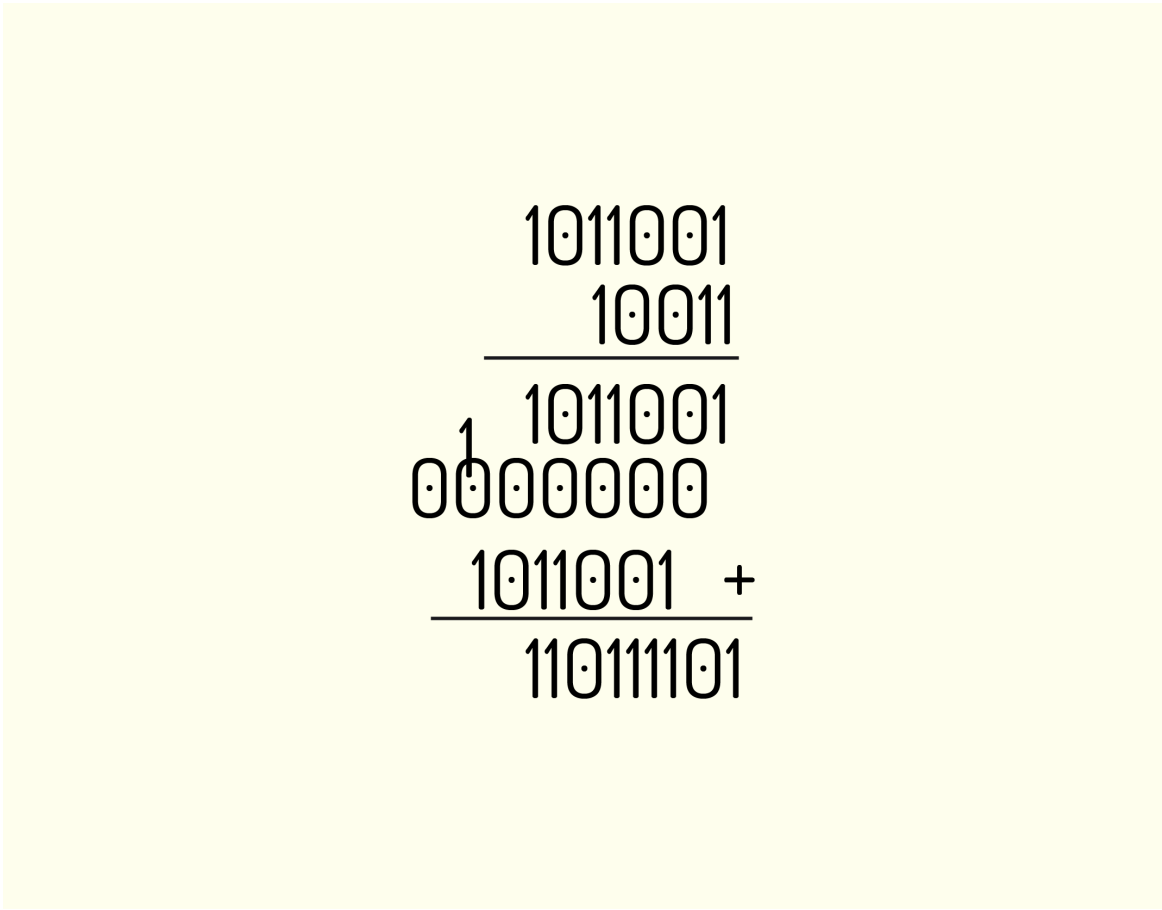
1. 1 = 1

Segundo Monteiro (2012), o procedimento consiste em multiplicar cada algarismo do multiplicador pelos algarismos do multiplicando, resultando em sucessivos produtos parciais, tantos quantos forem os algarismos do multiplicador. Cada produto parcial é colocado de modo a se posicionar uma casa para a esquerda do produto anterior, isto é, há um deslocamento do 2° produto para a esquerda em relação ao 1° produto e há um deslocamento à esquerda do 3° produto em relação ao 2° produto. Em seguida, os três produtos são somados produzindo o resultado desejado.

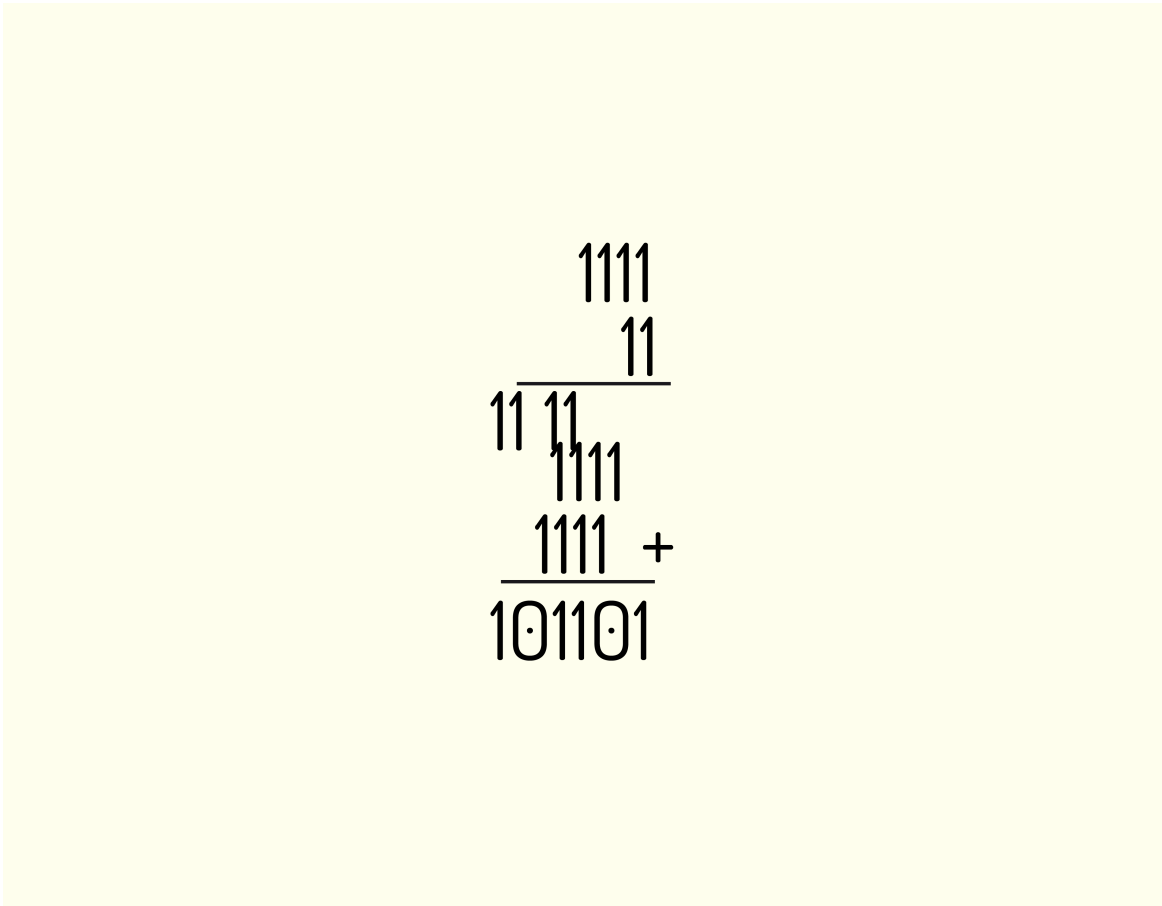
Monteiro (2012) ainda afirma que, no caso de sistemas binários, o procedimento é ainda mais simples porque os produtos parciais só podem ser zero (se o algarismo do multiplicando for zero) ou o próprio valor do multiplicador (se o algarismo do multiplicando for um).

Exemplo:

a. Multiplicar os seguintes valores, em binário: 1011001 e 10011:



b. Multiplicar os seguintes valores, em binário: 1111 e 11:



Note que, nesse caso, apareceram duas novidades:

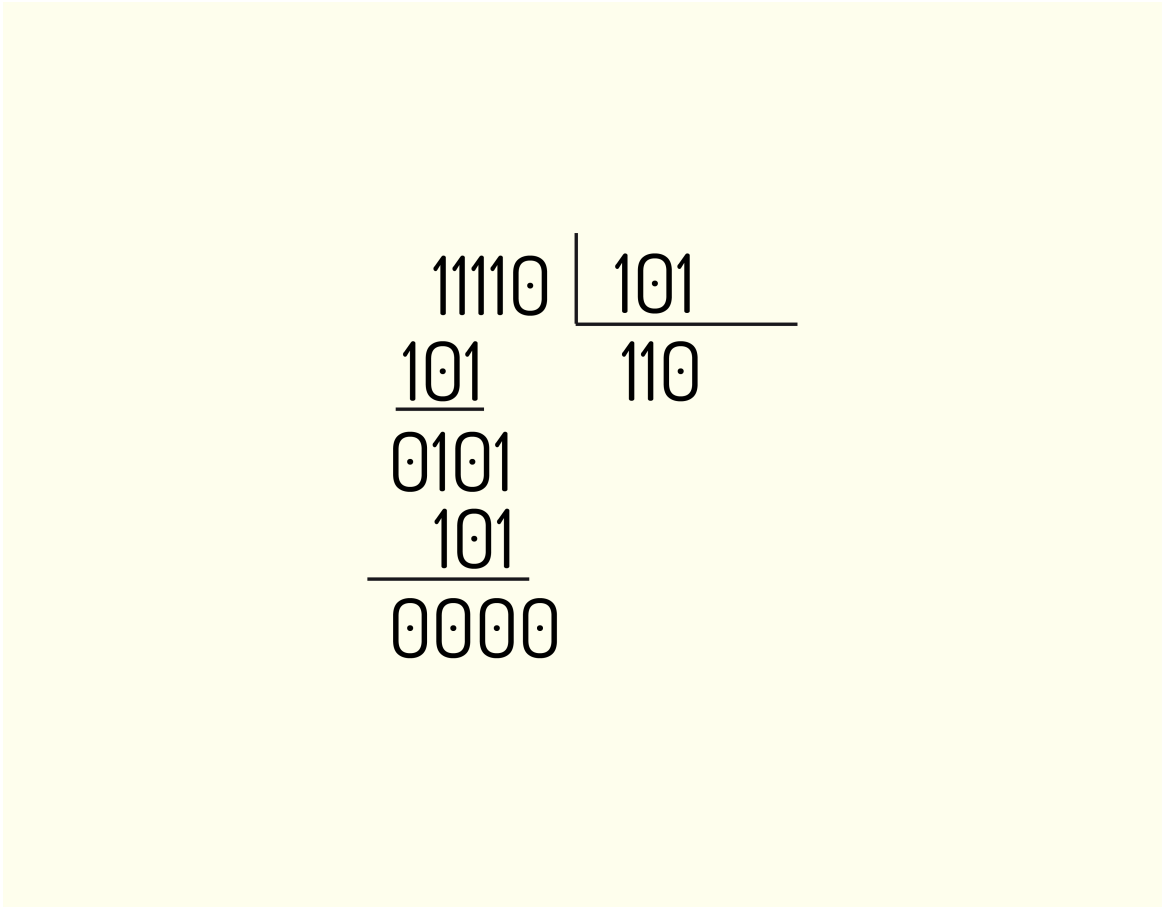
1. O maior número de bits das parcelas é diferente do número de bits do resultado. Se esse número estivesse fixado, haveria *overflow*, e o resultado final estaria comprometido.
2. Na soma entre as parcelas, temos 1 + 1 + 1. O resultado desse cálculo é 1 e vai um.

Operação de divisão

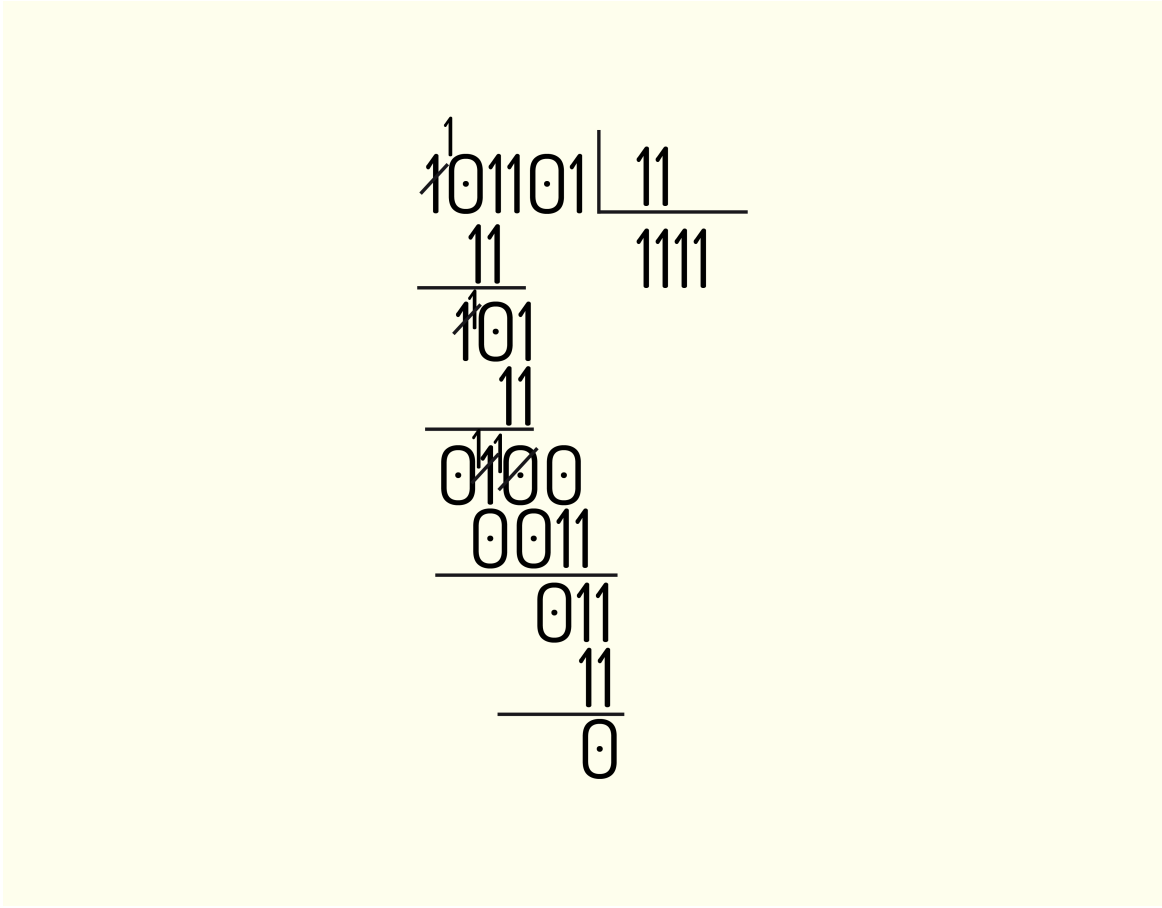
O processo para dividir um número binário (dividendo) por outro (divisor) é o mesmo seguido para números decimais, ao qual normalmente nos referimos como "divisão longa". De fato, esse processo é mais simples com números binários, pois quando verificamos quantas vezes o divisor "cabe" dentro do dividendo, existem apenas duas possibilidades, 0 ou 1 (TOCCI, 2011).

Exemplo:

a. Realizar a divisão do valor binário 11110 por 101, também em binário:



b. Realizar a divisão do valor binário 101101 por 11, também em binário:^li^



Note que, no segundo exemplo, para a subtração que origina o resto, houve a necessidade de utilizar o pede emprestado.

Segundo Idoeta (2012), a divisão de números binários é a mais complexa das operações aritméticas binárias, pois abrange operações de multiplicação e divisão.

Representação Numérica em Ponto Fixo

Segundo Monteiro (2012), para se trabalhar em computação com valores numéricos, deve-se levar em consideração três fatos que podem acarretar inconvenientes no projeto e na utilização da máquina e que, na nossa vida cotidiana (aritmética com papel e lápis), não causam nenhum problema:

- A representação do sinal do número;
- A representação da vírgula (ou ponto) que separa a parte inteira da parte fracionária de um número não inteiro;
- A quantidade limite de algarismos possível de ser processada pela UAL (Unidade Aritmética e Lógica) de um processador.

As representações numéricas vistas anteriormente não levavam em consideração o sinal do número. Tem-se duas maneiras de representar números em ponto fixo (que não possui parte fracionária ou a vírgula é fixa no final do número) com sinal: por Sinal e Magnitude ou por Complemento de 2.

Representação em Sinal e Magnitude

A representação de números com *n* algarismos binários (*n* bits) em sinal e magnitude é obtida atribuindo-se 1 bit (em geral, na posição mais à esquerda do número) para indicar o valor do sinal, e os n-1 bits restantes para indicarem a magnitude (a grandeza) do número (MONTEIRO, 2012).

Os valores utilizados para o sinal são: 0 para números positivos e 1 para números negativos. A grandeza do número é representada da forma estudada anteriormente.

Levando-se isso em consideração, tem-se o seguinte formato para a representação dos números:

SINAL	MAGNITUDE
1 bit	(n-1) bits

A faixa de representação, ou os limites de representação, para os números em sinal e magnitude é:

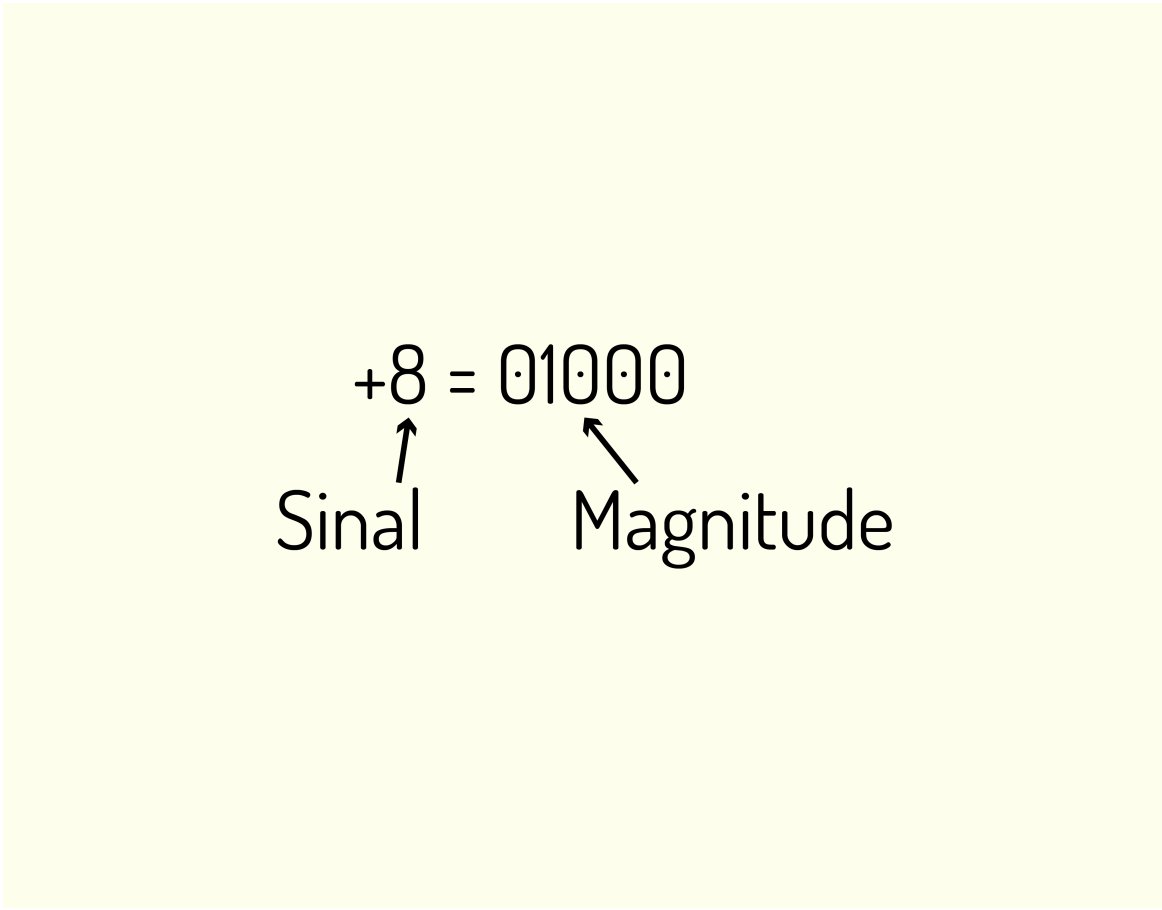
Para n bits: $-(2^{n-1} - 1)$ a $+(2^{n-1} - 1)$.

Assim, para **8 bits**, tem-se: $-(2^{8-1} - 1)$ a $+(2^{8-1} - 1) = -(2^7 - 1)$ a $+(2^7 - 1) = -(128 - 1)$ a $+(128 - 1)$.

Ou seja, com 8 bits, pode-se representar valores entre **- 127 e + 127**.

Exemplos de representação numérica em sinal e magnitude:

a. Utilizando 5 bits:



-8 = 11000

b. Utilizando 8 bits:

+8 = 00001000

-8 = 10001000

A referida representação possui algumas características que, comparadas com as demais representações em ponto fixo (complemento de 1 e complemento de 2), tornam-na menos vantajosa que as outras, razão por que não é utilizada atualmente nos processadores (MONTEIRO, 2012).

As desvantagens são:

a. Existem duas representações para o zero, o que é matematicamente incorreto e ocasiona a necessidade de um circuito lógico exclusivo para evitar uma má interpretação do valor;
b. para que as operações matemáticas possam ser realizadas na ULA (Unidade Aritmética e Lógica) precisa-se de dois circuitos diferentes, um para realizar a adição e outro para realizar a subtração, tornando o algoritmo de soma/subtração mais complexo e mais lento.

Representação em Complemento de 2

Assim como a representação sinal-magnitude, a representação em complemento de dois utiliza o bit mais significativo como bit de sinal, o que torna fácil testar se um número inteiro é positivo ou negativo. Entretanto, os demais bits são interpretados de maneira diferente (STALLINGS, 2010).

Segundo Tocci (2011), tem-se:

- Se o número for positivo, a magnitude é representada na forma binária direta, e um bit de sinal 0 é colocado em frente ao bit mais significativo (*most significant* bit - MSB);
- se o número for negativo, a magnitude é representada na forma do complemento de 2, e um bit de sinal 1 é colocado em frente ao MSB.

Para que se possa encontrar o complemento de 2 de um número binário, primeiro é necessário calcular seu complemento de 1. Isso é obtido, de maneira prática, simplesmente invertendo cada um dos bits, ou seja, troca-se cada um dos dígitos por seu complemento.

Exemplo:

Cálculo de complemento de 1 do número binário 10110101:

NÚMERO BINÁRIO	**10110101**
Complemento de 1	01001010

Agora, para encontrar o complemento de 2 do mesmo número, somamos 1 ao resultado encontrado:

Assim, 1101011111112 = 6BF16

NÚMERO BINÁRIO	**10110101**
Complemento de 1	01001010 +1
Complemento de 2	01001011

Para n bits: -2^{n-1} a $+(2^{n-1} - 1)$.

Assim, para 8 bits, tem-se: -2^{8-1} a $+(2^{8-1} - 1) = -2^7$ a $+(2^7 - 1) = -128$ a $+(128 - 1)$.

Ou seja, com 8 bits, pode-se representar valores entre - 128 e + 127.

¶ Fique por dentro

Intervalo de variação

A diferença no intervalo de variação entre a representação em sinal e magnitude e a em complemento de 2 é de um número. Esse número corresponde justamente ao zero com sinal negativo, que não aparece nesse método.

Exemplos de representação numérica em complemento de 2:

Monteiro (2012) afirma que se pode dizer, sem risco de erro, que a quase totalidade dos computadores modernos utilizam aritmética de complemento a 2 (quando se trata de representação em ponto fixo), devido às duas grandes vantagens daquele método sobre sinal e magnitude, e mesmo sobre complemento a 1:

- Possuir uma única representação para o zero;
- Necessitar de apenas um circuito somador para realizar, não só operações de soma, mas também operações de subtração (mais barato).

Aritmética em Complemento de 2

Como, segundo Weber (2012), em sistemas atuais, a base é binária e utiliza-se para tratamento de números negativos a representação em complemento de 2, esse é o método que iremos usar para realizar a soma, na base 2, de números com sinal.

Adição em Complemento de 2

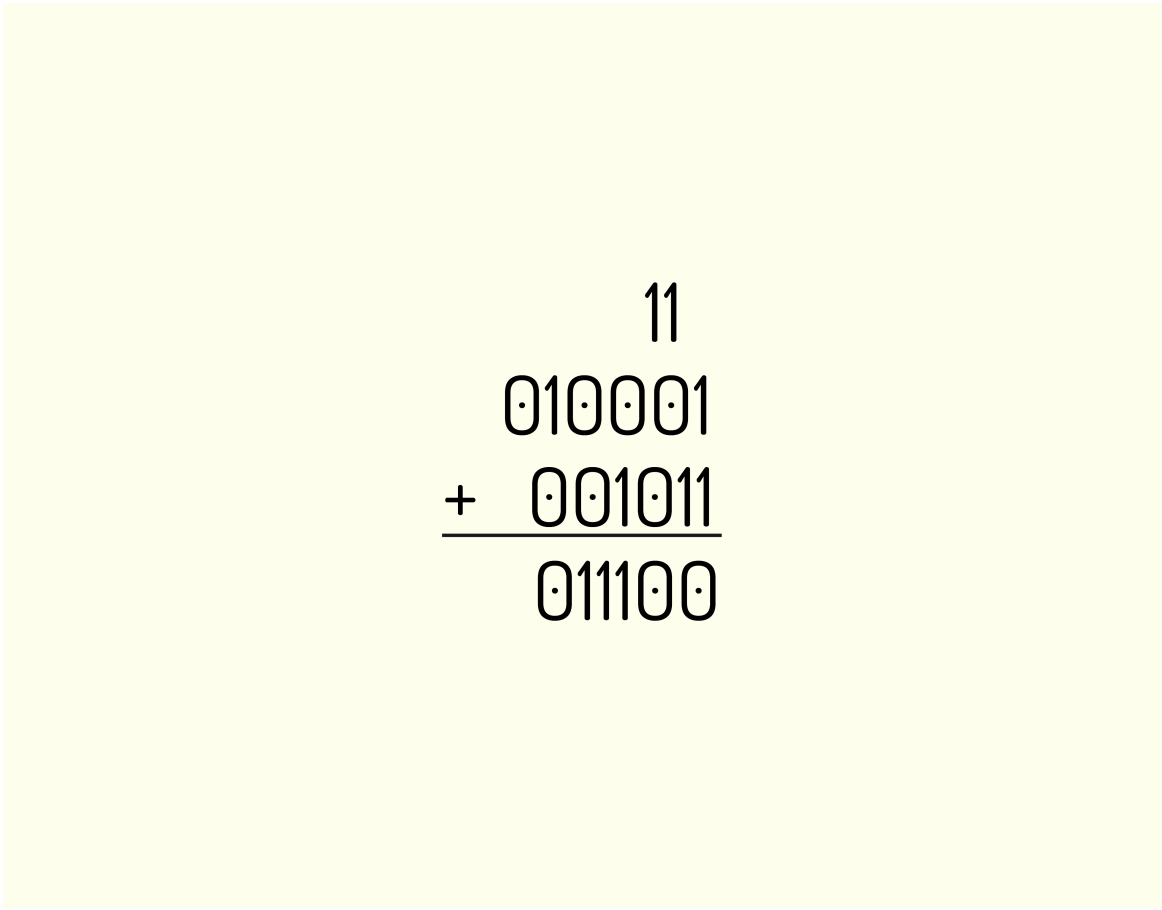
Monteiro (2012) sugere o seguinte algoritmo para a operação de adição em complemento de 2:

a. Somar os dois números, bit a bit, inclusive o bit de sinal;
b. Desprezar o último "vai 1" (para fora do número), se houver;
c. se, simultaneamente, ocorrer "vai 1" para o bit de sinal e "vai 1" para fora do número, ou se ambos não ocorrerem, o resultado está correto;
d. Se ocorrer apenas um dos dois "vai 1" (ou para o bit de sinal ou para fora), o resultado está incorreto. Ocorreu um overflow. O overflow somente pode ocorrer se ambos os números tiverem o mesmo sinal (seja positivo ou ambos negativos) e, nesse caso, se o sinal do resultado for oposto ao dos números.

Exemplos:

Realize as adições a seguir, utilizando a notação em complemento de 2 com 6 bits e analisando se o resultado está correto ou não:

a. 17 + 11
   17 = 010001
   11 = 001011



Nesse caso, o resultado está correto, não houve *overflow*, pois não houve vai um para o bit de sinal nem para fora. Se convertermos o valor para decimal, podemos visualizar isso facilmente:

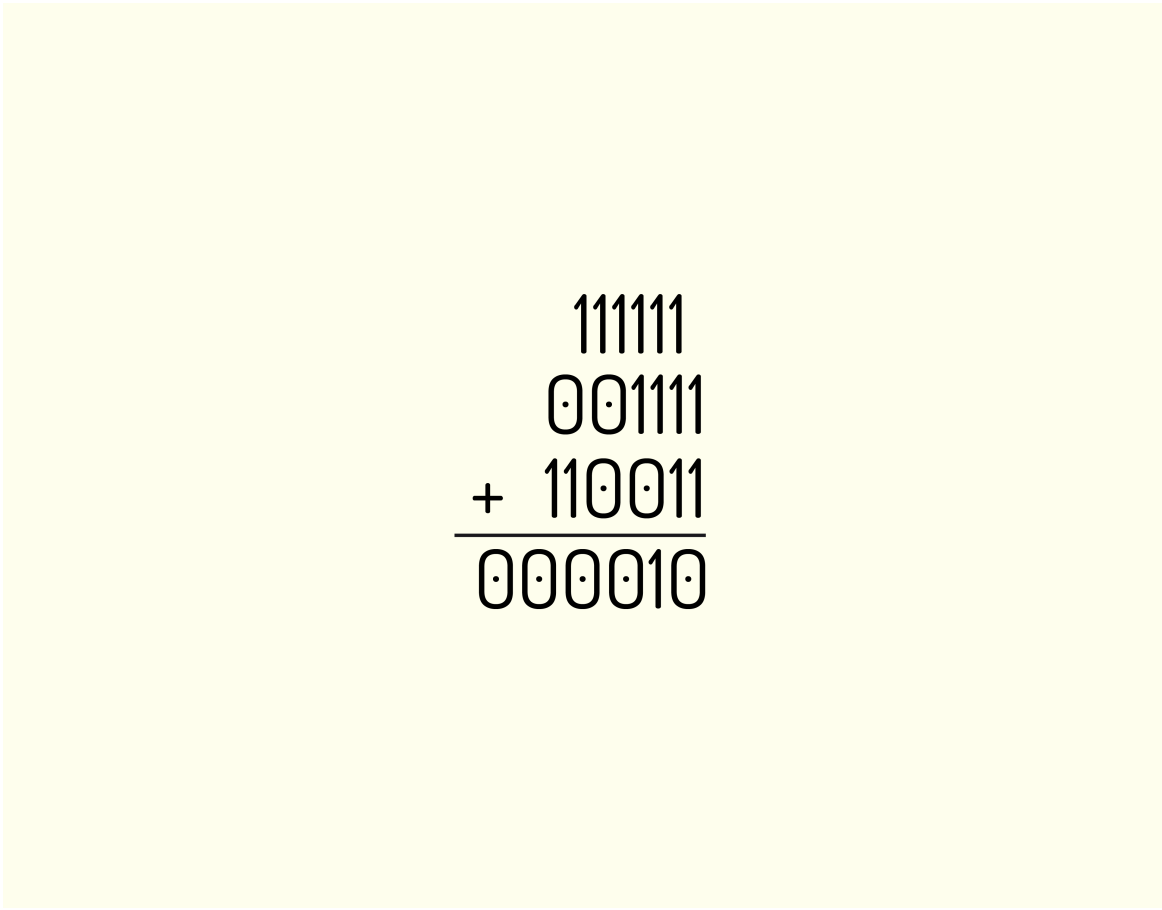
17 + 11 = 28 = 011100

b. 15 + (-13)

15 = 001111

13 = 001101

-13 = C2 13 = 110011



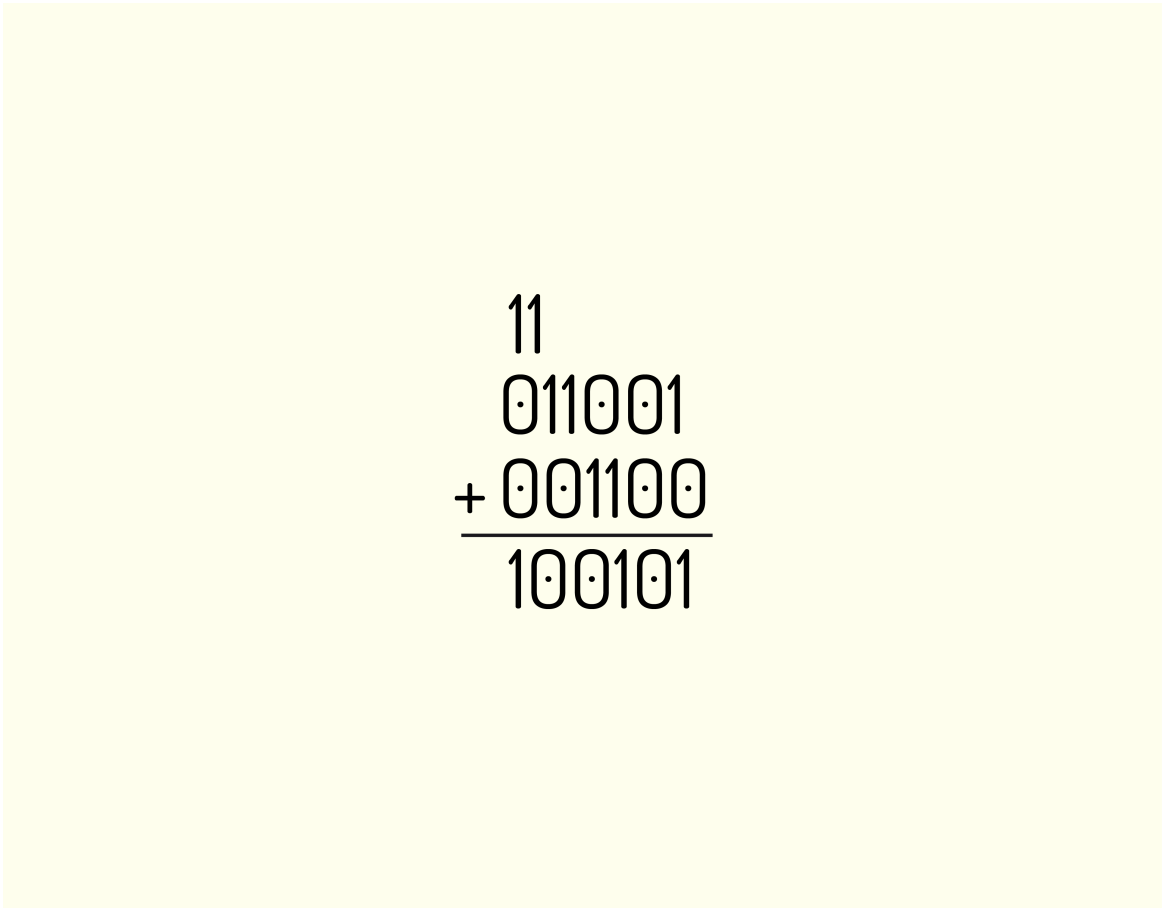
O resultado, nesse caso, também está correto, não houve *overflow*, pois houve vai um para o bit de sinal e para fora. Se convertermos o valor para decimal, podemos visualizar isso facilmente:

15 + (-13) = 2 = 000010

c. 25 + 12

25 = 011001

12 = 001100



Agora, o resultado está incorreto, houve *overflow*, pois houve vai um para o bit de sinal e não teve para fora. Se convertermos o valor para decimal, podemos visualizar isso facilmente:

25 + 12 = 37 ≠ 100101 = -27

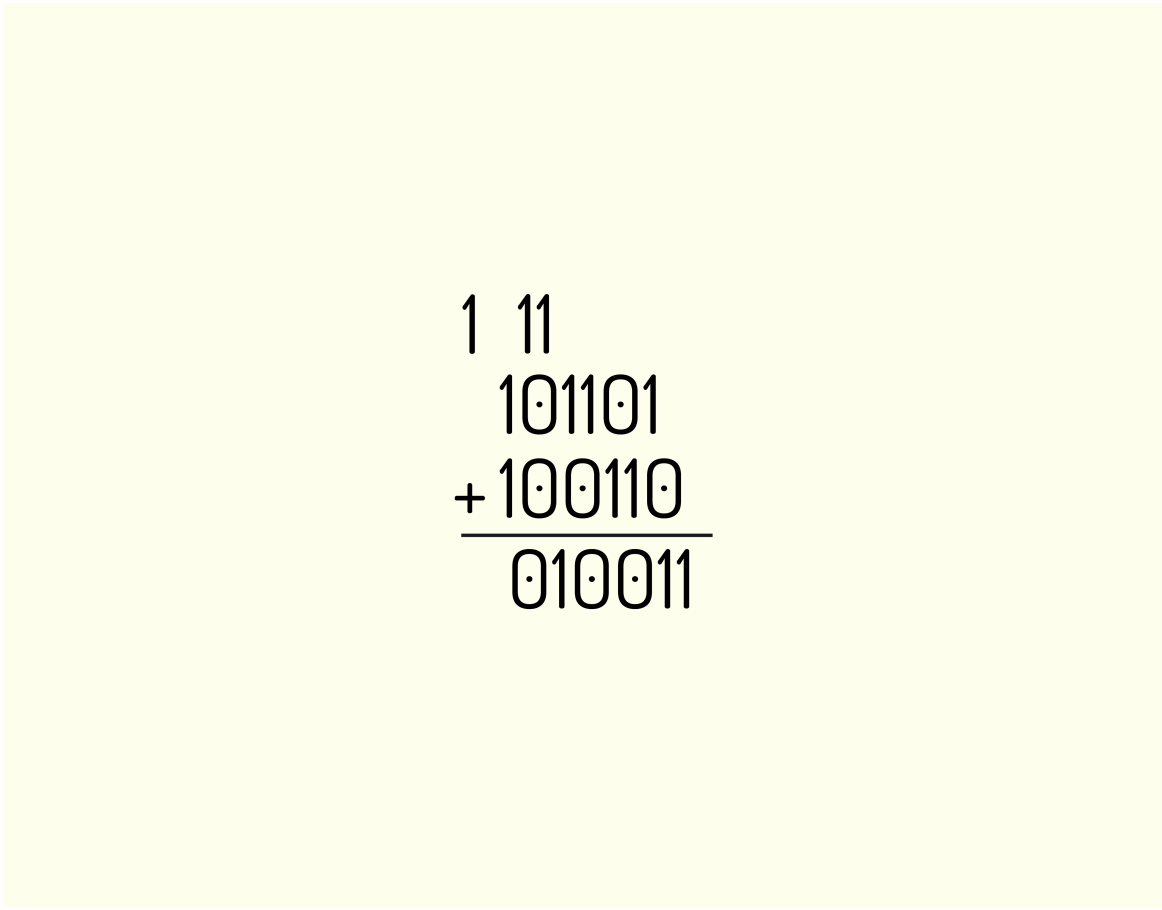
d. -19 + (-26)

19 = 010011

-19 = C2 19 = 101101

26 = 011010

-26 = C2 26 = 100110



Esse o resultado também está incorreto, houve *overflow*, pois ocorreu vai um somente para fora. Se convertermos o valor para decimal, podemos visualizar isso facilmente:

-19 + (-26) = - 45 ≠ 010011 = 19

Subtração em Complemento de 2

Monteiro (2012) também sugere um algoritmo para a operação de subtração em complemento de 2:

a. Complementar a 2 o subtraendo, independentemente se é um valor positivo ou negativo;
b. Somar ambos os números, utilizando o algoritmo de adição já mostrado antes.

Exemplos:

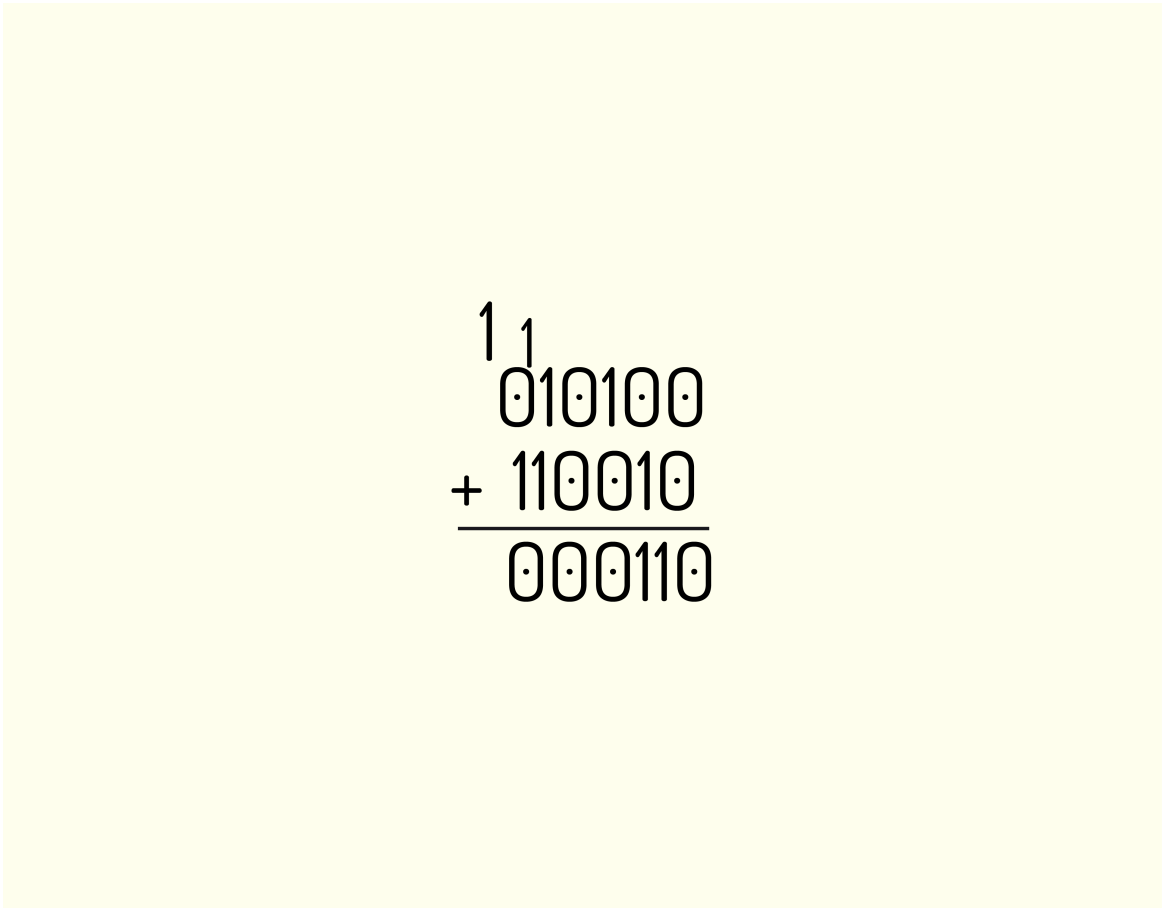
Realize as subtrações a seguir, utilizando a notação em complemento de 2 com seis bits e analisando se o resultado está correto ou não:

a. a. 20 - 14

20 = 010100

14 = 001110

-14 = C2 14 = 110010



Nesse caso, o resultado está correto, não houve *overflow*, pois houve vai um para o bit de sinal e para fora. Se convertermos o valor para decimal, podemos visualizar isso facilmente:

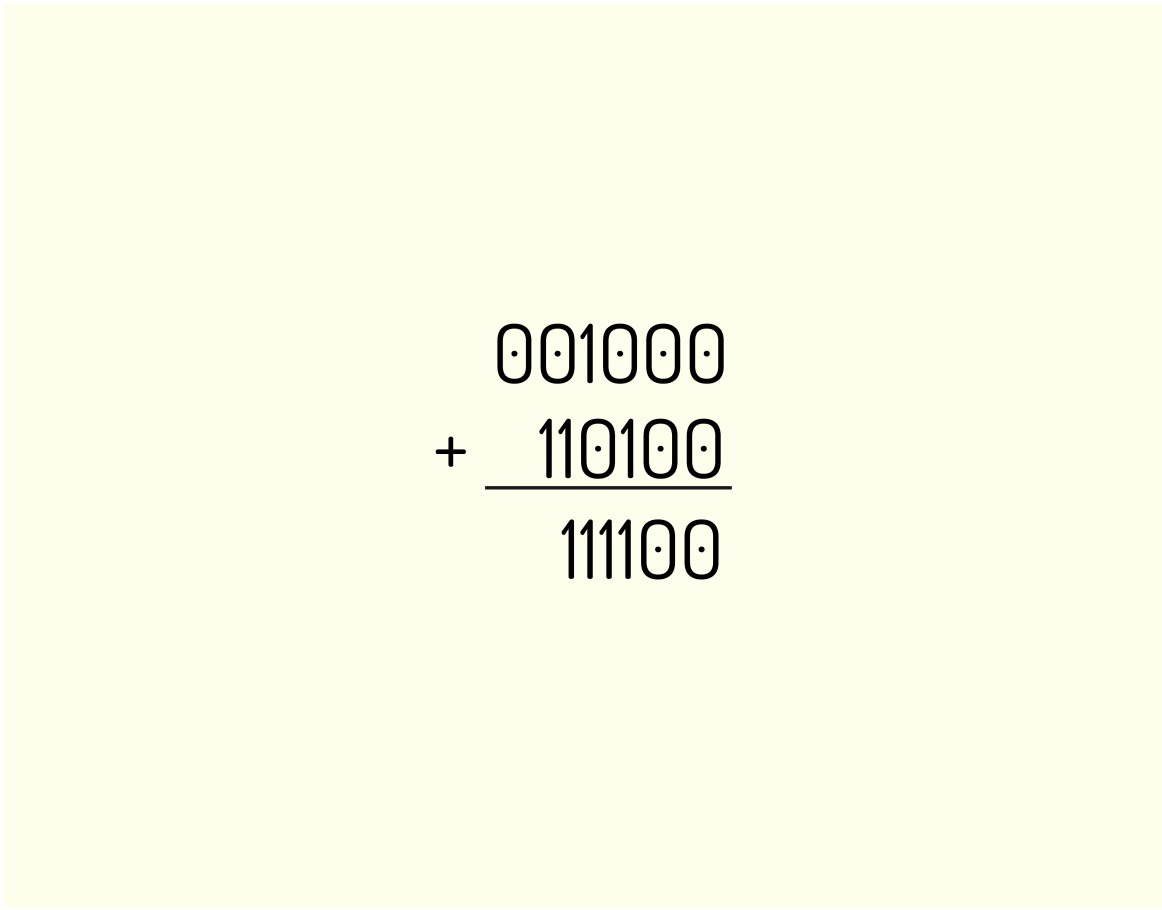
20 - 14 = 6 = 000110

b. 8 - 12

8 = 001000

12 = 001100

-12 = C2 12 = 110100



Nesse caso, o resultado também está correto, não houve *overflow*, pois não houve vai um para o bit de sinal e nem para fora. Se convertermos o valor para decimal, podemos visualizar isso facilmente:

8 - 12 = -4 = 111100

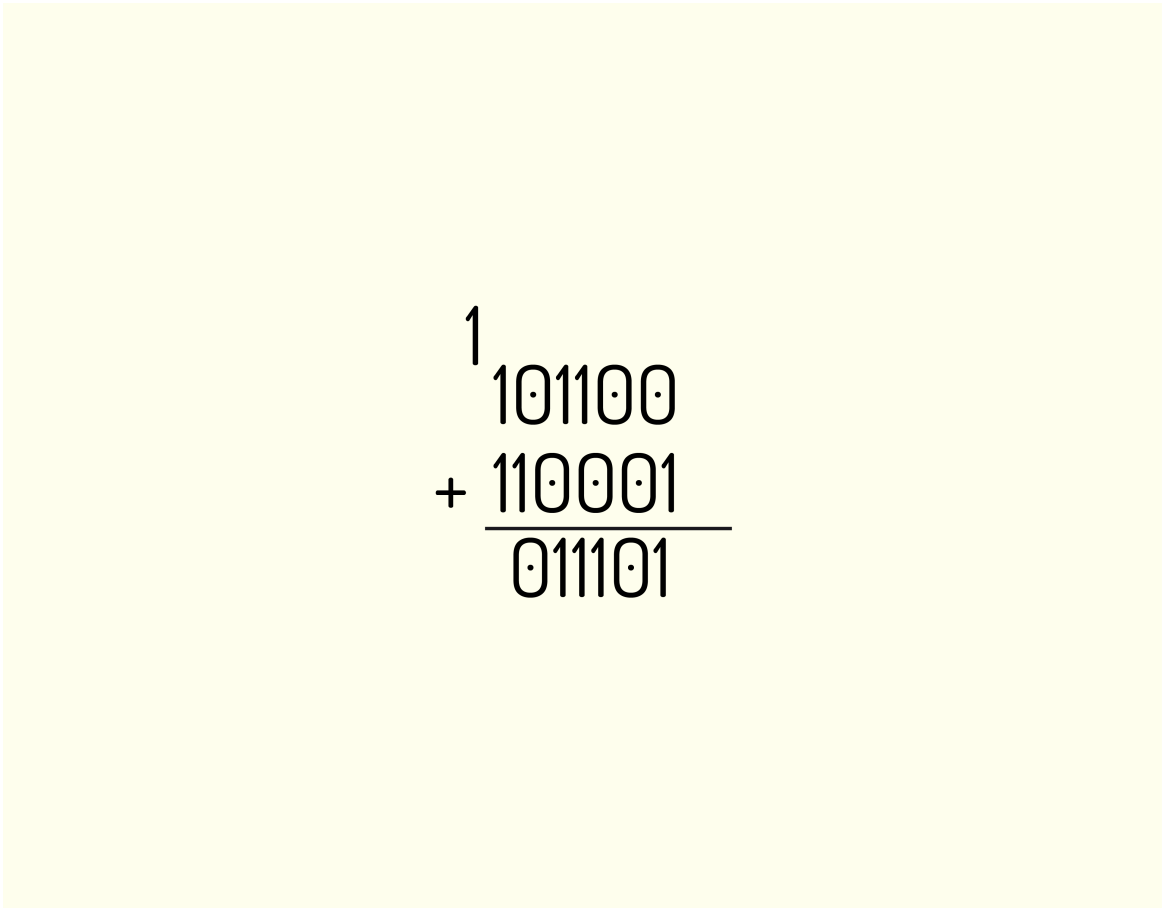
c. -20 - 15

20 = 010100

-20 = C2 20 = 101100

15 = 001111

-15 = C2 15 = 110001



Agora, o resultado está incorreto, houve *overflow*, pois não houve vai um para o bit de sinal e para fora sim. Se convertermos o valor para decimal, podemos visualizar isso facilmente:
-20 - 15 = -35 ≠ 011101 = 29

Um computador tem um circuito especial para detectar qualquer condição de *overflow* quando dois números são somados ou subtraídos. Esse circuito de detecção enviará um sinal para a unidade de controle do computador, informando que ocorreu *overflow* e o resultado está incorreto (TOCCI, 2011).

Representação em Ponto Flutuante

Nas seções anteriores, vimos como trabalhar com os números em ponto fixo, ou seja, sem a parte fracionária.

Agora, passamos a trabalhar com a representação numérica em binário utilizando o chamado ponto flutuante, que nos auxilia a trabalhar com os números que possuem uma vírgula ou ponto separando a parte inteira da parte fracionária. A representação em ponto flutuante também é muito útil quando precisamos representar números muito grandes ou muito pequenos, conforme veremos mais a seguir.

Exemplos de números com ponto flutuante: 12,5; 11011,101; AF, E6.

Para realizar a conversão entre as bases decimal e binária, utilizamos um método muito parecido com o estudado anteriormente.

Conversão da base decimal para a base binária

Quando realizamos a conversão de números representados na base decimal para a base binária, devemos dividir o processo em duas partes:

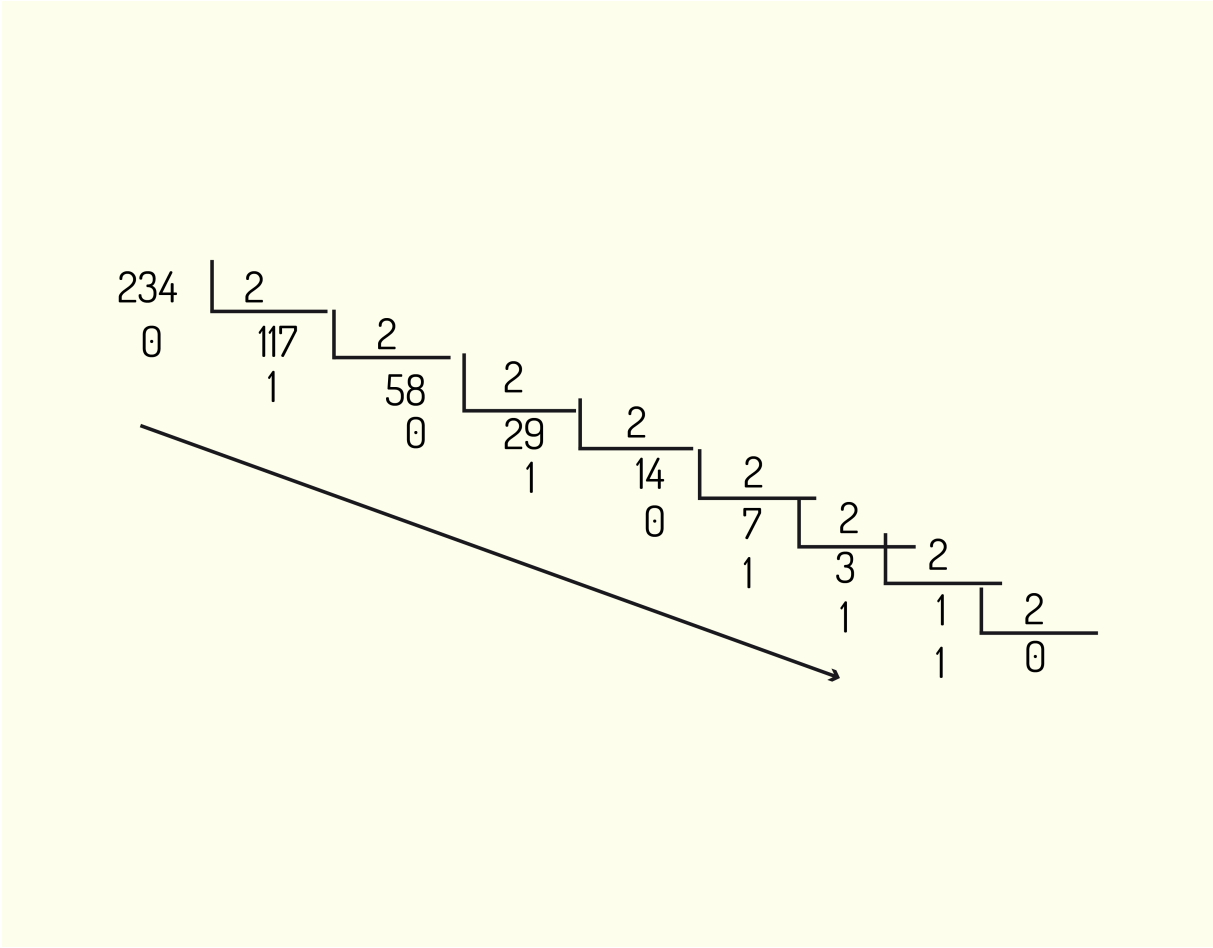
- Conversão da parte inteira;
- Conversão da parte fracionária.

Exemplo:

Converta o valor 234,75, dado em decimal, para binário.

Conversão da parte inteira

A conversão da parte inteira do número em ponto flutuante é realizada como já visto anteriormente. Chegamos à conclusão que a maneira mais fácil de realizar essa transformação é utilizando o método das divisões:



Assim, temos que $234_{10} = 11101010_2$.

Conversão da parte fracionária

Para realizar a conversão da parte fracionária, ao invés de realizarmos a divisão por 2, fazemos a multiplicação por esse mesmo valor e utilizamos a parte inteira do resultado, "pegando" os valores de cima para baixo, ou seja, na ordem normal:

	INTEIRO	FRAÇÃO	COEFICIENTE
0,75 * 2	1	0,50	a1 = 1
0,50 * 2	1	0,00	a2 = 1

1FIGURA 5.7 - Conversão da parte fracionária de 0,75 FONTE: A autora.

Realizamos esse processo até que a parte fracionária seja 0.

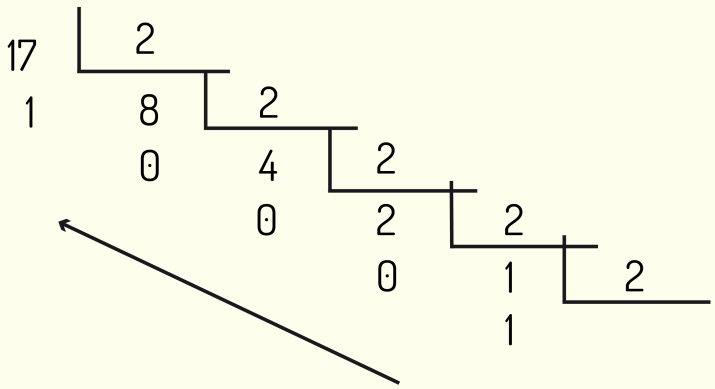
Assim, 0, 75 = 0, 11

Para finalizar, agrupamos os dois resultados: 234, 75 = 11101010, 11

Mas em alguns casos, não conseguimos finalizar o cálculo da parte fracionária com uma quantidade pequena de bits, ou, às vezes, isso não é possível realmente. Temos, como no sistema decimal, uma dízima periódica ou um número infinito de casas. Nessas situações, devemos limitar uma quantidade de bits para a representação e desprezar o restante.

Vejamos um segundo exemplo: converter o valor 17, 235_{10} para binário:

Conversão da parte inteira: $17_{10} = 10001_2$



Conversão da parte fracionária: $0{,}235_{10}$

	INTEIRO	FRAÇÃO	COEFICIENTE
0,235. 2	0	0,47	$a_1 = 0$
0,47. 2	0	0,94	$a_2 = 0$
0.94. 2	1	0,88	$a_3 = 1$
0,88. 2	1	0,76	$a_4 = 1$
0,76. 2	1	0,52	$a_5 = 1$
0,52. 2	1	0,04	$a_6 = 1$
0,04. 2	0	0,08	$a_7 = 0$
...	...	...	...

1FIGURA 6.7 - Conversão da parte fracionária de 0,235 FONTE: A autora.

Conforme podemos observar, o valor da parte fracionária reduziu bastante, assim ainda teremos várias operações até que ele se aproxime de 1 novamente. Além disso, pode ser que não fique exatamente 1 e tenhamos que começar o processo novamente.

É nesse tipo de caso que definimos o número de casas fracionárias que desejamos trabalhar e desprezamos as outras.

Assim, temos para o valor 17, 235:

- Com 7 casas fracionárias: 10001, 0011110...
- Com 5 casas fracionárias: 10001, 00111...
- Com 4 casas fracionárias: 10001, 0011...

Notamos que, quanto mais casas considerarmos após a vírgula, teremos uma maior precisão, ou seja, devemos aplicar o método até atingir a precisão desejada (IDOETA, 2012).

Conversão da base binária para a base decimal

A conversão de valores com ponto flutuante da base binária para a base decimal é feita utilizando-se o conceito de notação posicional, ou o método polinomial.

Mas agora, também devemos trabalhar com a parte fracionária.

Para a parte inteira, vamos aumentando o expoente da base da direita para a esquerda. Utilizando o mesmo raciocínio, para a parte fracionária, vamos diminuindo o expoente da esquerda para a direita.

Veja no exemplo: converter os valores a seguir, que estão em binário, para decimal:

a. 11010, 11 = $1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 + 1.2^{-1} + 1.2^{-2} = 16 + 8 + 0 + 2 + 0 + \frac{1}{2^1} + \frac{1}{2^2} = 26,75_{10}$

b. 1011, 0101 = $1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 0.2^{-3} + 1.2^{-4} = 8 + 0 + 2 + 1 + \frac{1}{2^2} + 0 + \frac{1}{2^4} = 11,3125_{10}$

c. 111, 1111 = $1.2^2 + 1.2^1 + 1.2^0 + 1.2^{-1} + 1.2^{-2} + 1.2^{-3} + 1.2^{-4} = 4 + 2 + 1 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} = 7,9375_{10}$

Representação em ponto flutuante

Muitas vezes, há a necessidade de representar números muito grandes ou muito pequenos, por exemplo, em decimal:

0,00000000000000000000000000000000000000000234

e

420000000000000000000000000000000000000000000.

Imagine converter esses valores para binário: quantos bits seriam precisos para representá-lo? Isso, no computador, é um problema muito sério, pois teríamos que ter registradores (que são os elementos que armazenam os números internamente para as operações realizadas no processador) e barramentos (que servem para transportar as informações) muito grandes também.

No sistema decimal, para representar números desse tipo, usamos a notação científica, que é facilmente visualizada nas calculadoras científicas, quando o número fica no formato, por exemplo, 123 E32.

A notação científica é representada da seguinte maneira:

N = ±F.B±E, onde se tem:

- **N:** número a ser representado;
- **±:** sinal do número;
- **F:** algarismos significativos do número (também chamado de mantissa);
- **B:** base da exponenciação; e
- **±E:** expoente, com seu devido sinal.

Podemos citar como exemplo, para os números apresentados anteriormente:

0, 00000000000000000000000000000000000000000234 = 0,234. 10-39 = 2, 34. 10-40 = 23,4 10-41 = 234. 10-42

42000000000000000000000000000000000000000000 = 42. 1040 = 4,2. 1041 = 0,42. 1042

Conforme podemos ver, existem várias maneiras de representação de um número utilizando notação científica.

Em binário, temos a mesma coisa:

11101,1011 = 0, 111011011. 2+101 = 111011011. 2-100

Segundo Monteiro (2012), a representação do expoente é definida pelo fabricante do processador. Pode-se usar representações do tipo sinal e magnitude, complemento de 2 e excesso de N. A representação sinal e magnitude é a mais simples de descrever e exemplificar, razão por que será utilizada... A representação usando-se complemento a 2 é implementada da mesma forma descrita anteriormente, sendo usada em alguns sistemas devido à rapidez da execução das operações de soma e subtração dos expoentes.

Portanto, para armazenar o número desejado, o sistema computacional deverá guardar: o sinal do número (que pode ser 0 ou 1), os dígitos significativos (ou mantissa) e o exponente acompanhado de seu sinal.

Monteiro (2012) também complementa:

A base de exponenciação não precisa ser armazenada para cada número (como o sinal, a mantissa e o expoente precisam), pois o sistema reconhece sua existência e valor por ser igual para qualquer número.

Desse modo, em uma representação de números em ponto flutuante, dois fatores são basicamente considerados:

- Precisão do número - expressa pela quantidade de algarismos da parte fracionária ou mantissa;
- Grandeza do número - expressa pelo valor do expoente.

E, o mesmo autor explica:

A precisão mede a exatidão do número... A grandeza, ou limite de representação, indica quão grande ou quão pequeno é o número que se deseja representar.

Como temos diferentes maneiras de representar um número utilizando a notação científica, é importante definir um padrão para sua utilização. Esse padrão é chamado de representação normalizada.

Stallings (2010) denomina a mantissa de significando. Para ele, um número normalizado é aquele em que o dígito mais significativo do significando é diferente de zero. Para a representação na base 2, um número normalizado é, portanto, um número em que o bit mais significativo do significando é 1.

O formato mais utilizado, didaticamente, para representar valores em ponto flutuante é o que utiliza 32 bits, apresentado a seguir:

S	EXPOENTE	MANTISSA
1 bit	7 bits	24 bits
32 bits		

Lembre-se que a mantissa sempre deverá estar normalizada.

Pode-se citar, como exemplo, a conversão do valor 17,23510, para binário, vista anteriormente. Mas agora vamos colocá-la no formato de ponto flutuante:

1. Conversão para binário (já realizada): 17, 235_{10} = 10001, 00111_2
2. Ajuste do expoente: 10001, 00111 = 10001, 00111 . 2^0 = 0, 1000100111 . 2^{+101}
3. Indicar os valores do formato:
   S = 0
   E = 0 000101
   M = 0, 1000100111

Para os números negativos, o processo é o mesmo, mas muda o bit de sinal:

Exemplo

Converter o valor -26, 625 para a notação binária em ponto flutuante:

2. Ajuste do expoente:
   11010, 101 = 11010, 101. 20 = 0, 11010101. 2+101
3. Indicar os valores do formato:
   S = 1
   E = 0 000101
   M = 0, 11010101

Padrão IEEE 754

Conforme vimos anteriormente, podemos utilizar diversas formas diferentes de representar um número em ponto flutuante. Por isso, em 1985, o IEEE (*Institute of Electrical and Electronics Engineers*) definiu o padrão IEEE 754, que é implementado na maioria dos microprocessadores.

Segundo Weber (2012), o formato reconhecido pelo IEEE é descrito a seguir (existem três formatos: simples, de 32 bits, duplo, de 64 bits e quádruplo, de 128 bits):

	CAMPOS	SIMPLES	DUPLO	QUÁDRUPLO
Campos	S = sinal	1 bit	1 bit	1 bit
	E = expoente	8 bits	11 bits	15 bits
	L = primeiro bit	(não representado)	(não representado)	1 bit
	F = fração	23 bits	52 bits	111 bits
Expoente	Excesso de	127	1023	16383
	Maior valor	255	2047	32767
	Menor valor	0	0	0

1FIGURA 7.7 - Padrão IEEE FONTE: Weber (2012).

Indicação de leitura

Nome do livro: Elementos de Eletrônica Digital

Editora: Érica

Autor: Francisco Gabriel Capuano / Ivan Valeije Idoeta

ISBN: 8571940193

O livro possui um conteúdo bem atual, abordando diversos elementos utilizados no sistema digital, com um capítulo exclusivo para sistemas de numeração e operações com o sistema binário. Possui vários exercícios resolvidos e outros propostos, com respostas, além da teoria ser explicada de forma simples e didática. É uma excelente opção para quem deseja estudar mais os conteúdos já trabalhados ou aprofundar seus conhecimentos.

Indicação de leitura

Nome do livro: Sistemas Digitais

Editora: Pearson

Autor: Ronal J. Tocci / Neal S. Widmer / Gregory L. Moss

ISBN: 978-85-7605-922-6

O livro possui uma linguagem fácil, abordando vários conteúdos da lógica digital estudados nessa unidade, como sistemas de numeração e aritmética digital. Além dos vários exercícios resolvidos e questões para revisão, também apresenta outros problemas/exercícios que podem ser utilizados para estudo e complementação dos temas vistos.

UNIDADE II

Introdução à Lógica Digital

Ana Cláudia de Oliveira Pedro Andrêo

Caro(a) aluno(a), na Unidade I, trabalhamos com os sistemas numéricos e com os principais métodos utilizados para realização de conversão entre eles.

A partir desse momento, passaremos realmente a perceber a necessidade de se trabalhar internamente com a representação binária das informações, aprendendo como esses dados são manipulados dentro do sistema computacional para geração dos resultados necessários e desejados.

Para isso, nesta unidade, vamos trabalhar com os elementos mais básicos que formam o sistema computacional: as portas lógicas. Veremos quais são as mais utilizadas e a função desempenhada por cada uma na máquina.

Também aprenderemos como simplificar as expressões que irão nos possibilitar implementar os circuitos que executam várias das atividades presentes no computador, utilizando a chamada Álgebra de Boole e os Mapas de Vetch-Karnaugh.

Finalizaremos com as noções básicas dos circuitos combinatórios, suas definições e principais usos.

Portas e Operações Lógicas

Apenas recordando, caro(a) aluno(a), temos quatro sistemas numéricos utilizados na área de computação: o sistema decimal, que é o utilizado pelo usuário, o sistema binário, utilizado pelo sistema computacional e os sistemas octal e hexadecimal. Como o sistema binário origina números com uma quantidade muito grande de dígitos, utilizamos os sistemas octal e hexadecimal como interface para que possamos trabalhar como o computador.

Atualmente, o sistema hexadecimal é o mais utilizado para esse interfaceamento com o sistema computacional e o sistema octal em circuitos eletrônicos. É importante lembrar que o computador não "pensa". Ele apenas realiza operações com os sinais recebidos e armazenados. Essas operações é que geram os resultados das operações aritméticas, lógicas e controlam todo o funcionamento interno do sistema computacional, que veremos na próxima unidade.

Segundo Hennessy (2014), a eletrônica digital trabalha com somente dois níveis de tensão de interesse: uma tensão alta e uma tensão baixa. Todos os outros valores de tensão são temporários e só ocorrem na transição entre dois valores válidos.

O autor complementa que:

> em várias das famílias lógicas, os valores e as relações entre as duas tensões são diferentes. Portanto, em vez de nos referirmos diretamente ao nível de tensão, falaremos de sinais que são (logicamente) verdadeiros, representados pelo valor binário 1, conhecidos também como ativos; ou sinais que são (logicamente) falsos, representados pelo valor binário 0, conhecidos como inativos .
>
> (HENNESSY, 2014, p.484)

As operações aritméticas e lógicas são realizadas pelas portas lógicas que, combinadas entre si, constituem os circuitos lógicos, que realizam as atividades desejadas.

Monteiro (2012, p.65) apresenta uma definição muito importante: *"uma porta (gate) é, então, um elemento de hardware (é um circuito eletrônico) que recebe um ou mais sinais de entrada e produz um sinal de saída, cujo valor é dependente do tipo de regra lógica estabelecida para a construção do referido circuito".*

O mesmo autor, também, complementa com outra informação, na mesma página:

> a porta lógica se constitui no elemento básico mais elementar de um sistema de computação. Grande parte do hardware do sistema é fabricado através da adequada combinação de milhões desses elementos, como a UCP, memórias principal e cache, interfaces de E/S e outros .
>
> (MONTEIRO, 2012, p.65)

Iniciaremos, agora, o estudo de diferentes tipos de portas lógicas e suas respectivas funções.

Tabela Verdade

Todos os resultados que podem ser encontrados após a implementação de uma porta ou circuito lógico podem ser mostrados em uma tabela, chamada de tabela verdade.

Ela apresenta todos os valores possíveis para os sinais de entrada e os que serão encontrados na saída, de acordo com a função utilizada.

De forma simplificada, a montagem de uma tabela verdade é feita da seguinte maneira: por meio do número de variáveis que serão utilizadas na entrada, define-se o número de linhas que ela irá ter, mediante a fórmula:

Número de combinações de entrada = 2^n, em que n é o número de variáveis.

Exemplo: se utilizarmos as variáveis A, B e C na entrada de um determinado circuito lógico, teremos a tabela verdade apresentada a seguir, com S representando o resultado final:

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

Verifique que, como temos 3 variáveis diferentes na entrada (A, B e C), o número de linhas que a tabela verdade possui é 8 (2^3). A base 2 refere-se justamente aos dois valores possíveis para cada variável de entrada, 0 e 1. Esse número de linhas representa todas as combinações diferentes que podemos ter entre os sinais de entrada.

Portas *NOT*

A porta *NOT* ou NÃO inverte o valor da variável de entrada, ou complementa o mesmo.

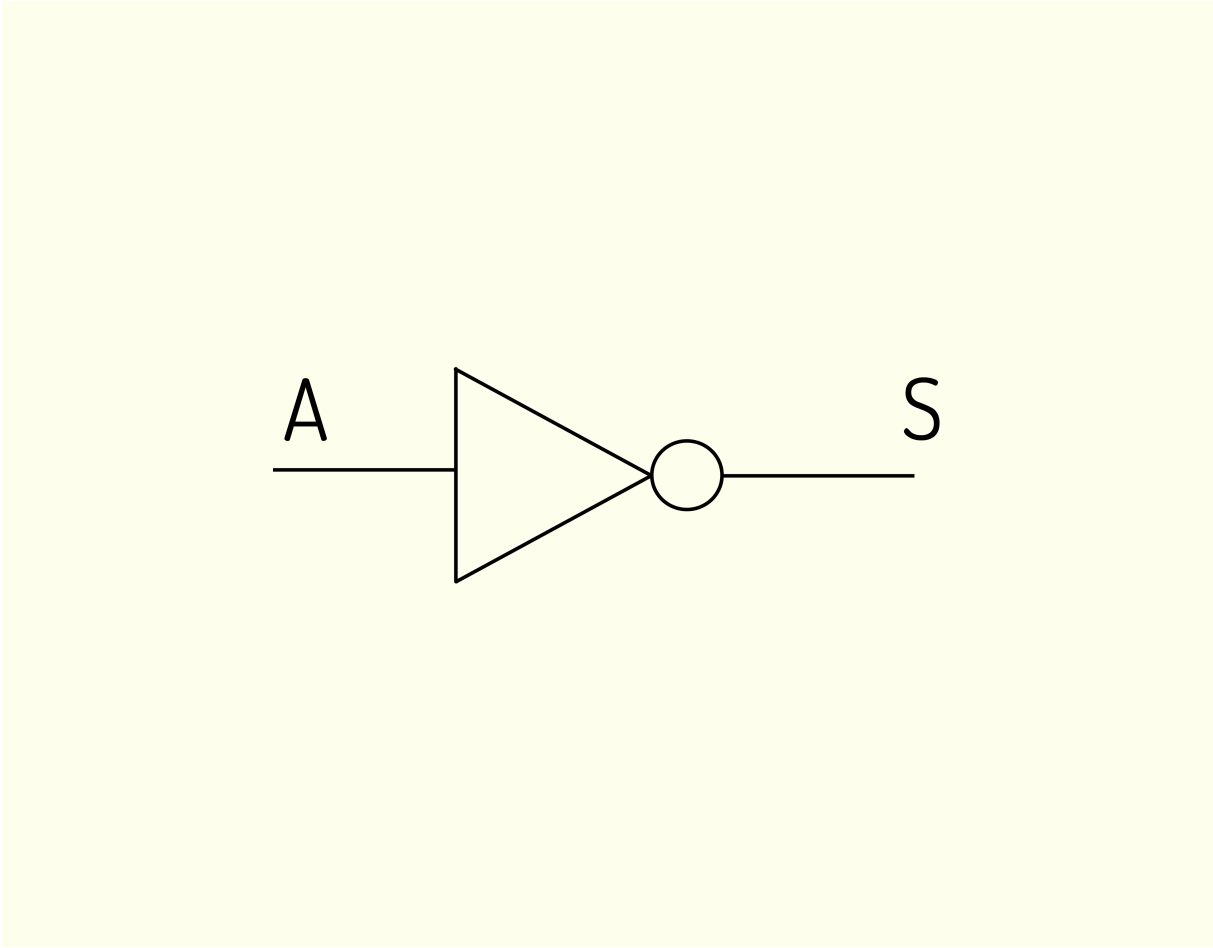
Ou seja, quando se tem o sinal lógico 0 na entrada, após a passagem pela porta *NOT* o sinal passará a ser 1 e se for 1, passará a ser 0.

A porta *NOT* é representada por $S = \bar{A}$, que se lê: S = A barrado, S = complemento de A, S = NOT A, dentre outras maneiras.

A porta *NOT* é também chamada de porta inversora (pois inverte os valores da entrada) e sua tabela verdade é:

A	$S = \bar{A}$
0	1
1	0

O símbolo dessa porta lógica é:



É importante lembrar que a porta *NOT* só possui uma entrada.

Portas *AND*

A porta *AND* é a responsável pela realização da operação lógica E.

A principal característica dessa porta lógica é que ela irá fornecer, na saída, o resultado 1 somente quando todas as suas entradas forem compostas por esse sinal digital. Caso contrário, o resultado será 0.

Algebricamente, para duas variáveis, a porta lógica *AND* é representada por S = A. B, que pode ser lida como S = A e B ou S = A *and* B.

Pela própria simbologia utilizada, pode-se ver que a porta *AND* é aquela que realiza a multiplicação entre duas variáveis booleanas, sua operação sendo chamada de produto lógico.

¶ Fique por dentro

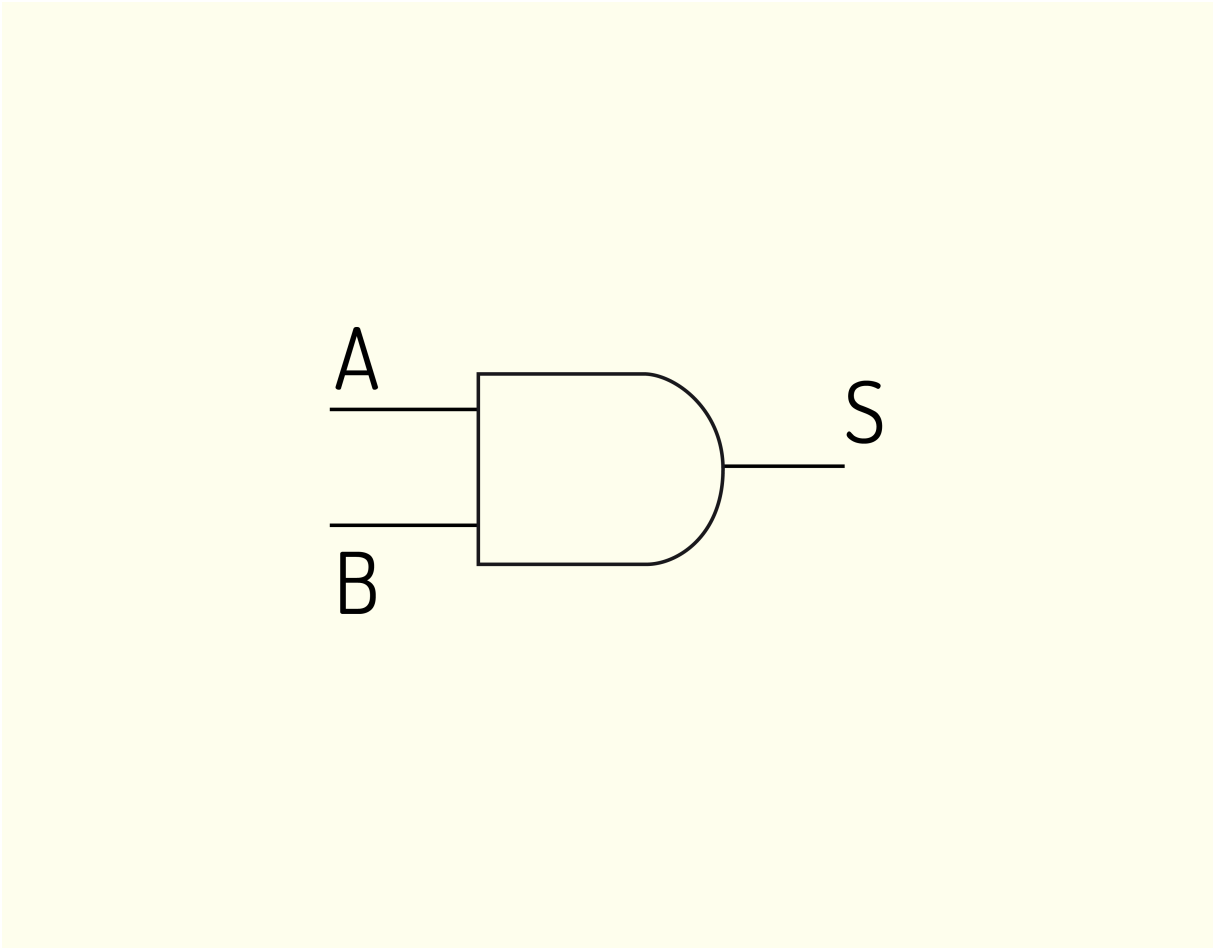
Álgebra booleana

Segundo Monteiro (2012), assim como na álgebra comum, a álgebra booleana trata de variáveis e de operações a serem realizadas com essas variáveis. A diferença é que, no que se refere à álgebra booleana, as variáveis usadas são binárias, tendo apenas dois valores possíveis, VERDADE – V (equivalente ao bit 1) e FALSO – F (equivalente ao bit 0).

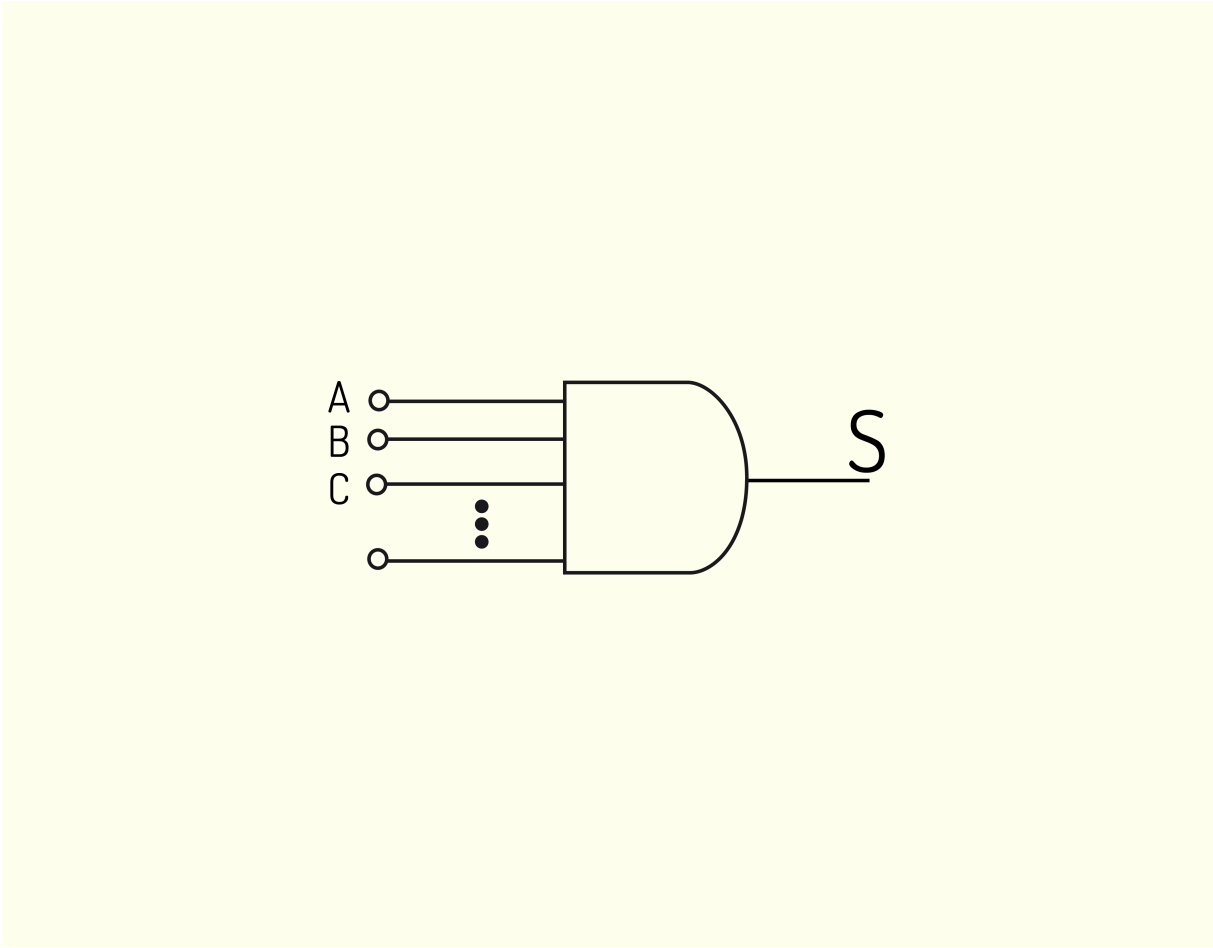
A tabela verdade para a função *AND* com duas variáveis é:

A	**B**	**S = A. B**
0	0	0
0	1	0
1	0	0
1	1	1

A porta lógica *AND* é quem executa a função mostrada e é representada por:



Pode-se estender o conceito da função E para qualquer número de entradas. Levando-se isso em consideração, tem-se a figura a seguir para sua representação:



Portas *OR*

A porta lógica *OR* é quem realiza a operação lógica OU.

Assim como a porta *AND*, ela também tem uma característica que a representa: ela irá fornecer, na saída, o resultado 0 somente quando todas as suas entradas forem compostas por esse sinal digital. Caso contrário, o resultado será 1.

Algebricamente, para duas variáveis, a porta lógica *OR* é representada por S = A + B, que pode ser lida como S = A ou B ou S = A *or* B.

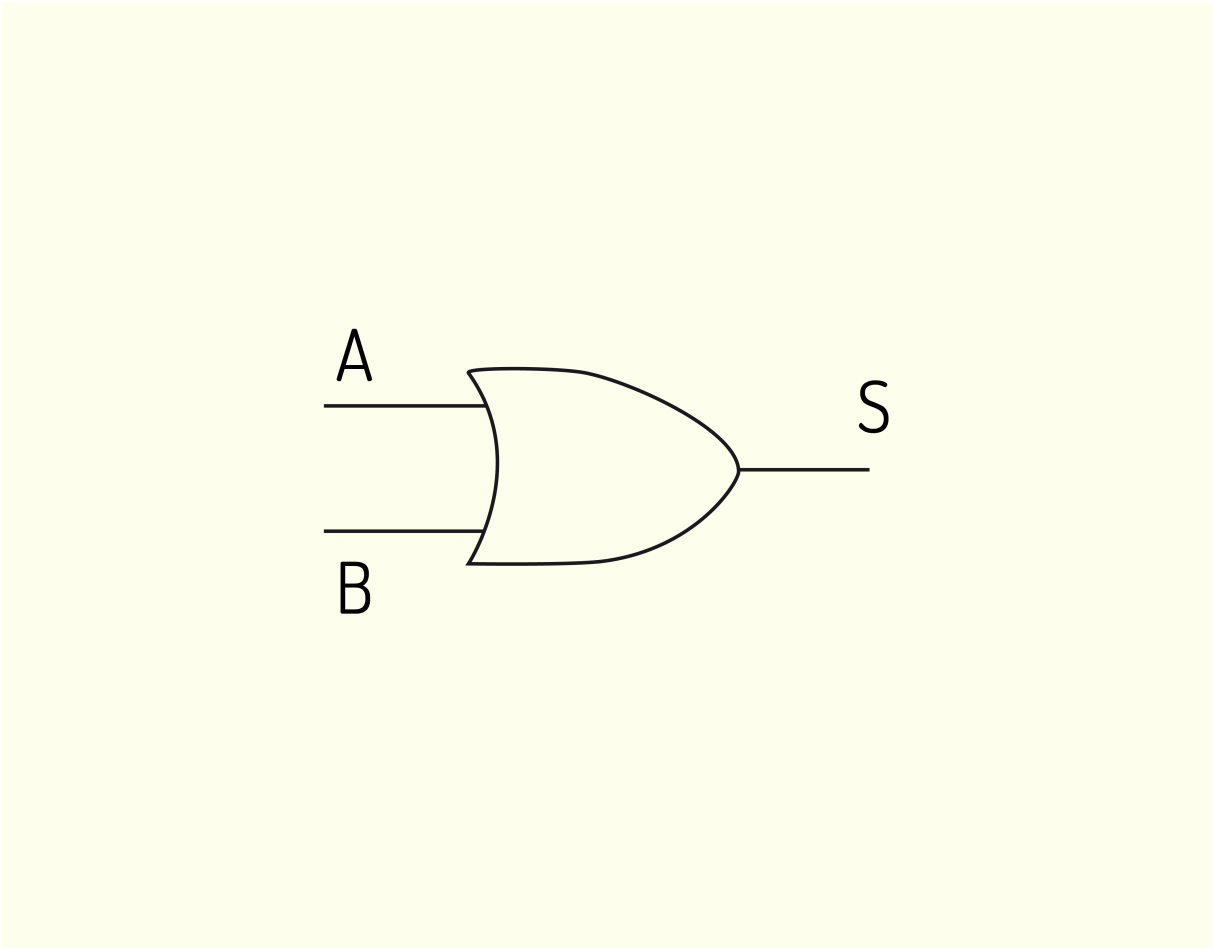
A simbologia da porta *OR* indica que ela é quem realiza a soma entre duas variáveis booleanas, sendo chamada de soma lógica. Mas muito cuidado: na realidade quem realiza a soma entre dois bits mesmo é a porta XOR, que será vista mais adiante.

A tabela verdade que representa a porta *OR* é:

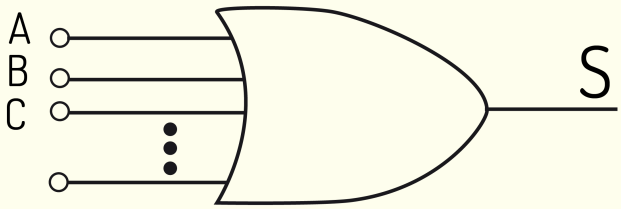
A	B	S = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Conforme estudamos na unidade I, se somarmos 1 + 1 obteremos 0 e vai um e não 1, como a tabela mostra, o que comprova que não é essa realmente a porta lógica que realiza a soma entre dois bits.

A porta lógica *OR* é quem executa a função mostrada e é representada por:



Assim como com a porta *AND*, pode-se estender o conceito da função *OR* para qualquer número de entradas. Considerando-se um número maior de entradas, tem-se a figura a seguir para sua representação:



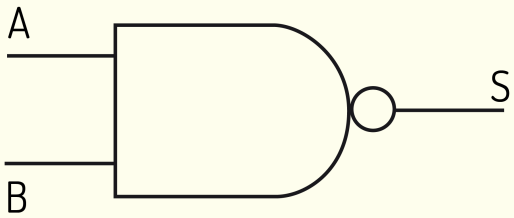
Portas NAND

A porta lógica NAND é uma junção da porta *NOT* com a *AND*, ou seja, pode-se falar que ela é a inversão da porta *AND*. Essa porta também é chamada de NE (NÃO E).

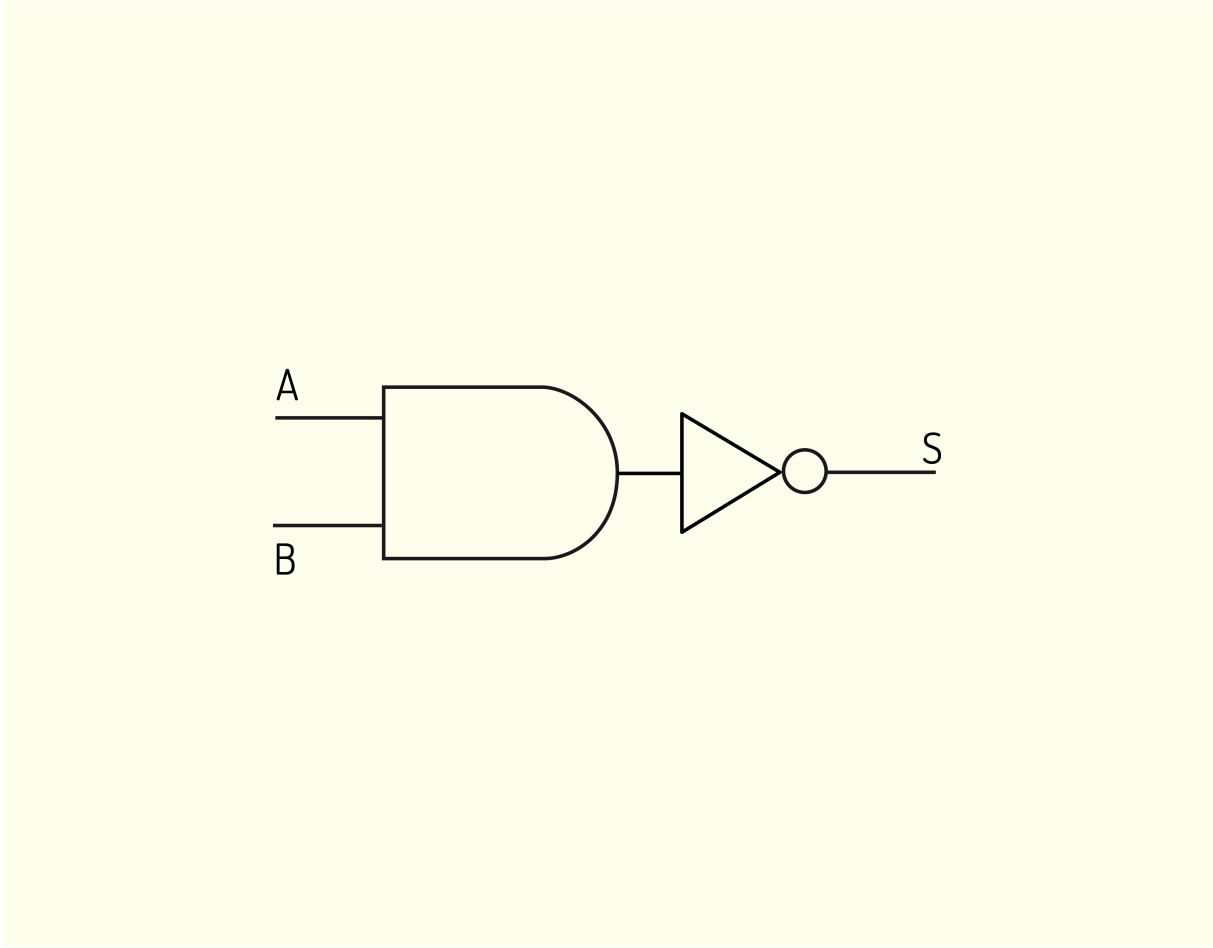
Sua representação algébrica é $S = \overline{A.B}$ e sua tabela verdade representada por:

A	B	S = A.B	$S = \overline{A.B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A representação gráfica da porta NAND é:



Em algumas montagens de circuitos lógicos, nem sempre é interessante utilizar muitas portas lógicas diferentes. Portanto, se não tivermos a porta NAND, podemos utilizar uma porta *NOT* acoplada a uma porta *AND*, como mostrado na figura a seguir:



Essa porta lógica também pode ter mais do que duas entradas.

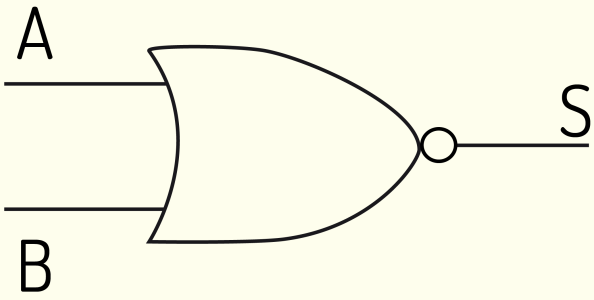
Portas NOR

Assim como a porta NAND, a porta NOR também é uma junção de duas portas distintas, mas neste caso, temos a porta *NOT* ligada a uma porta *OR*. Portanto, pode-se dizer que a porta NOR é a inversão da porta *OR*, podendo ser chamada, também, de porta NOU (NÃO OU).

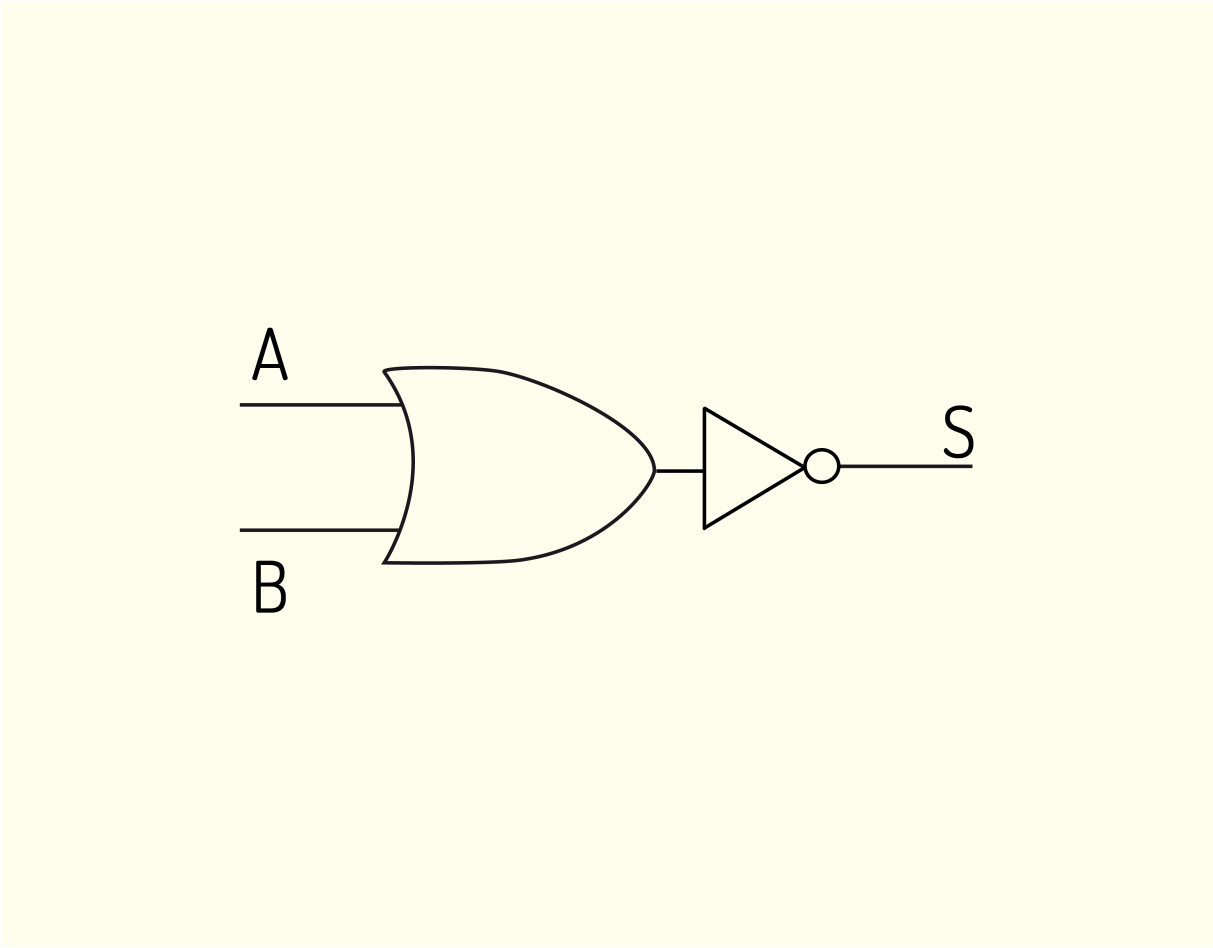
Sua representação algébrica é $S = \overline{A + B}$ e sua tabela verdade representada por:

A	B	S = A + B	$S = \overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Graficamente, podemos representar a porta lógica NOR como:



Quando a porta NOR não estiver disponível, podemos acoplar uma porta *NOT* a uma porta *OR*, obtendo o mesmo resultado desejado:



Assim como as outras portas lógicas apresentadas anteriormente, a porta NOR pode ter duas ou mais entradas.

Portas XOR

A porta XOR é chamada de *Exclusive OR* ou OU exclusivo.

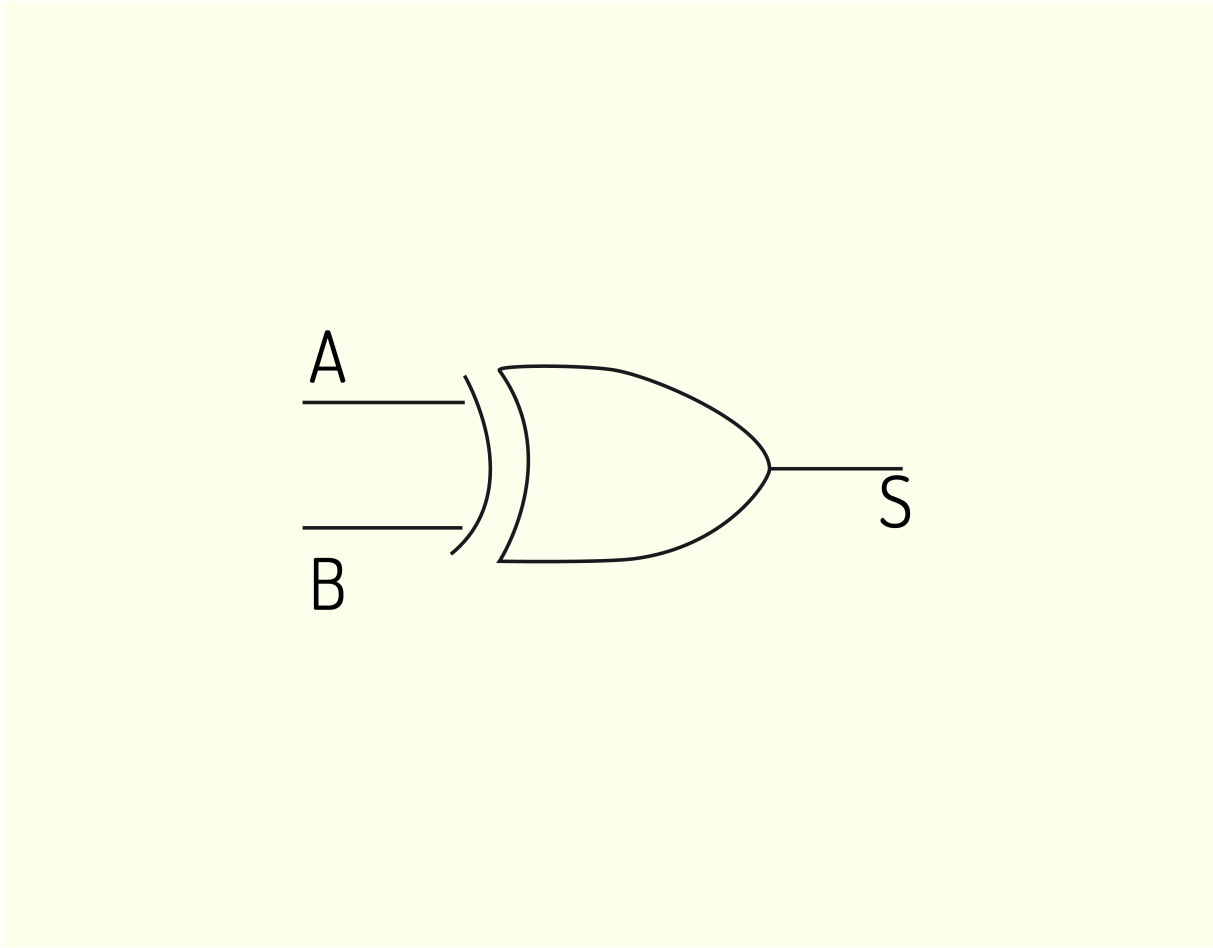
Sua função é fornecer o valor lógico 1 à saída quando tivermos valores diferentes nas variáveis de entrada.

Algebricamente, representamos a porta lógica XOR como $S = A\,B$. Sua tabela verdade é:

A	B	$S = A\,B$
0	0	0
0	1	1
1	0	1
1	1	0

Conforme comentamos anteriormente, essa é a porta lógica que realmente efetua a soma entre dois dígitos binários. Isso pode ser observado pelos resultados obtidos na tabela verdade.

Graficamente, a porta XOR é representada por:



Diferente das portas apresentadas anteriormente, a porta XOR só pode possuir duas variáveis de entrada.

Uma das principais aplicações dessa porta lógica é a implementação de parte do circuito de soma na Unidade Aritmética e Lógica (UAL).

¶ Reflita

Uma maneira fácil de entender o resultado fornecido pelas portas lógicas é fazendo uma analogia com a lógica matemática. Esta lógica, assim como a booleana, também trabalha com somente dois valores: verdadeiro e falso.

Monteiro (2012) relaciona algumas aplicações para algumas das portas lógicas estudadas:

- Porta *AND*: ativação de uma linha de dados para movimentar bits de um registrador (ou células) para outro;
- Porta *NOT*: operações aritméticas em ponto fixo, quando se usa aritmética de complemento;

- Porta XOR: fabricação de um testador de igualdade entre valores, para testar se duas palavras de dados são iguais ou se os sinais de dois valores são iguais ou não.

Noções de Álgebra Booleana

Como dissemos anteriormente, as portas lógicas são os elementos mais primários presentes em um sistema computacional, que devem ser agrupadas nos circuitos lógicos ou digitais a fim de que as atividades desse sistema sejam realizadas.

Para que se possa realizar o projeto de um circuito lógico, segundo Güntzel (2001), alguns passos devem ser seguidos:

1. Escolher um símbolo para cada variável de entrada e para cada variável de saída;
2. A partir da especificação do problema, determinar a tabela verdade (caso ela já não faça parte da especificação do problema);
3. Obter as equações simplificadas;
4. Mapear o circuito para a biblioteca de portas disponível (se for o caso);
5. Desenhar o circuito final.

Vamos utilizar o exemplo a seguir, idealizado pelo mesmo autor, para mostrar esses passos.

Exemplo: projetar a parte inicial de um circuito que recebe um inteiro binário de 3 bits e determina se este número é menor ou igual a 3.

Iniciamos definindo para os símbolos de cada variável de entrada os nomes A_2, A_1 e A_0. A variável de saída será chamada de S.

Construímos a tabela verdade:

A_2	A_1	A_0	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Por meio de técnicas de definição de circuitos, a expressão que representa esse circuito será:

$$S = \overline{A_2}\,\overline{A_1}\,\overline{A_0} + \overline{A_2}\,\overline{A_1}\,A_0 + \overline{A_2}\,A_1\,\overline{A_0} + \overline{A_2}\,A_1\,A_0$$

Mas ao obtermos o circuito final para a resolução do problema, nem sempre ele está da forma mais otimizada que pode ser apresentada, como o que foi obtido no exemplo anterior. Assim, há a necessidade de simplificarmos esse circuito, como fazemos em várias expressões matemáticas.

Quando simplificamos um circuito, obtemos outro chamado de equivalente. É muito importante entender a diferença entre igual e equivalente. O circuito equivalente é aquele que executa a mesma função do original, mas é diferente dele. Ou seja, tem-se a mesma saída na tabela verdade, com um circuito mais simples.

Essa etapa de simplificação dos circuitos digitais é extremamente importante, uma vez que, devido a restrições de espaço e necessidade de velocidade no sistema computacional, o circuito a ser utilizado deverá ser o mais simples e otimizado possível.

Existem duas maneiras principais de se simplificar os circuitos lógicos: utilizando os princípios da Álgebra de Boole e os Diagramas de Vetch-Karnaugh, que serão vistos logo a seguir.

Tocci (2011, p.49) fala um pouco sobre a origem da álgebra booleana:

> Em 1854, um matemático chamado George Boole escreveu Uma Investigação das leis do pensamento, em que descrevia o modo como se toma decisões lógicas com base em circunstâncias verdadeiras ou falsas. O método que ele descreveu é hoje conhecido como lógica booleana, e o sistema que emprega símbolos e operadores para descrever essas decisões é chamado de álgebra booleana.

Portanto, para que possamos reduzir, ou simplificar um circuito, utilizando a Álgebra de Boole, é necessário conhecer seus postulados, propriedades, teorema e identidades, para que possam ser aplicadas no processo.

Postulados da Álgebra Booleana

Significados (2016, *on-line*) apresenta a seguinte definição:

> Postulado é uma sentença que não é provada ou demonstrada, e por isso se torna óbvia ou se torna um consenso inicial para a aceitação de uma determinada teoria. O postulado não é necessariamente uma verdade muito clara, é uma expressão formal usada para deduzir algo, a fim de obter um resultado mais facilmente, através de um conjunto de sentenças. O postulado é uma proposição que, apesar de não ser evidente, é considerada verdadeira sem discussão.

A Álgebra Booleana possui três postulados:

Postulado da Complementação

O postulado da complementação é descrito da seguinte maneira, chamando-se de $\bar{A}$ o complemento de A:

1. Se A = 0, $\bar{A}$ = 1
2. Se A = 1, $\bar{A}$ = 0

Utilizando-se este postulado, pode-se chegar à seguinte identidade:

$$\bar{A} = A$$

Postulado da Adição

No caso da adição, utilizando-se a porta *OR*, temos:

1. 0 + 0 = 1
2. 0 + 1 = 1
3. 1 + 0 = 1
4. 1 + 1 = 1

Assim, podemos chegar às identidades:

1. A + 0 = A
2. A + 1 = 1
3. A + A = A
4. A + $\bar{A}$ = 1

Postulado da Multiplicação

Utilizando o mesmo raciocínio que no postulado da adição, mas agora pensando na multiplicação booleana e utilizando a tabela verdade da porta *AND*, tem-se:

1. 0. 0 = 0
2. 0. 1 = 0
3. 1. 0 = 0
4. 1. 1 = 1

Agora, as identidades obtidas são:

1. A. 0 = 0
2. A. 1 = A
3. A. A = A
4. A. $\bar{A}$ = 0

Propriedades da Álgebra Booleana

Assim como a álgebra tradicional, a álgebra booleana também possui algumas propriedades, muito semelhantes. São elas:

Propriedade Comutativa

A propriedade comutativa é válida tanto para a adição quanto para a multiplicação.

Para a adição, tem-se:

$$A + B = B + A$$

E, para a multiplicação,

$$A. B = B. A$$

Propriedade Associativa

A propriedade associativa também é válida para a adição e multiplicação. Tem-se:

Para a adição:

$$A + (B + C) = (A + B) + C = A + B + C$$

Para a multiplicação:

$$A. (B. C) = (A. B). C = A. B. C$$

Propriedade Distributiva

A propriedade distributiva indica que:

$$A. (B + C) = A. B + A. C$$

Teoremas de De Morgan

Segundo Idoeta (2012), os teoremas de De Morgan são muito empregados na prática, em simplificações de expressões booleanas e, ainda, no desenvolvimento de circuitos digitais.

Primeiro teorema de De Morgan

O primeiro teorema de De Morgan afirma que: o complemento do produto é igual à soma dos complementos.

Assim:

$$\overline{(A\,.\,B)} = \bar{A} + B$$

Segundo teorema de De Morgan

O segundo teorema de De Morgan afirma que: o complemento da soma é igual ao produto dos complementos.

Assim:

$$\overline{(A + B)} = \bar{A}.\,B$$

Lembre-se: os dois teoremas apresentados são válidos para qualquer número de variáveis:

$$\overline{(A\,.\,B\,.\,C\ldots N)} = \bar{A} + \overline{B} + \overline{C} + \ldots + \overline{N}$$

$$\overline{(A + B + C + \ldots + N)} = \bar{A}\,.\,\overline{B}\,.\,\bar{C}\ldots\,\bar{N}$$

Identidades Auxiliares

1. A + A. B = A
2. (A + B). (A + C) = A + B. C
3. A + $\bar{A}$. B = A + B

Essas três identidades podem ser demostradas utilizando-se os conhecimentos adquiridos até o momento.

¶ Reflita

Todos os postulados, identidades, propriedades e teoremas da Álgebra de Boole podem ser provados utilizando-se as tabelas verdade. Este é um excelente exercício para que se relembre os resultados que podem ser obtidos nas portas lógicas.

Vamos utilizar o exemplo apresentado no início desta seção para mostrar como a álgebra de Boole pode ser usada.

Obtivemos, para o nosso circuito que determina se um determinado número inteiro binário de 3 bits é menor ou igual a 3, a seguinte expressão booleana:

$$S = \overline{A_2}\,\overline{A_1}\,\overline{A_0} + \overline{A_2}\,\overline{A_1}\,A_0 + \overline{A_2}\,A_1\,\overline{A_0} + \overline{A_2}\,A_1\,A_0$$

Utilizando a álgebra booleana para simplificar a expressão, temos:

- Usando a propriedade distributiva: $S = \overline{A_2}\left(\overline{A_1}\,\overline{A_0} + \overline{A_1}\,A_0 + A_1\,\overline{A_0} + A_1\,A_0\right)$
- Usando novamente a propriedade distributiva: $S = \overline{A_2}\left[\overline{A_1}\left(\overline{A_0} + A_0\right) + A_1\left(\overline{A_0} + A_0\right)\right]$
- Usando a Identidade da Adição (A + $\bar{A}$ = 1): $S = \overline{A_2}\left[\overline{A_1}\,(1) + A_1\,(1)\right]$
- Usando a Identidade da Multiplicação (A. 1 = A): $S = \overline{A_2}\left[\overline{A_1} + A_1\right]$
- Usando novamente a Identidade da Adição (A + $\bar{A}$ = 1): $S = \overline{A_2}\,[1]$
- E, finalmente, utilizando novamente a Identidade da Multiplicação (A. 1 = A): $S = \overline{A_2}$

É nítido que o circuito obtido após a simplificação da expressão booleana original é muito mais simples do que o projetado anteriormente. Na realidade, agora, temos somente uma ligação realizada diretamente ao sinal A_2, não havendo nem a necessidade de se trabalhar com portas lógicas.

- Mais um exemplo: simplifique a expressão booleana: $S = \overline{\left(\overline{AB} + C + D\right)} + B\overline{(A.B.D)}$
- Utilizando o 1° Teorema de De Morgan ($\overline{(A\ .\ B)} = \bar{A} + \bar{B}$): $S = \overline{(\bar{A} + \bar{B} + C + D)} + B\,(\bar{A} + \bar{B} + \bar{D})$
- Utilizando o 2° Teorema de De Morgan ($\overline{(A+B)} = \bar{A}.\ \bar{B}$): $S = A\ .\ B\ .\ \bar{C}\ .\ \bar{D} + B\,(\bar{A} + \bar{B} + \bar{D})$
- Utilizando a Identidade da Complementação ($A = A$)): $S = A\ .\ B\ .\ \bar{C}\ .\ \bar{D} + B\,(\bar{A} + \bar{B} + \bar{D})$
- Utilizando a Propriedade Distributiva: $S = A\ .\ B\ .\ \bar{C}\ .\ \bar{D} + B\ .\ \bar{A} + B\ .\ \bar{B} + B\ .\ \bar{D}$
- Utilizando a Identidade da Multiplicação (A. $\bar{A}$ = 0): $S = A\ .\ B\ .\ \bar{C}\ .\ \bar{D} + B\ .\ \bar{A} + 0 + B\ .\ \bar{D}$
- Utilizando a Propriedade Distributiva: $S = B\overline{D}\ (A\ .\ \bar{C} + 1) + B\ .\ \bar{A}$
- Utilizando a Identidade da Adição (A + 1 = 1): $S = B\overline{D}\ .\ 1 + B\ .\ \bar{A}$
- Utilizando a Identidade da Multiplicação (A. 1 = A): $S = B\overline{D} + B\ .\ \bar{A}$
- Utilizando a Propriedade Distributiva: $S = B\left(\overline{D} + \bar{A}\right)$
- Se usarmos, também, o 1° Teorema de De Morgan $\overline{(A\ .\ B)} = \bar{A} + \bar{B}$)), obtemos: $S = B\left(\overline{D\ .\ A}\right)$

A seguir, é apresentado um quadro com o resumo das principais leis da Álgebra de Boole.

POSTULADOS		
COMPLEMENTAÇÃO	**ADIÇÃO**	**MULTIPLICAÇÃO**
A = 0 □ $\bar{A}$ = 1	0 + 0 = 0	0. 0 = 0
A = 1 □ $\bar{A}$ = 0	0 + 1 = 1	0. 1 = 0
	1 + 0 = 1	1. 0 = 0
	1 + 1 = 1	1. 1 = 1
IDENTIDADES		
COMPLEMENTAÇÃO	**ADIÇÃO**	**MULTIPLICAÇÃO**
$A = A$	A + 0 = A	A. 0 = 0
	A + 1 = 1	A. 1 = A
	A + A = A	A. A = A
	A + $\bar{A}$ = 1	A. $\bar{A}$ = 0
PROPRIEDADES		
Comutativa:	A + B = B + A A. B = B. A	
Associativa:	A + (B + C) = (A + B) + C = A + B + C A. (B. C) = (A. B). C = A. B. C	
Distributiva:	A. (B + C) = A. B + A. C	
TEOREMAS DE MORGAN		
$\overline{(A\ .\ B)} = \bar{A} + \bar{B}$		
$\overline{(A+B)} = \bar{A}.\ \bar{B}$		
IDENTIDADES AUXILIARES		
A + A. B = A		
$A + \overline{A}B = A + B$		
(A + B). (A + C) = A + B. C		

2FIGURA 1.1 - Principais leis da Álgebra de Boole FONTE: Idoeta (2012).

Diagramas ou Mapas de Vetch-Karnaugh

Os diagramas, ou também chamados mapas de Vetch-Karnaugh, ou simplesmente Mapas de Karnaugh, é uma outra forma que pode ser utilizada para realizar a simplificação de expressões booleanas, de uma forma muito mais rápida.

Tocci (2011) afirma que o mapa de Karnaugh (mapa K) é um método gráfico usado para simplificar uma equação lógica ou para converter uma tabela verdade no circuito lógico correspondente, de maneira simples e metódica. Embora um mapa de Karnaugh possa ser usado em problemas que envolvem qualquer número de variáveis, sua utilidade prática está limitada a cinco ou seis variáveis.

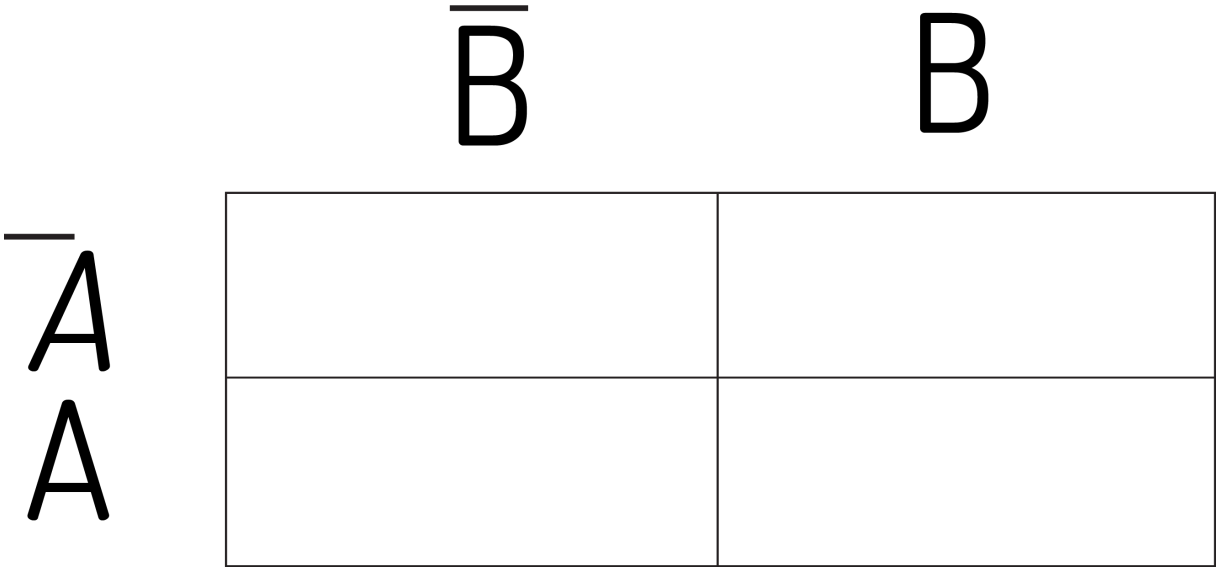
Apesar de ser um método bem mais simples do que a álgebra booleana, visto não precisarmos "pensar" muito para realizar as reduções, pois o processo é bem automático e visual, eles não são apropriados para a simplificação de todos os tipos de expressões, como no caso da função XOR (ou exclusivo).

Nesta seção, iremos estudar os mapas de Karnaugh para 2, 3 e 4 variáveis.

Mapas de Karnaugh para 2 variáveis

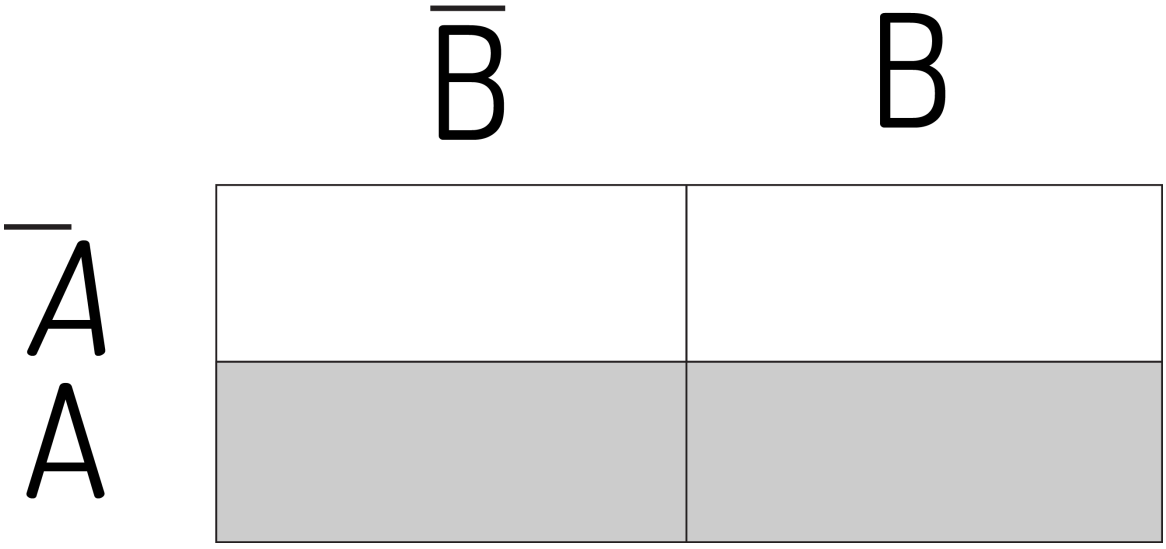
Existem diferentes maneiras de colocar as variáveis em um mapa de Karnaugh. Aqui iremos utilizar uma delas, mas outras também podem ser escolhidas, mudando-se somente a disposição dos termos.

A forma do diagrama para duas variáveis é:

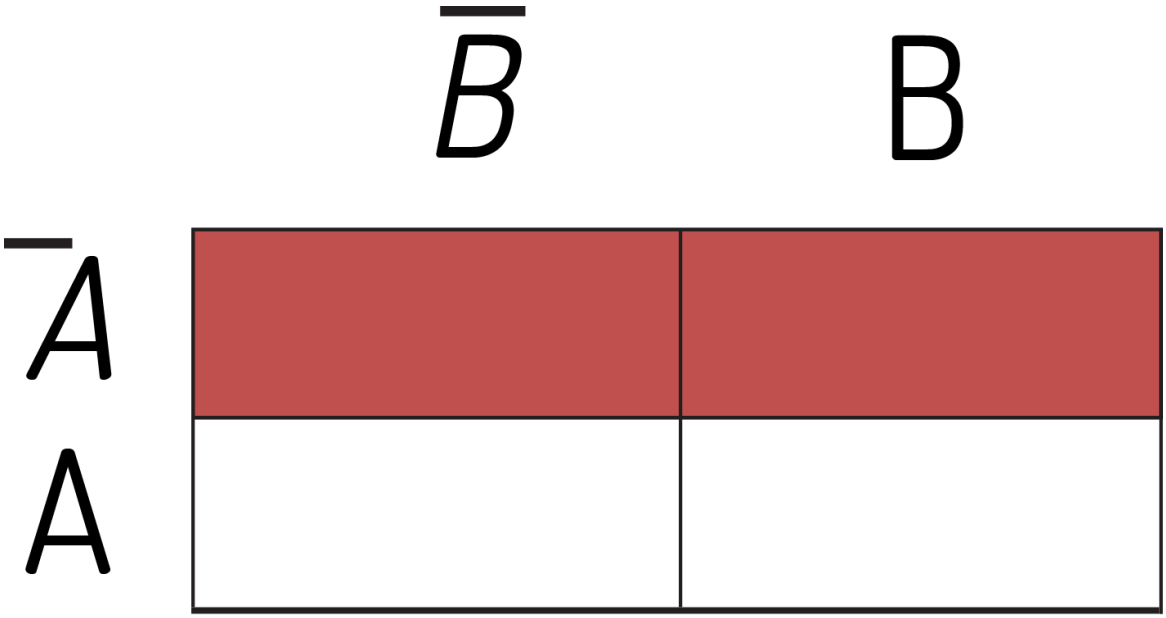


Neste formato, podemos encontrar todas as possibilidades que as variáveis escolhidas podem assumir:

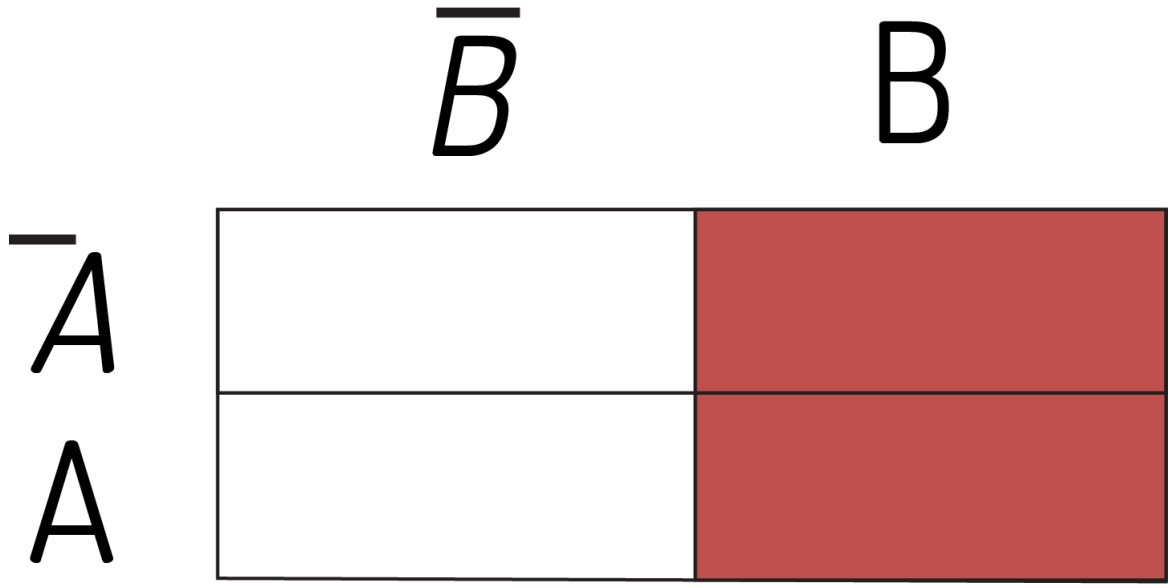
a. A = 1



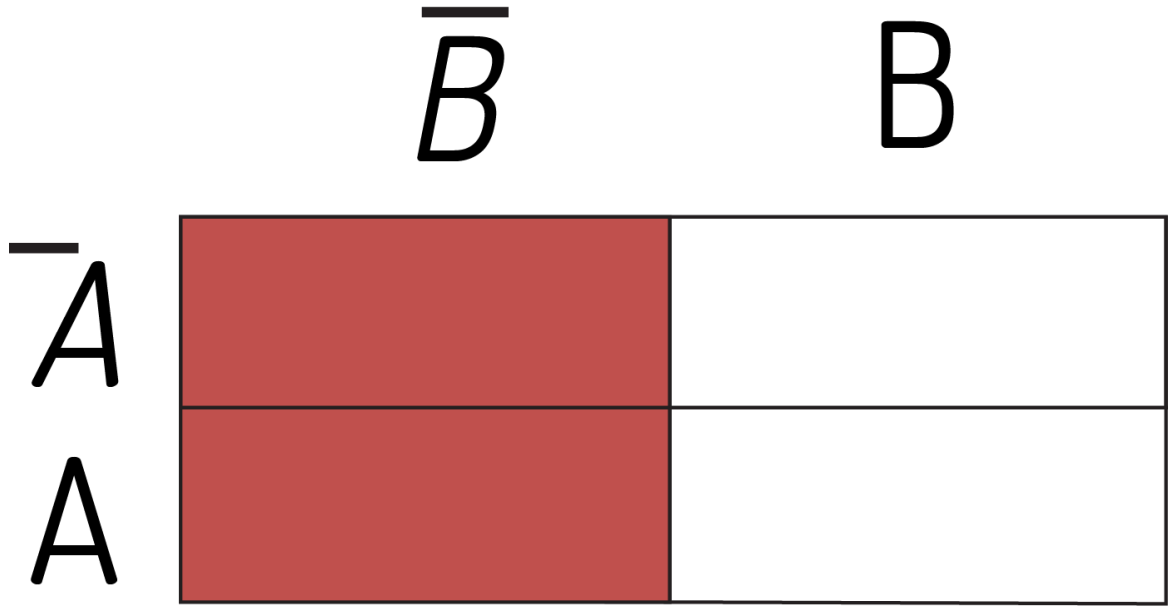
b. $\bar{A} = 1$ ou A = 0



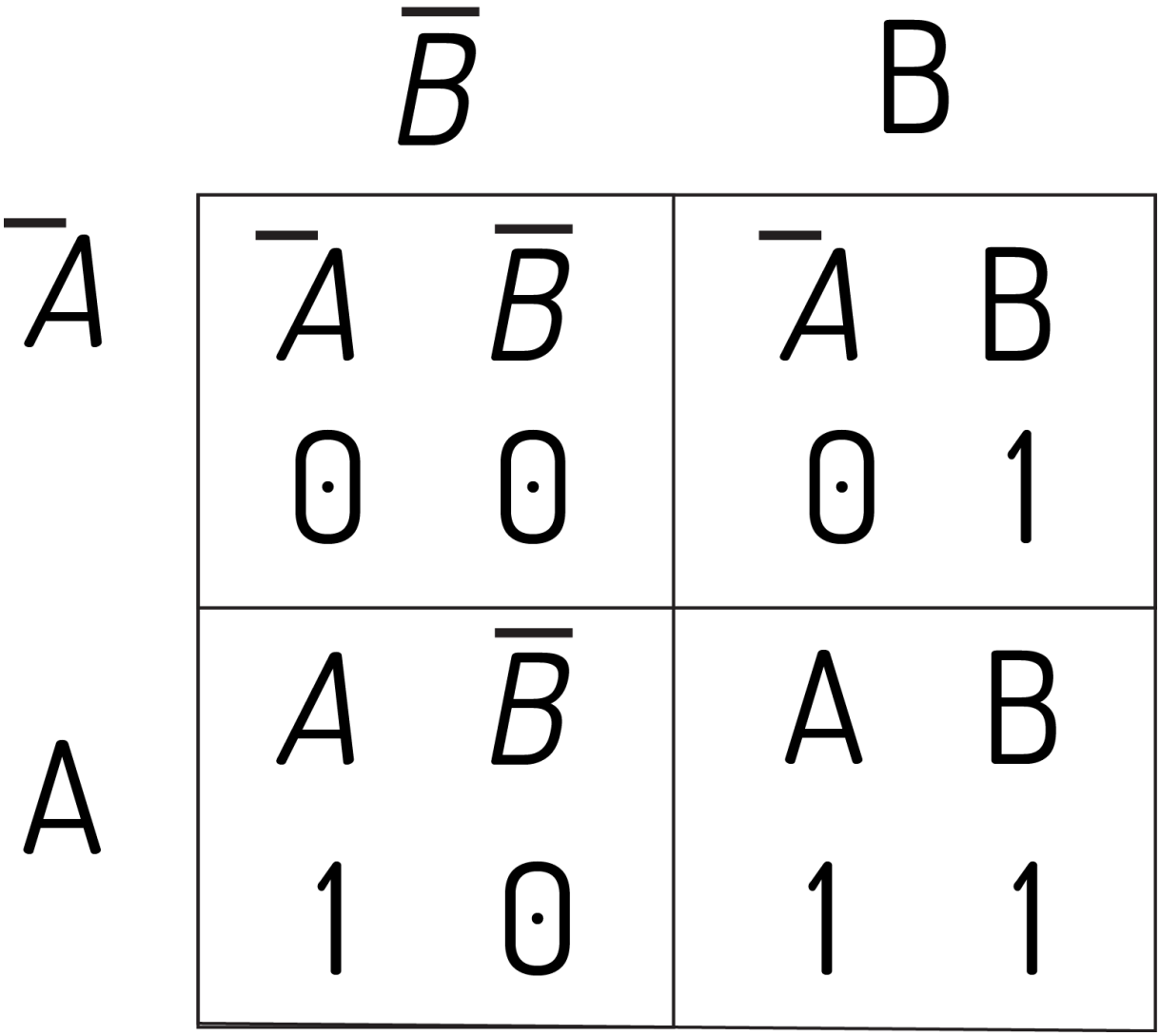
c. B = 1



d. $B = 1$ ou B = 0



Cada uma das células existentes no diagrama indica uma possibilidade que pode ocorrer entre as variáveis A e B, ou seja, uma linha da tabela verdade. Neste caso, temos:



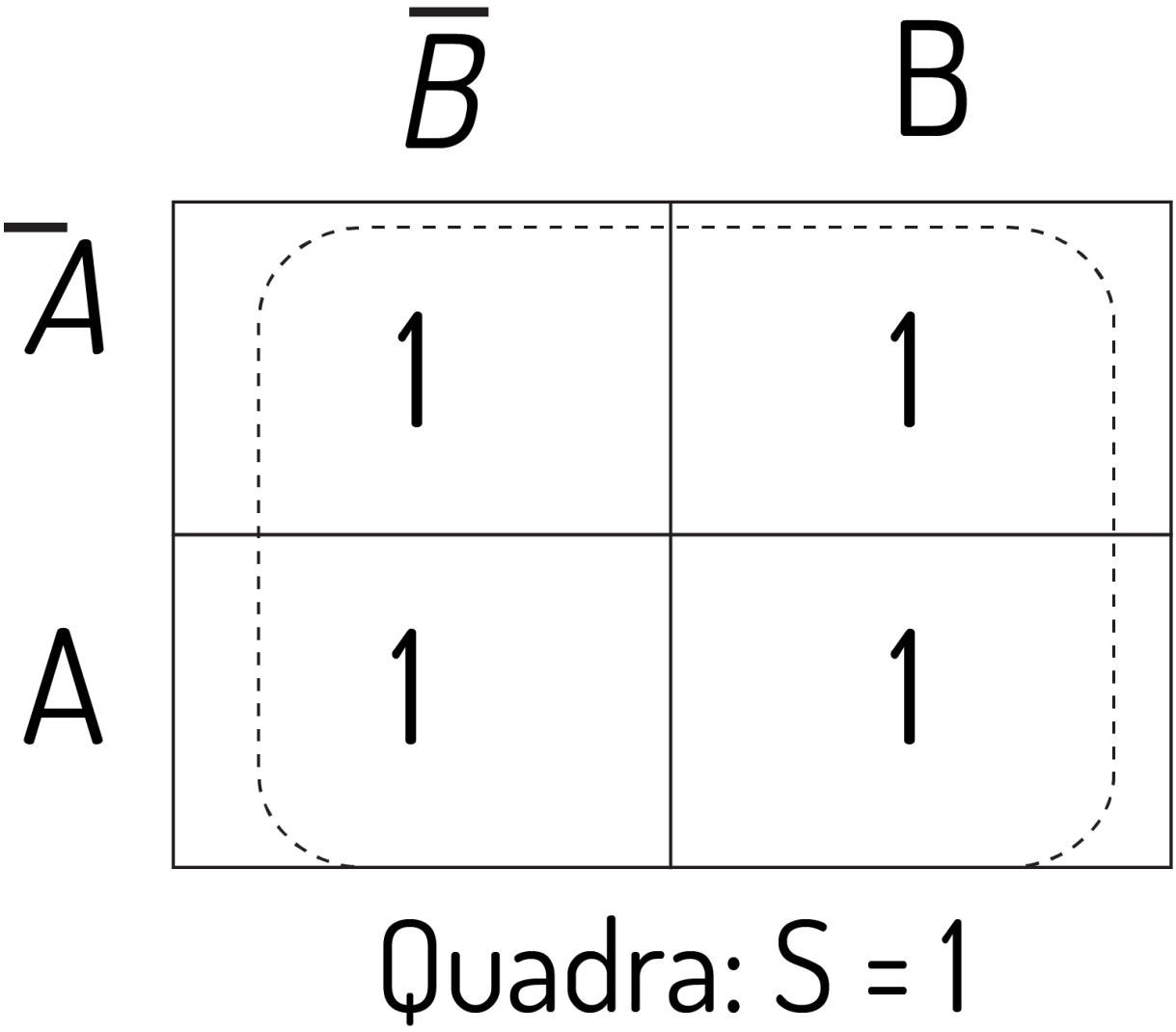
Utilização do mapa:

1. Coloque todos os valores presentes na expressão booleana nas posições adequadas do mapa, representando-as com 1;
2. Agrupe as regiões onde S = 1 (sempre procure agrupar o maior número possível de 1s de cada vez).

Dependendo do número de variáveis trabalhadas, tem-se diferentes tipos de agrupamentos. Para 2 variáveis, os agrupamentos são a quadra, os pares e os termos isolados.

A. Quadra

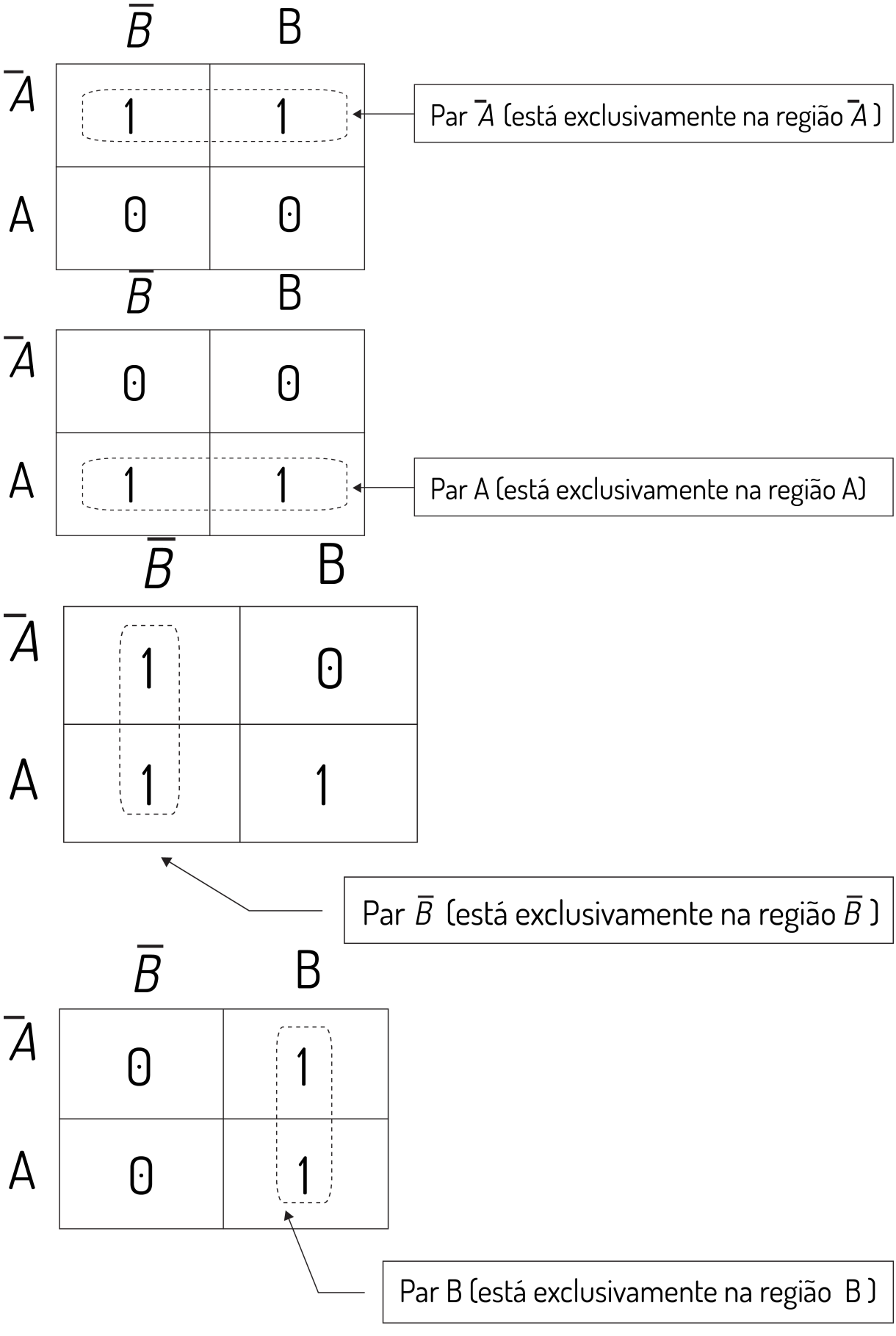
Quando trabalhamos com 2 variáveis, a quadra é o tipo de agrupamento máximo que podemos ter, com todas as posições do mapa sendo 1.



Quadra: S = 1

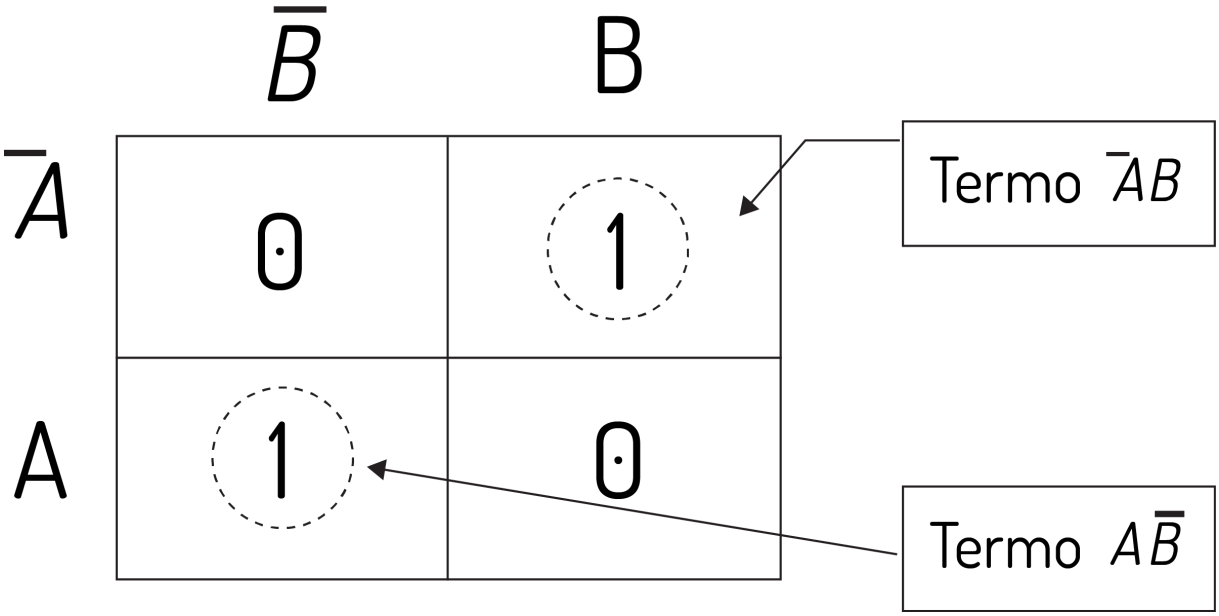
B. Pares

Para que possamos agrupar as variáveis em pares, temos que ter 2 regiões com S = 1 e que tenham um lado em comum.


$\overline{B}$
B
$\overline{A}$
A
1
1
0
0
Par $\overline{A}$ (está exclusivamente na região $\overline{A}$)
$\overline{B}$
B
$\overline{A}$
A
0
0
1
1
Par A (está exclusivamente na região A)
$\overline{B}$
B
$\overline{A}$
A
1
0
1
1
Par $\overline{B}$ (está exclusivamente na região $\overline{B}$)
$\overline{B}$
B
$\overline{A}$
A
0
1
0
1
Par B (está exclusivamente na região B)

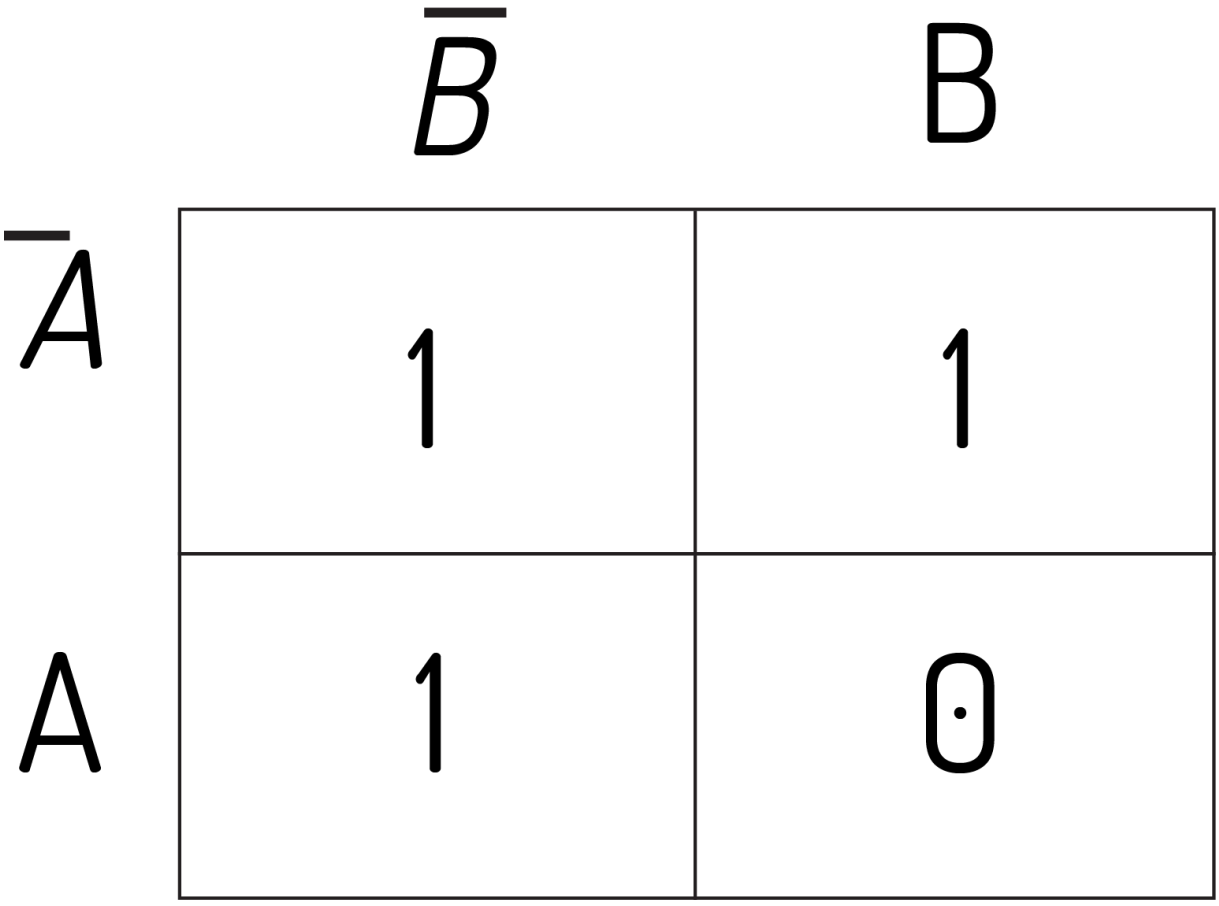
C. Termos isolados

Os termos isolados, na realidade, não são agrupamentos e sim os termos originais da expressão. A seguir, podemos ver alguns deles.

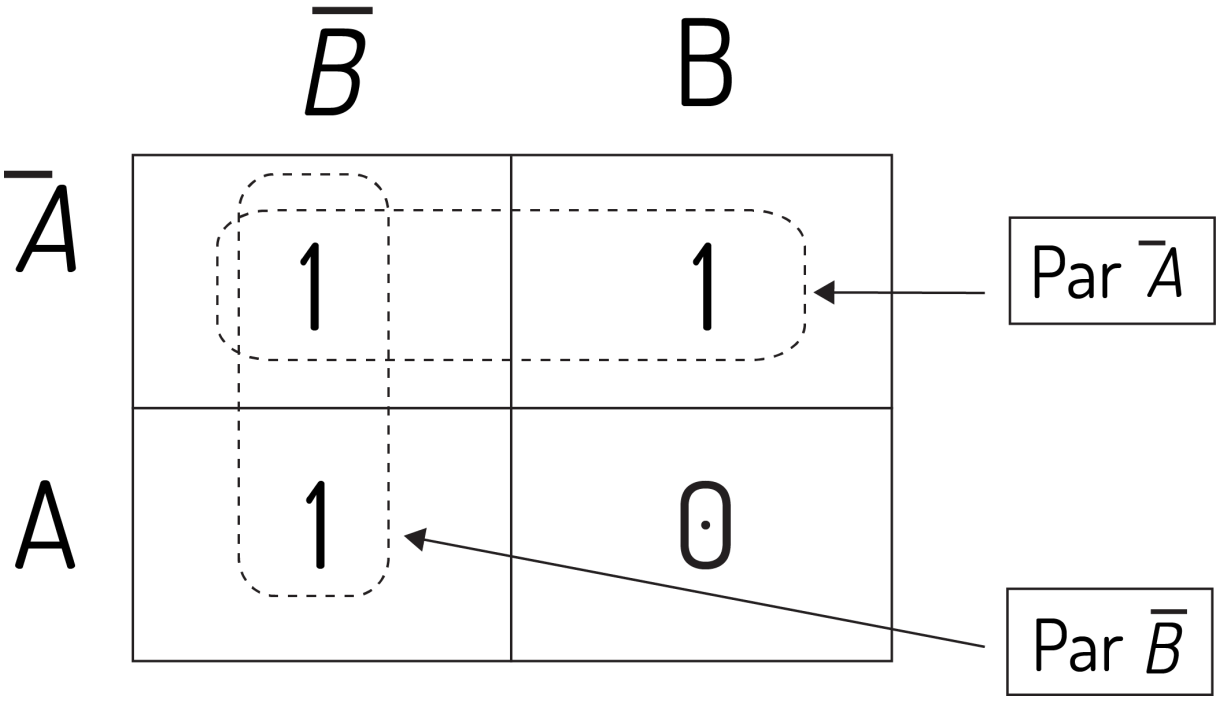


Exemplo: utilizando o mapa de Karnaugh, simplifique a expressão $S = \bar{A} \,.\, B + A \,.\, B + \bar{A} \,.\, B$

Primeiramente, vamos colocar os termos presentes na expressão nas posições adequadas do diagrama:



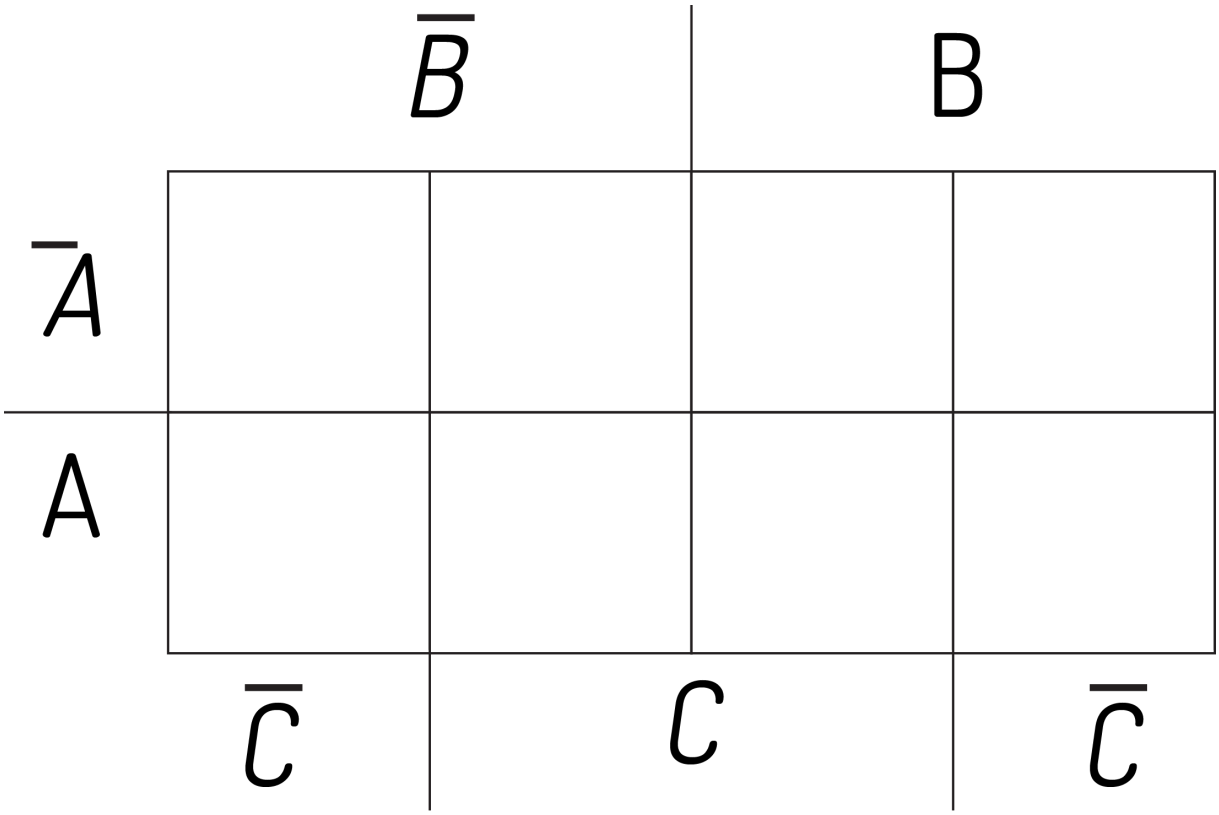
O próximo passo é agrupar os valores 1:



Assim, a expressão original é equivalente a $S = \bar{A} + B$

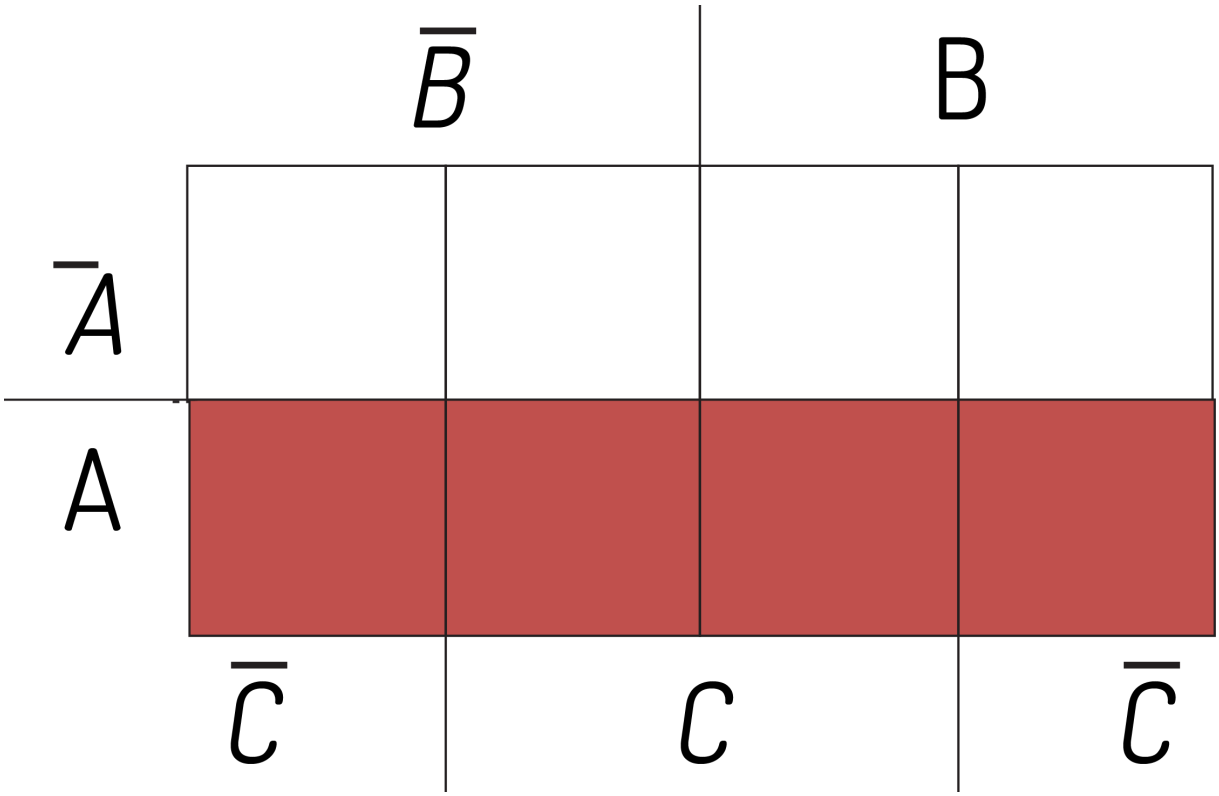
Mapas de Karnaugh para 3 variáveis

A forma do diagrama para três variáveis é:

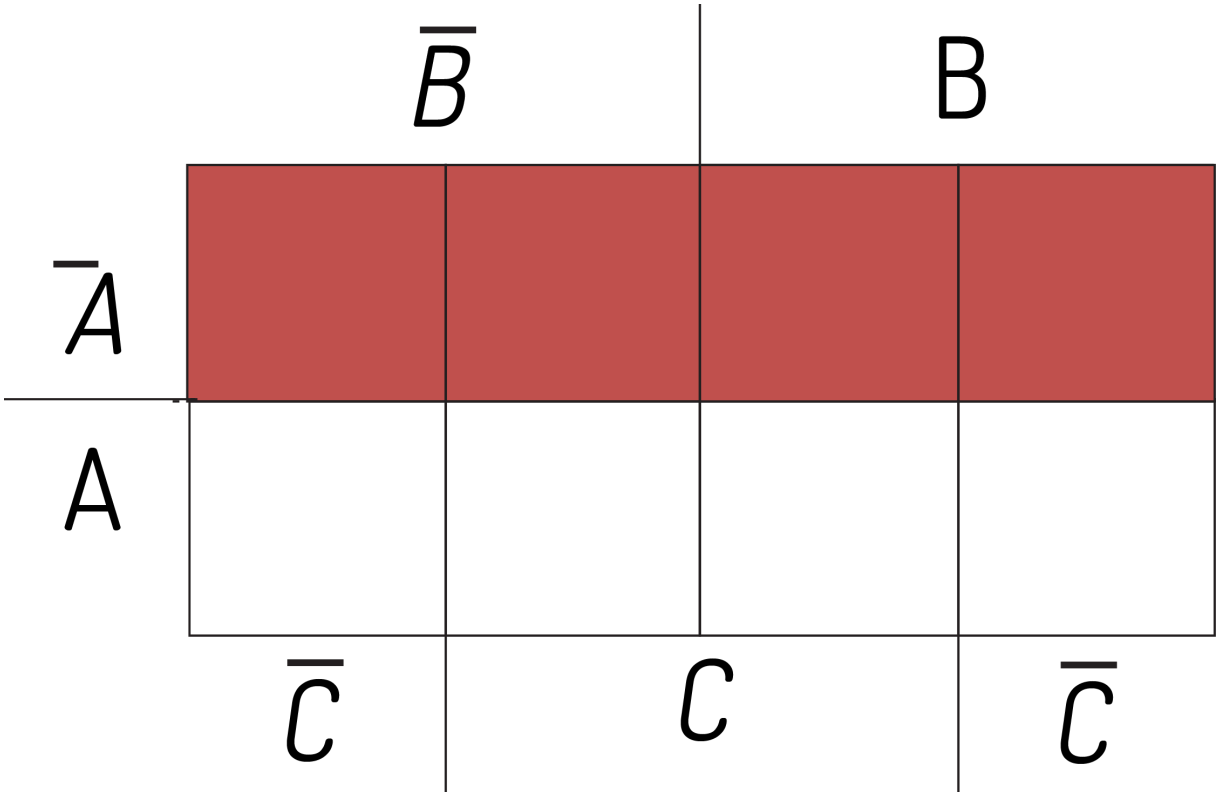


Neste formato, podemos encontrar todas as possibilidades que as variáveis escolhidas podem assumir. São elas:

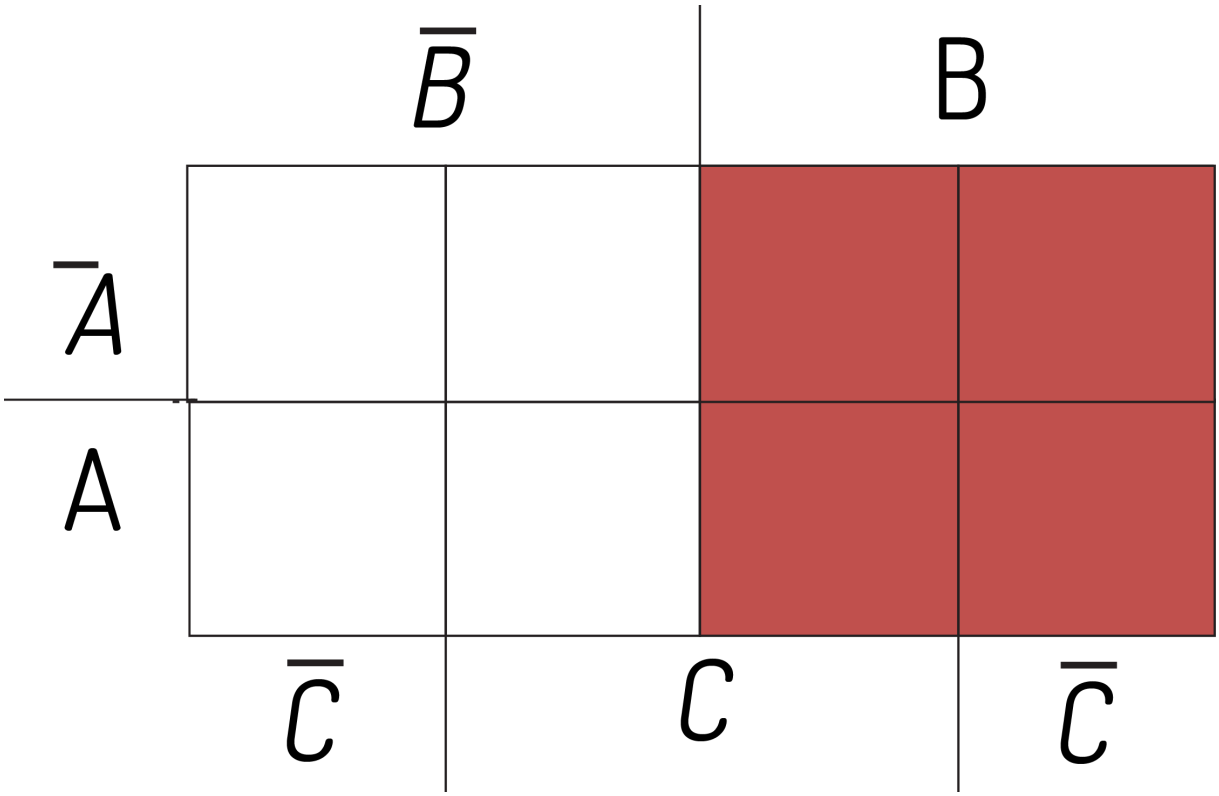
A. $A = 1$



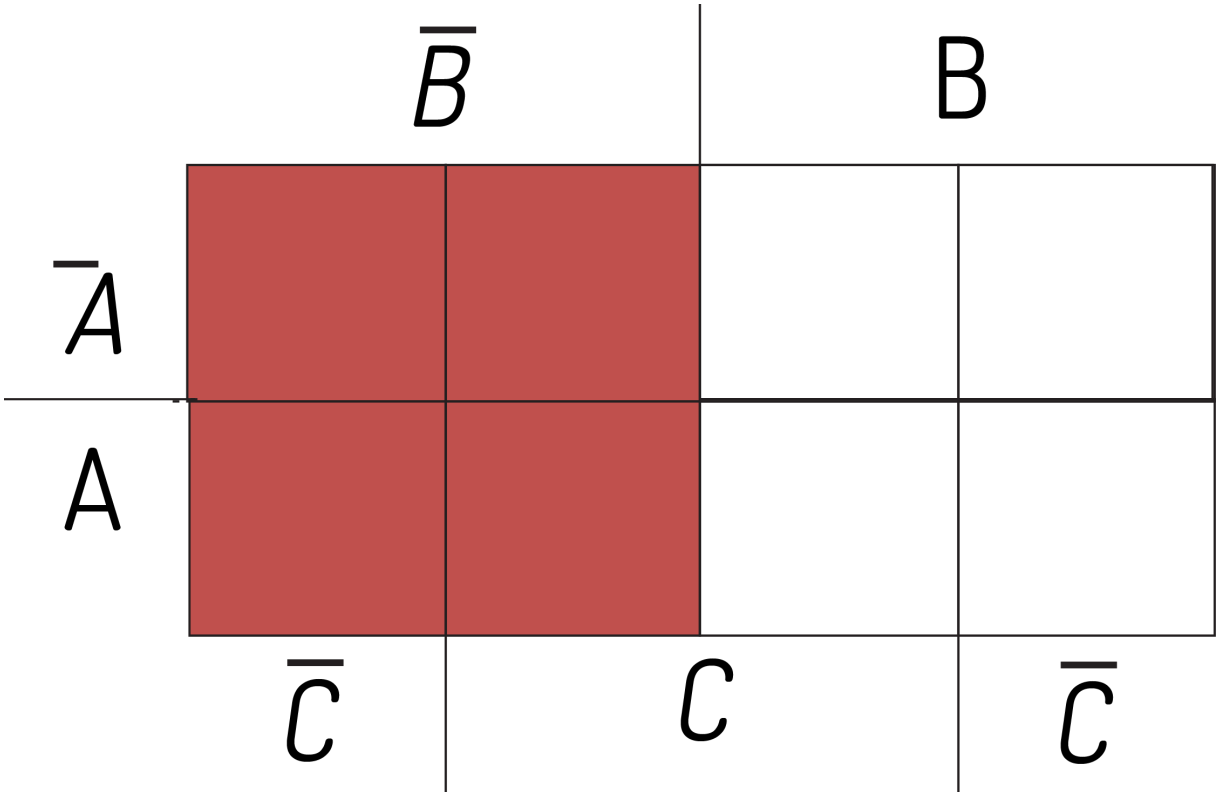
B. $\bar{A} = 1$ ou A = 0



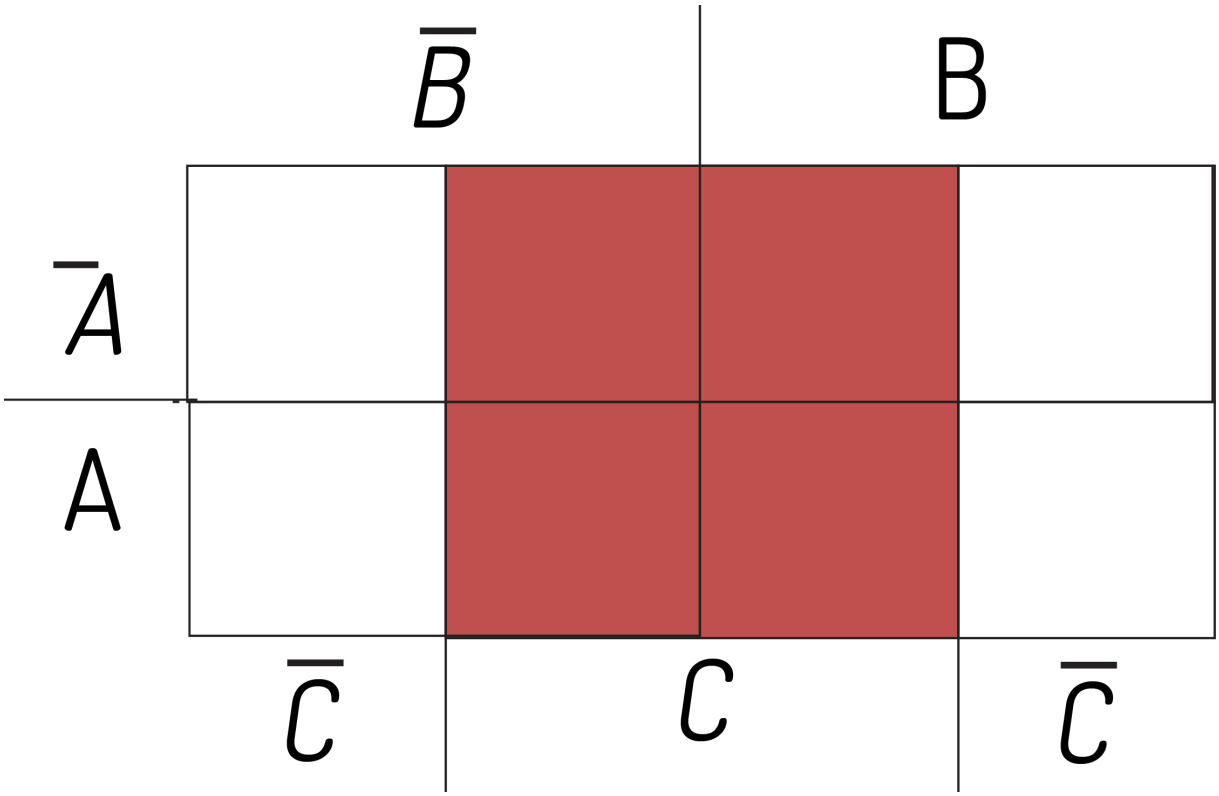
C. B = 1



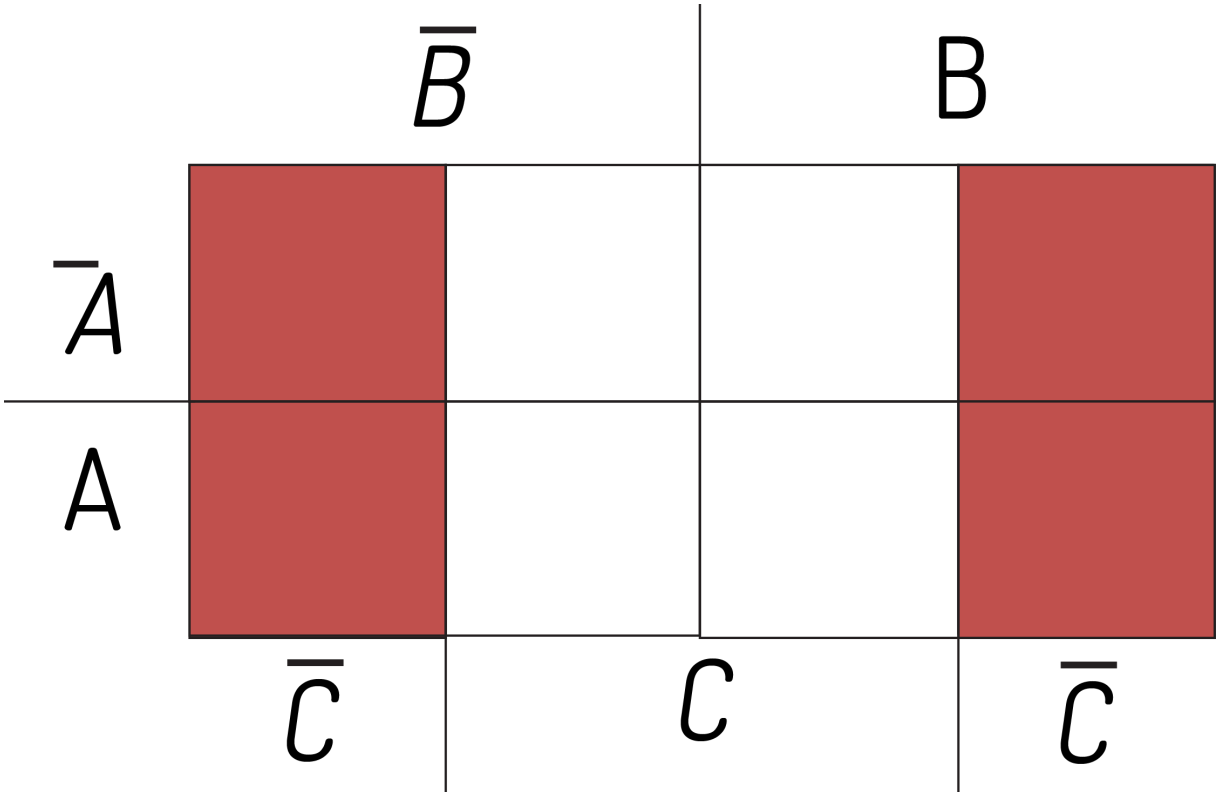
D. $\bar{B} = 1$ ou B = 0



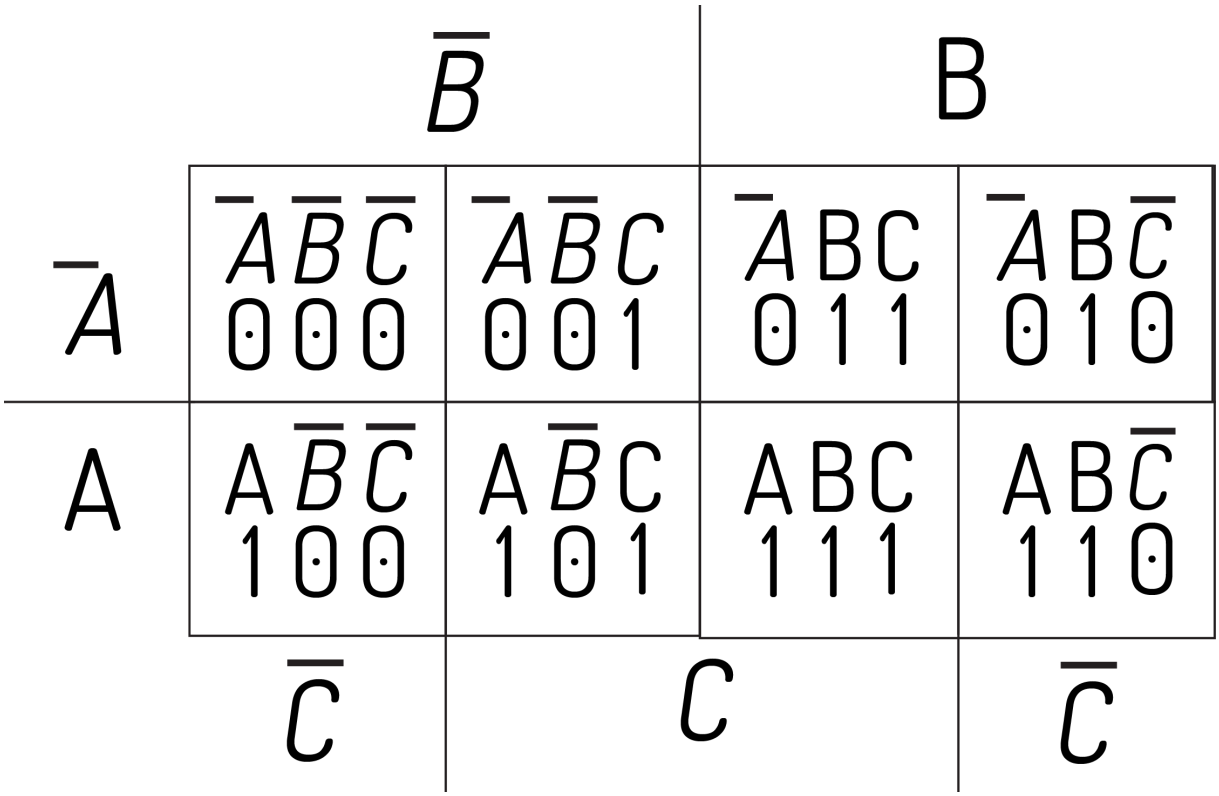
E. C = 1



F. $\bar{C} = 1$ ou C = 0



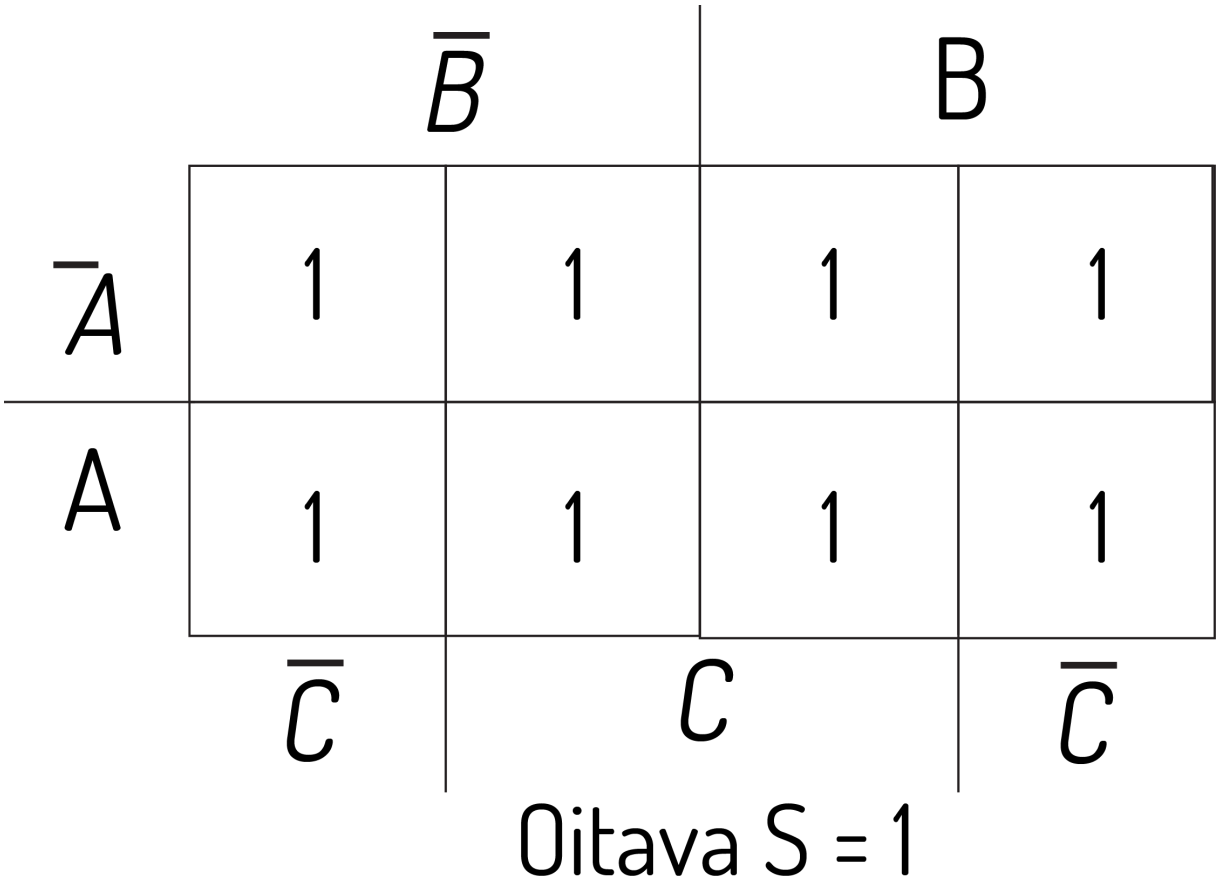
Cada uma das células existentes no diagrama, assim como no caso de duas variáveis, indica uma possibilidade que pode ocorrer entre as variáveis A, B e C, ou seja, uma linha da tabela verdade. Neste caso, temos:



Quando se trabalha com 3 variáveis, os agrupamentos são a oitava, as quadras, os pares e os termos isolados.

A. Oitava

No caso de 3 variáveis, a oitava é o tipo de agrupamento máximo que podemos ter, com todas as posições do mapa sendo 1.



Oitava: S = 1

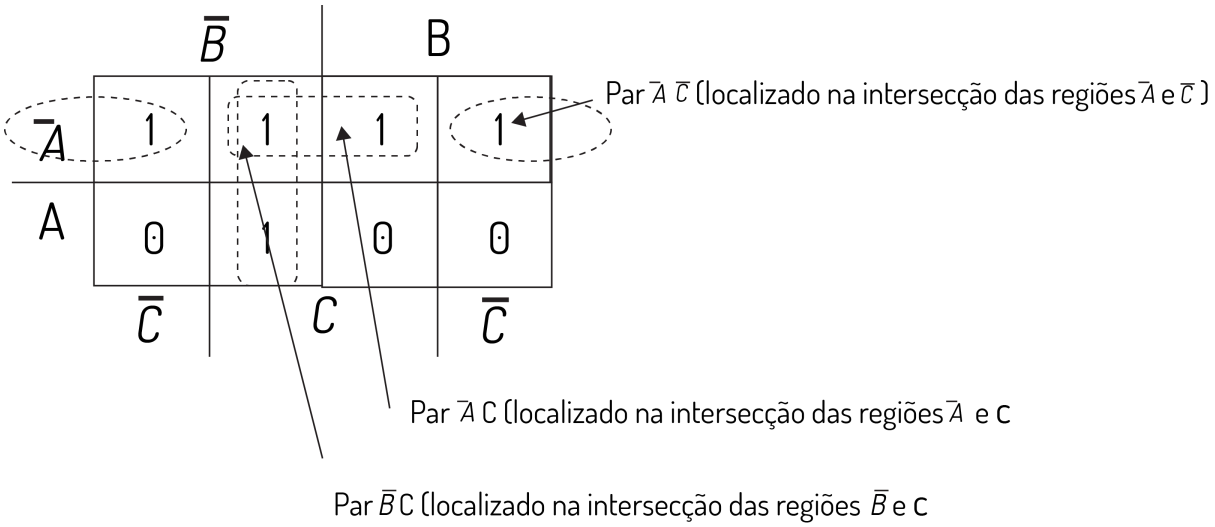
B. Quadras

No caso das quadras, há a necessidade de se agrupar 4 regiões iguais a 1. Elas devem estar em sequência ou ser adjacentes. As quadras presentes no diagrama de Karnaugh de 3 variáveis são as apresentadas nas possibilidades mostradas anteriormente.

C. Pares

Assim como no mapa para duas variáveis, para que possamos agrupar as variáveis em pares, temos que ter 2 regiões com S = 1 e que tenham um lado em comum.

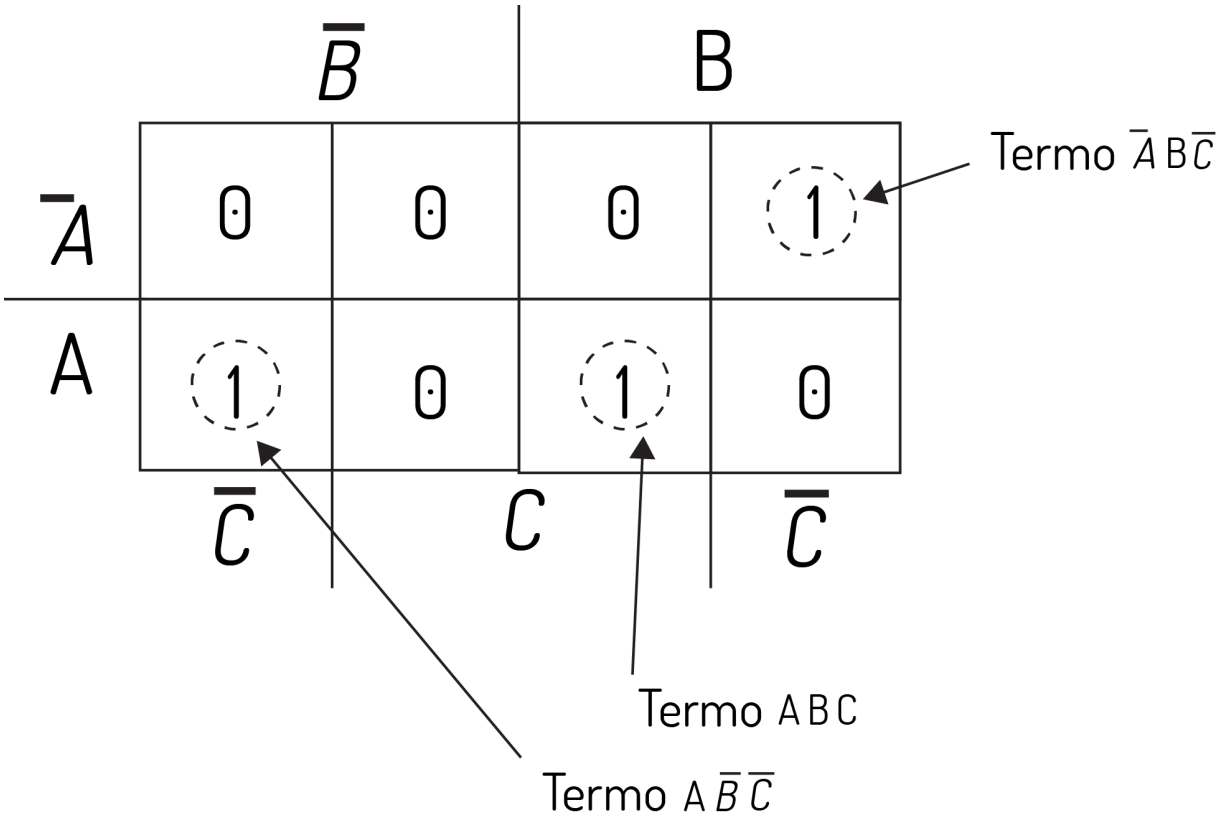
Podemos citar como exemplos:



D. Termos isolados

Neste caso, os termos isolados também são os termos de entrada.

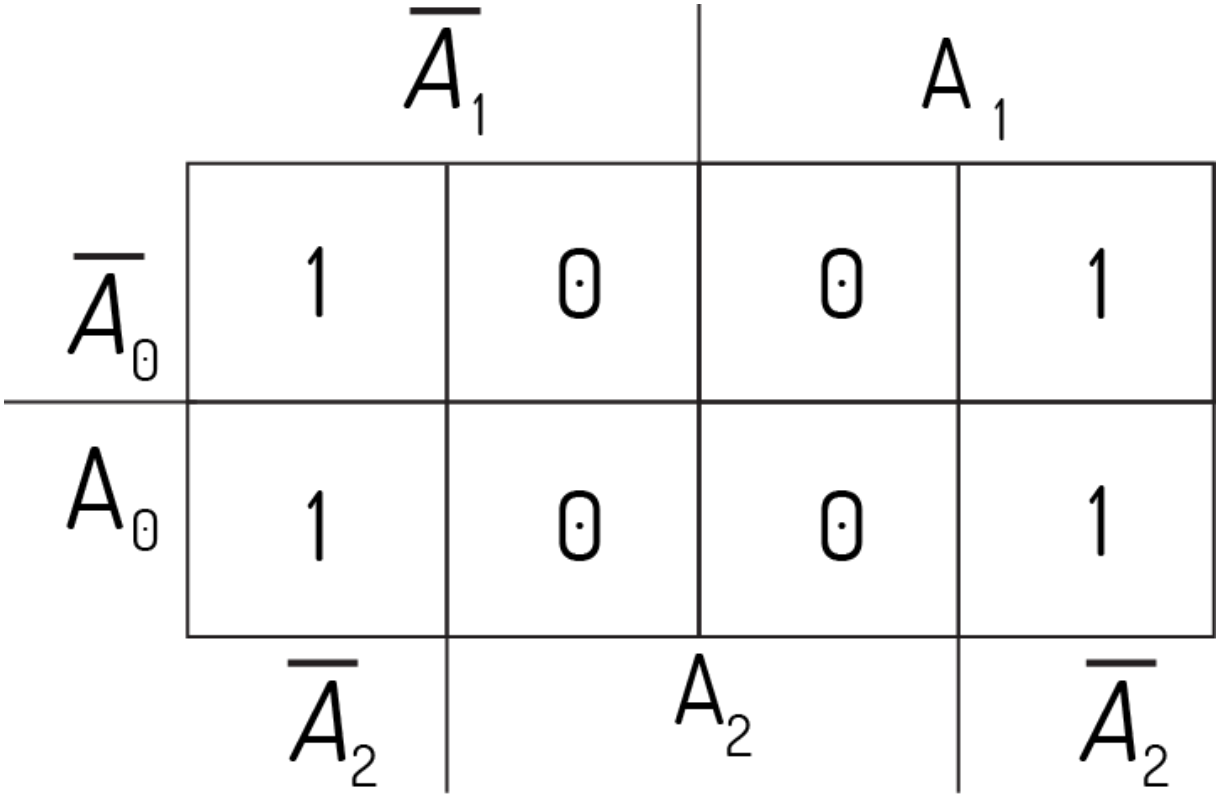
A seguir, apresentamos alguns deles:



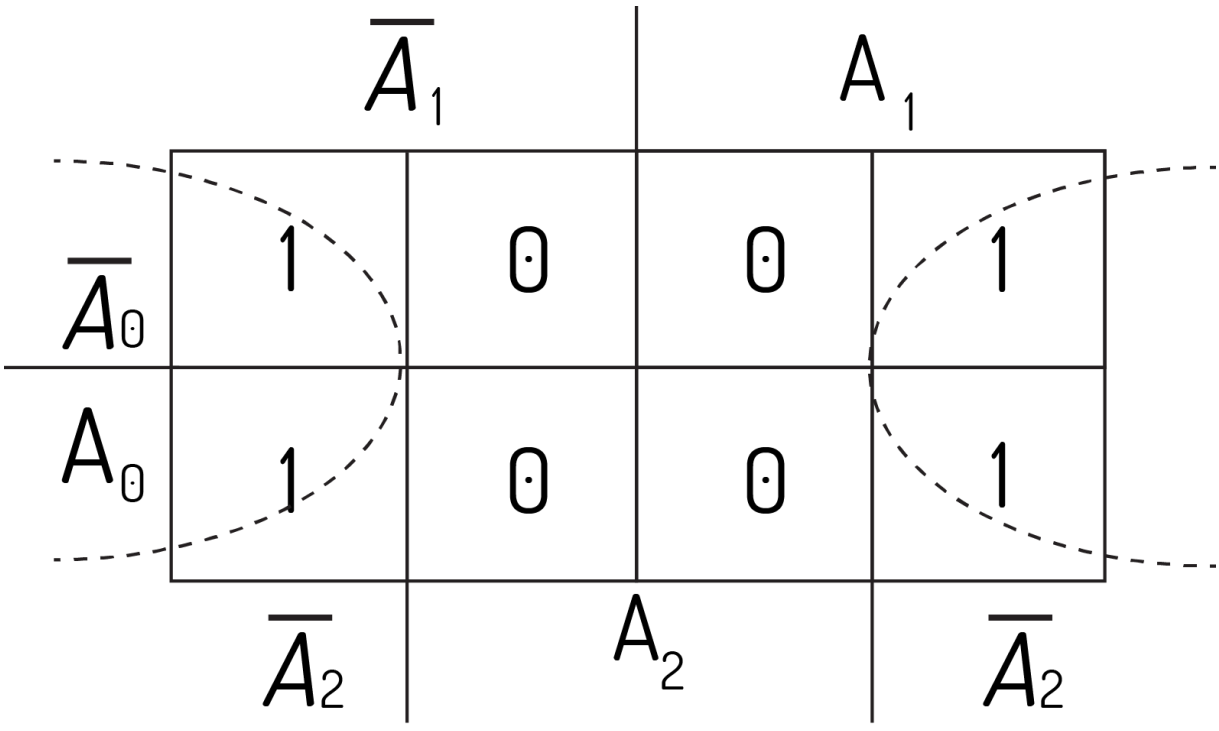
Exemplo: para mostrar a utilização do mapa de Karnaugh para 3 variáveis, vamos retornar ao exemplo utilizado na seção que trata da simplificação por Álgebra de Boole, em que o circuito determina se um número inteiro binário de 3 bits é menor ou igual a 3. A expressão booleana obtida foi:

$$S = \overline{A_2}\,\overline{A_1}\,\overline{A_0} + \overline{A_2}\,\overline{A_1}\,A_0 + \overline{A_2}\,A_1\,\overline{A_0} + \overline{A_2}\,A_1\,A_0$$

Primeiramente, vamos colocar os termos presentes na expressão nas posições adequadas do diagrama:



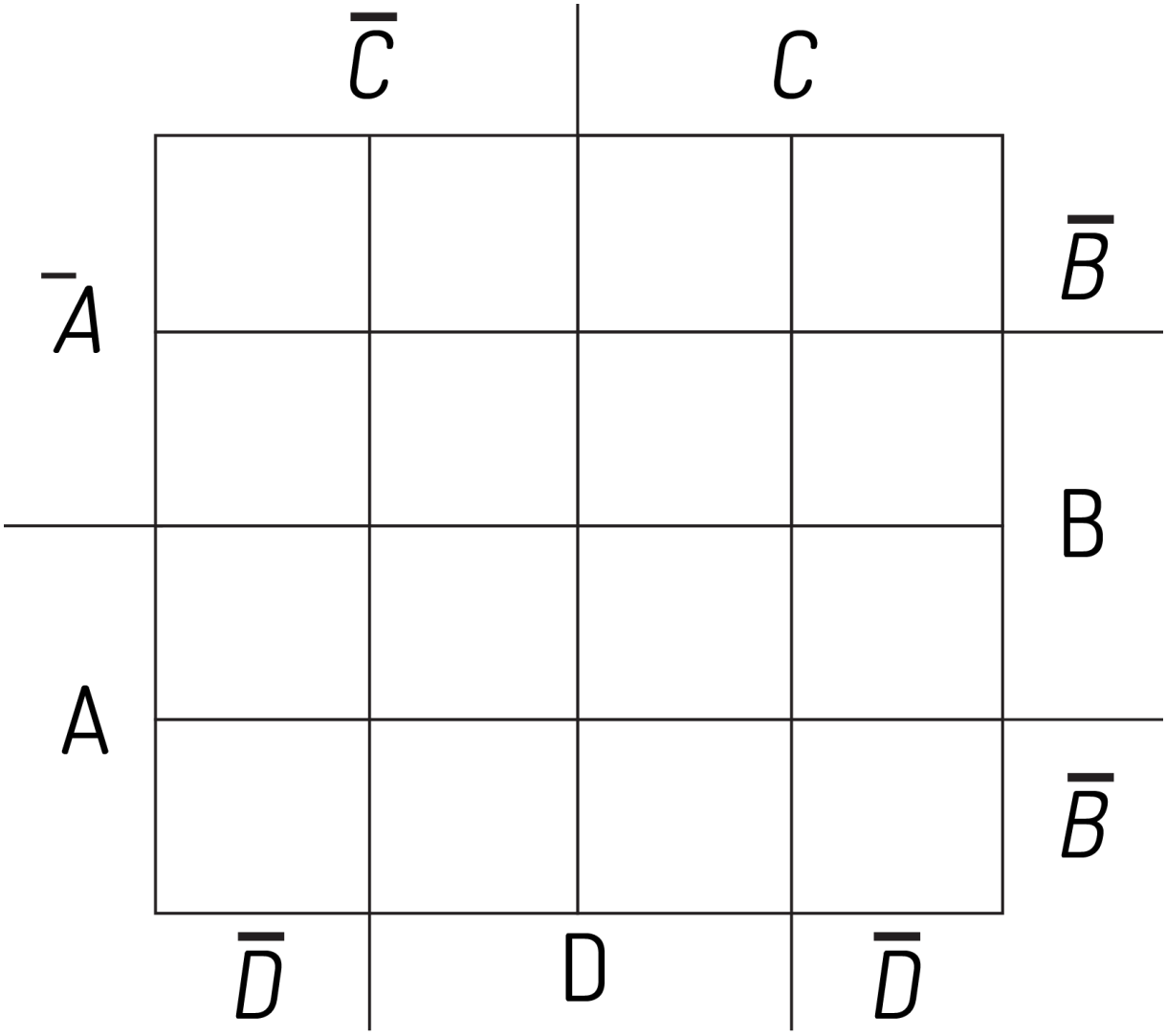
O próximo passo é agrupar os valores 1:



Assim, vemos que a expressão original é equivalente a $S = \overline{A_2}$, o mesmo valor encontrado anteriormente.

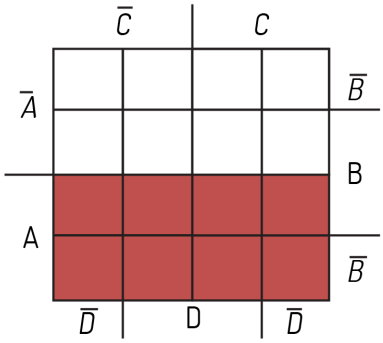
Mapas de Karnaugh para 4 variáveis

A forma do diagrama para quatro variáveis é:

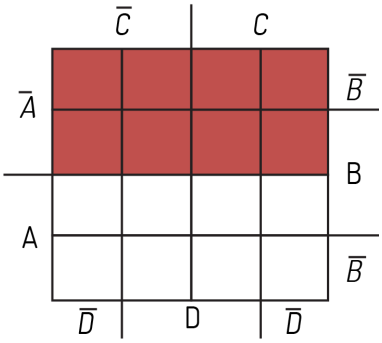


Assim como nos casos anteriores, nesta forma, podem ser colocadas todas as possibilidades assumidas entre as variáveis A, B, C e D.

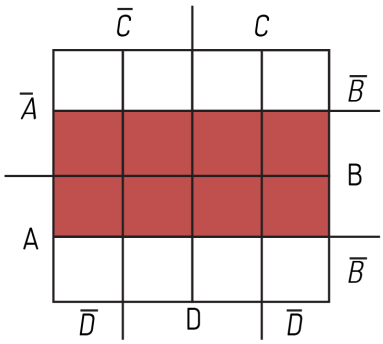
a) A = 1



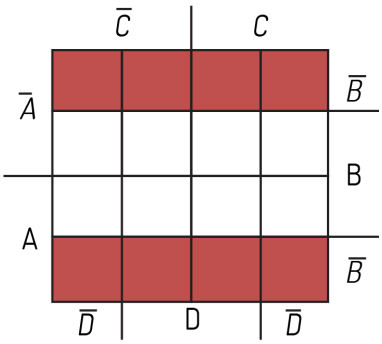
a) $\overline{A}$ = 1 ou A = 0



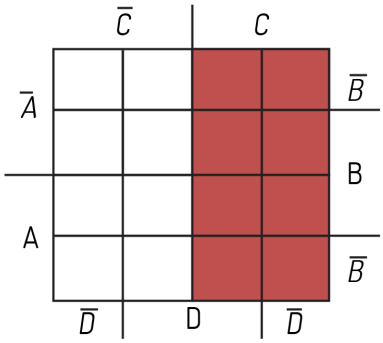
c) B = 1



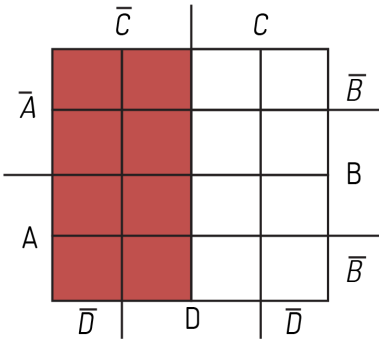
d) $\overline{B}$ = 1 ou B = 0



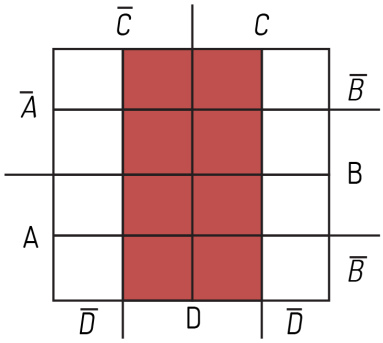
e) C = 1



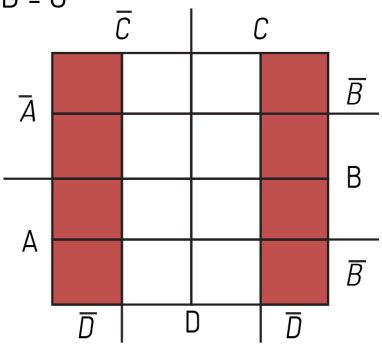
f) $\overline{C}$ = 1 ou C = 0



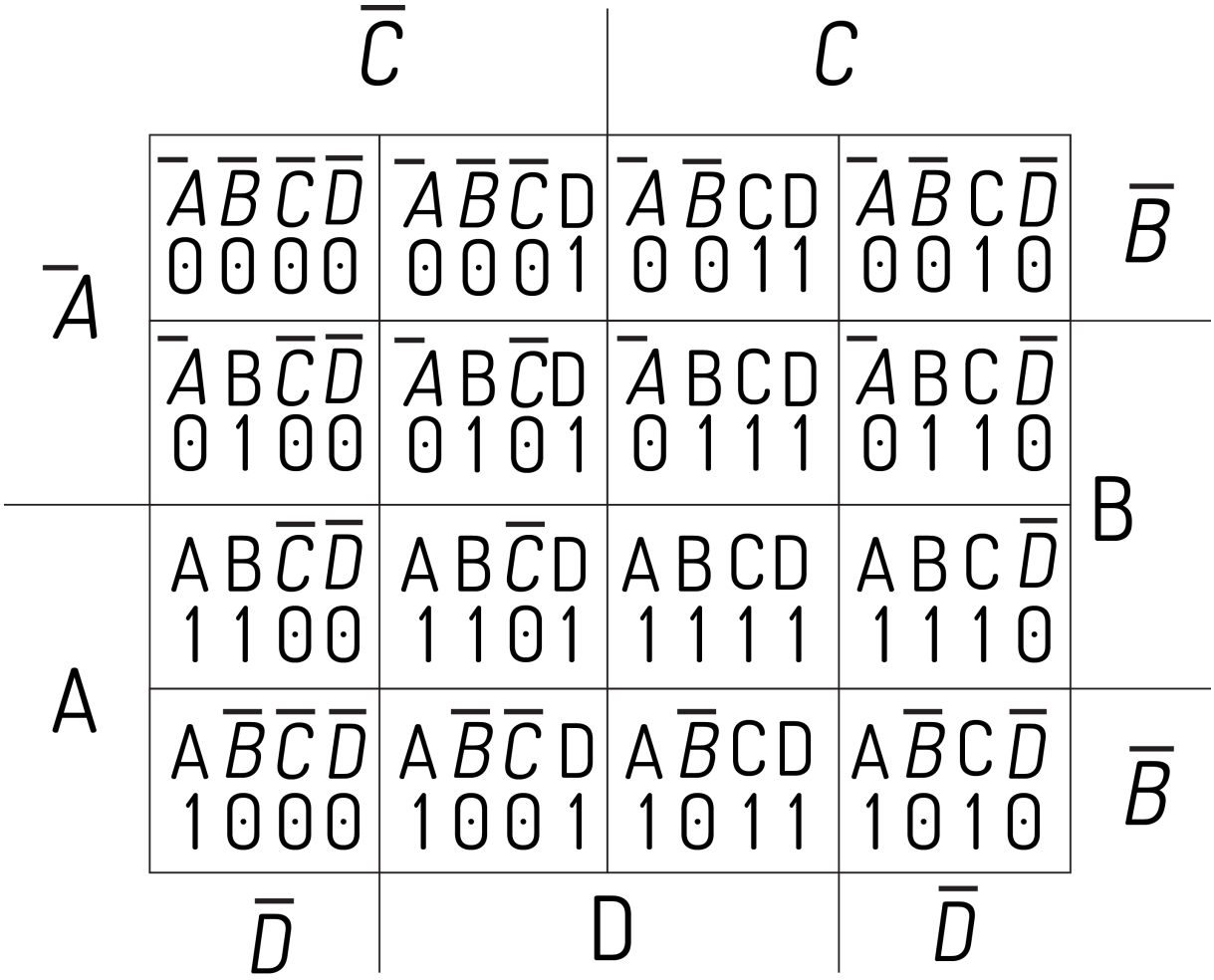
g) D = 1



h) $\overline{D}$ = 1 ou D = 0



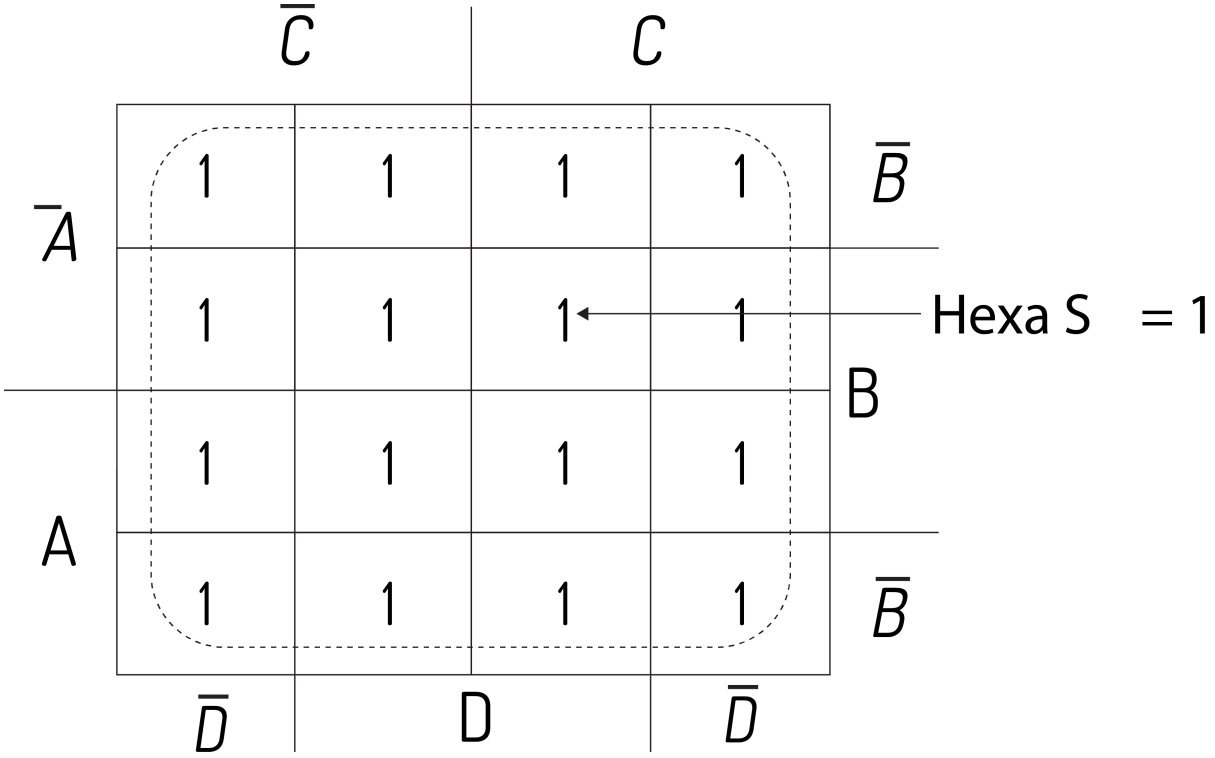
Cada uma das células existentes no diagrama, assim como nos casos anteriores, indica uma possibilidade que pode ocorrer entre as variáveis A, B, C e D, ou seja, uma linha da tabela verdade. Neste caso, temos:



Quando se trabalha com 4 variáveis, os agrupamentos são a hexa, as oitavas, as quadras, os pares e os termos isolados.

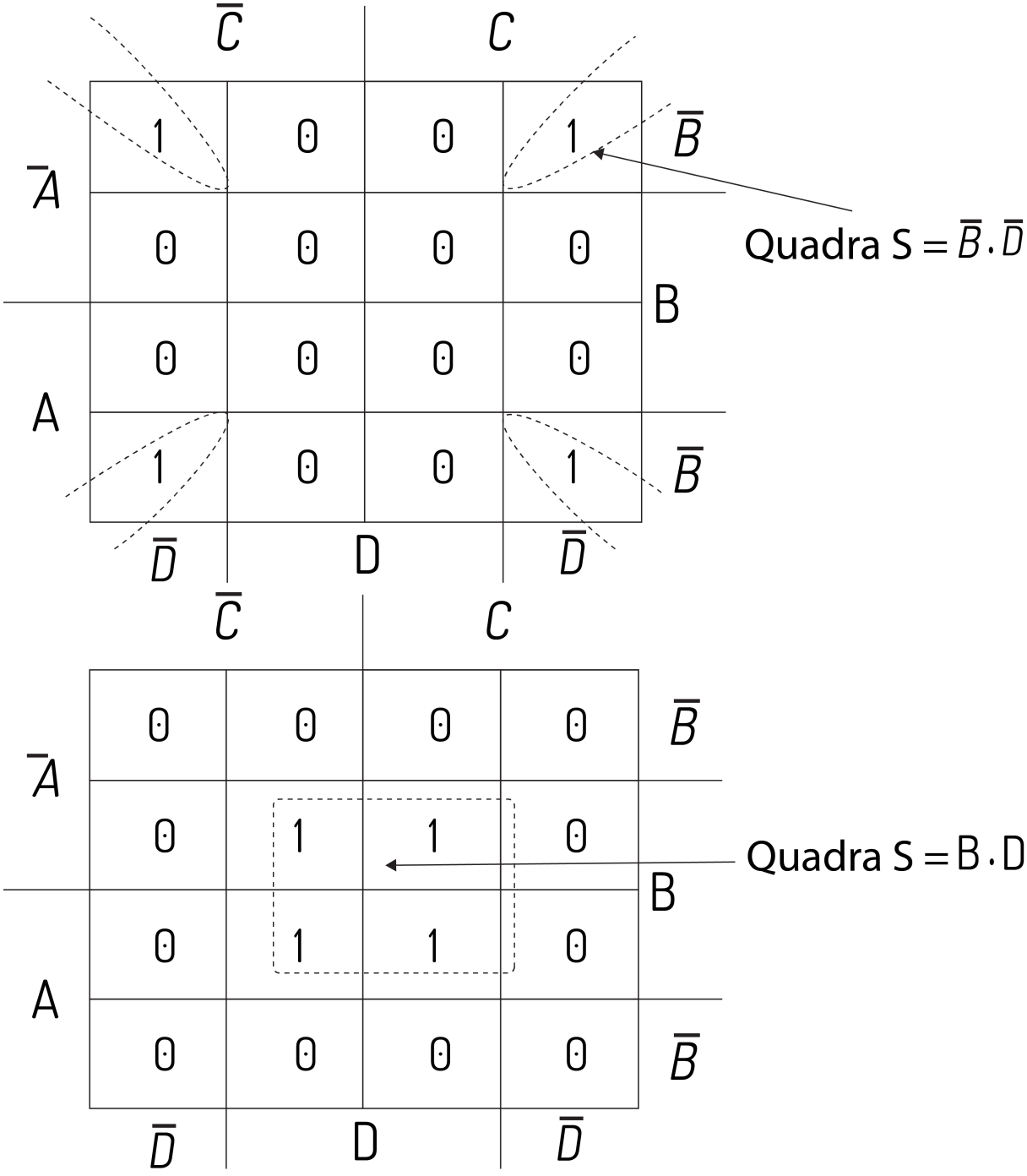
A. Hexa

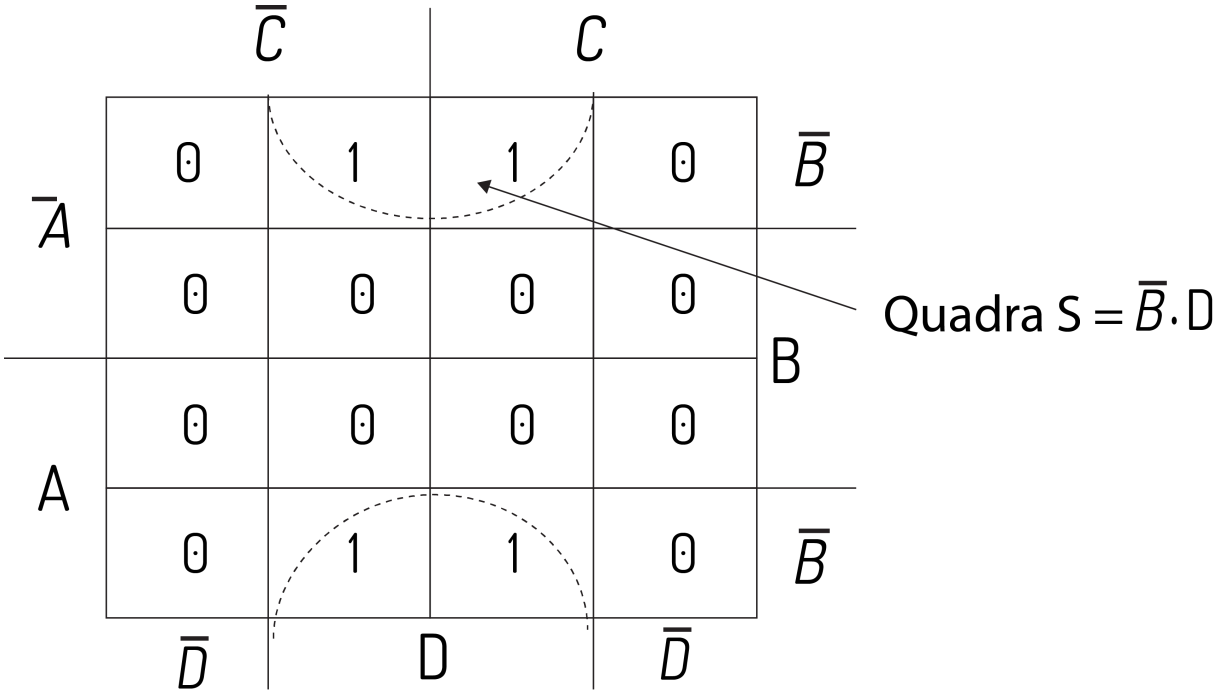
Agora, esse é o nosso agrupamento máximo.



B. Oitavas

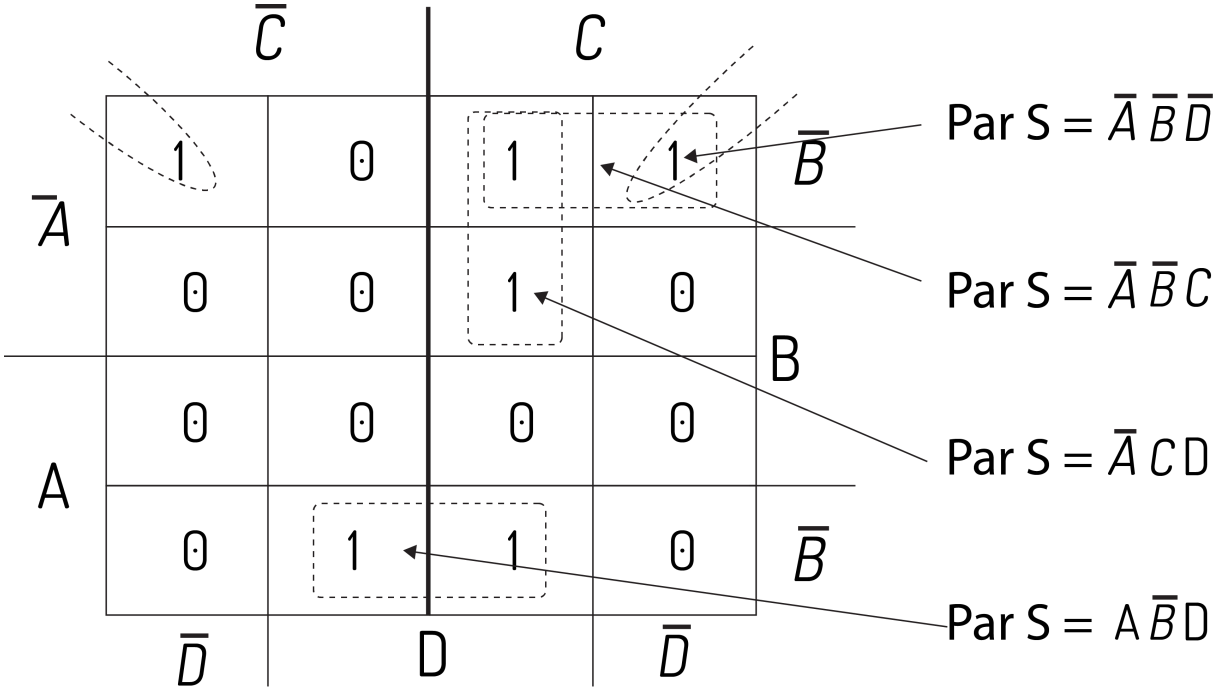
No caso das oitavas, precisamos agrupar 8 regiões iguais a 1, em sequência ou adjacentes. As oitavas para o mapa de Karnaugh de 4 variáveis são as apresentadas anteriormente, nas possibilidades.


$\overline{C}$
C
$\overline{A}$
A
B
$\overline{B}$
$\overline{D}$
D
1 0 0 1
0 0 0 0
0 0 0 0
1 0 0 1
Quadra S = $\overline{B} \cdot \overline{D}$
0 0 0 0
0 1 1 0
0 1 1 0
0 0 0 0
Quadra S = B · D



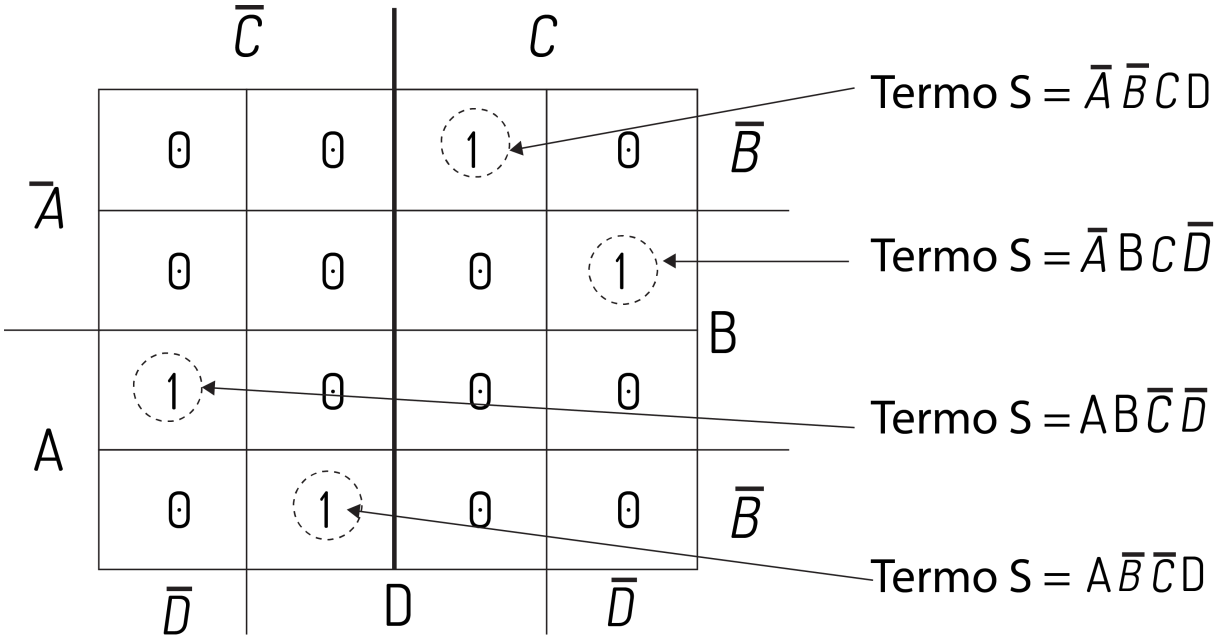
C. Quadras

Para as quadras, há a necessidade de se agrupar 4 regiões iguais a 1. Elas devem estar em sequência ou ser adjacentes. Podemos citar como exemplos:



D. Pares

Nos pares, temos que ter 2 regiões com S = 1 e que tenham um lado em comum.

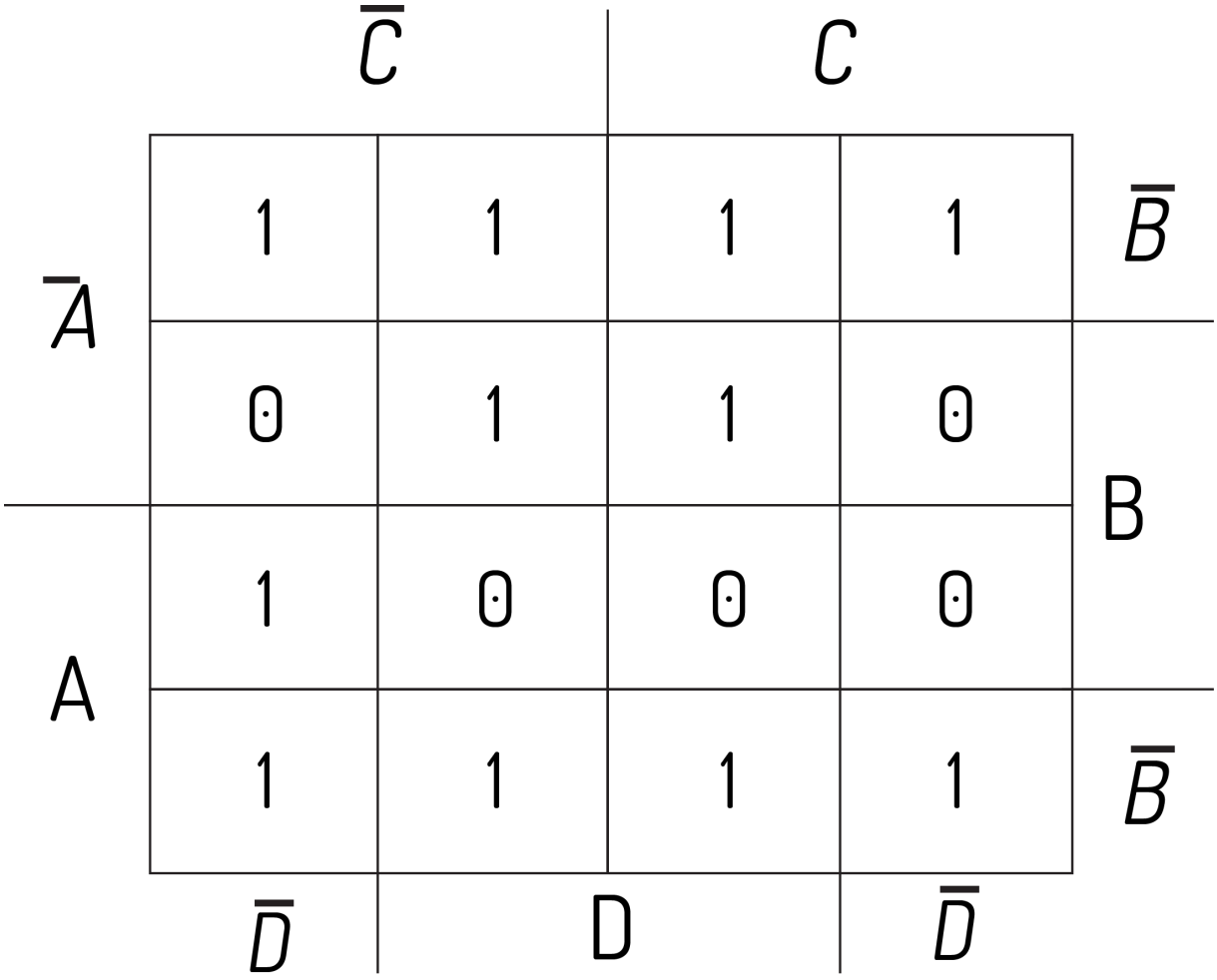


E. Termos isolados

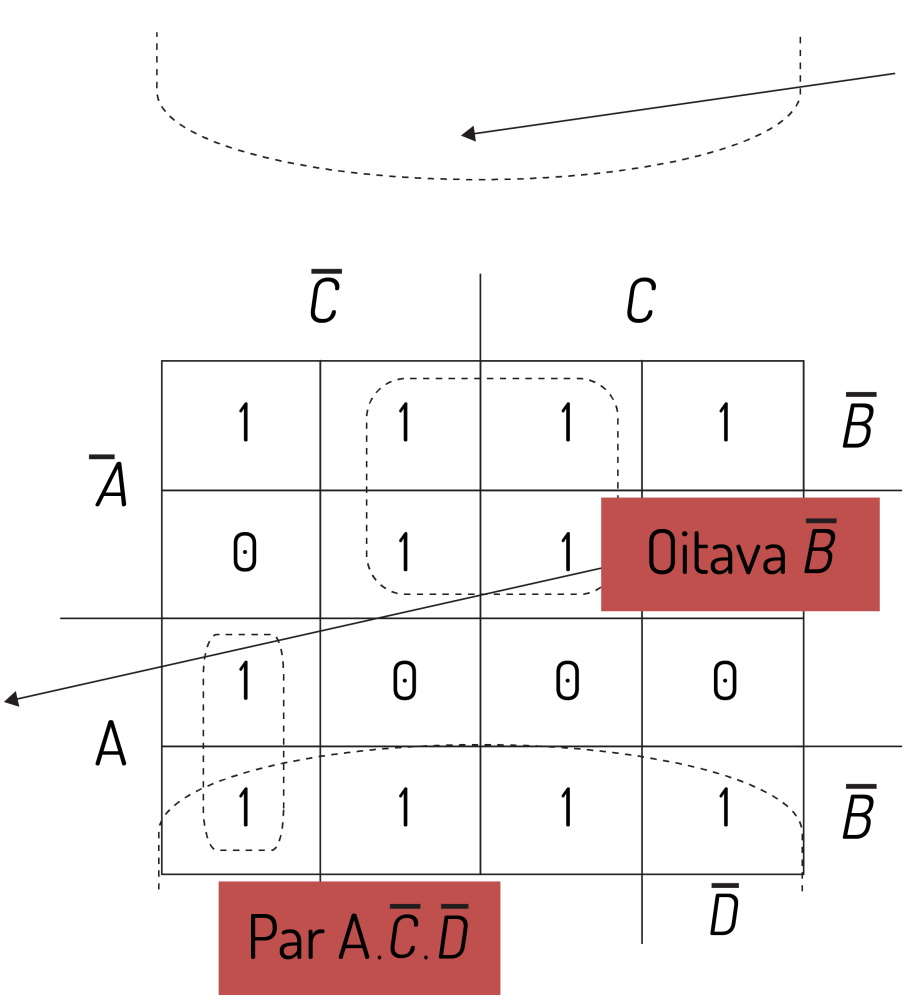
Como nos casos anteriores, os termos isolados também são os próprios termos de entrada.

Exemplo: utilizando o mapa de Karnaugh, simplifique a expressão $S = \bar{A} . \overline{B} . \overline{C} . D + \bar{A} . B . \bar{C} . D + \bar{A} . B . C . D + \bar{A} . B . C . D + \bar{A} . B . \bar{C} . D + \bar{A} . B . C . D + A . B . \bar{C} . D + A . B . \bar{C} . D + A . B . \bar{C} . D + A . B . C$.

Colocando os termos presentes na expressão nas posições adequadas do diagrama:



Agrupando os valores 1:



Assim, a expressão original é equivalente a $S = B + \bar{A} \cdot D + A \cdot \bar{C} \cdot D$.

¶ Fique por dentro

Agrupamento de zeros

Idoeta (2012) lembra que podemos, alternativamente, agrupar as células que valem 0 para obtermos a expressão simplificada em diagramas de Vetch-Karnaugh, porém, com esta prática, obtemos o complemento da função, ou seja, a saída $\bar{S}$.

Introdução aos Circuitos Combinatórios

Agora que já trabalhamos com as portas lógicas, entendendo suas funções e como funciona o agrupamento entre elas, os chamados circuitos lógicos ou digitais, além dos métodos utilizados para simplificar nossas expressões booleanas, vamos dar início ao nosso estudo dos circuitos combinatórios ou combinacionais.

Monteiro (2012) traz a seguinte definição: um circuito combinatório é definido como um conjunto de portas cuja saída em qualquer instante de tempo é função somente das entradas. Em contrapartida, um circuito sequencial, além de possuir portas, contém elementos de armazenamento, denominados *flip-flops*.

Como já falamos no início de nossos estudos sobre álgebra booleana, devemos seguir alguns passos para que um projeto lógico seja implementado, segundo Güntzel (2001):

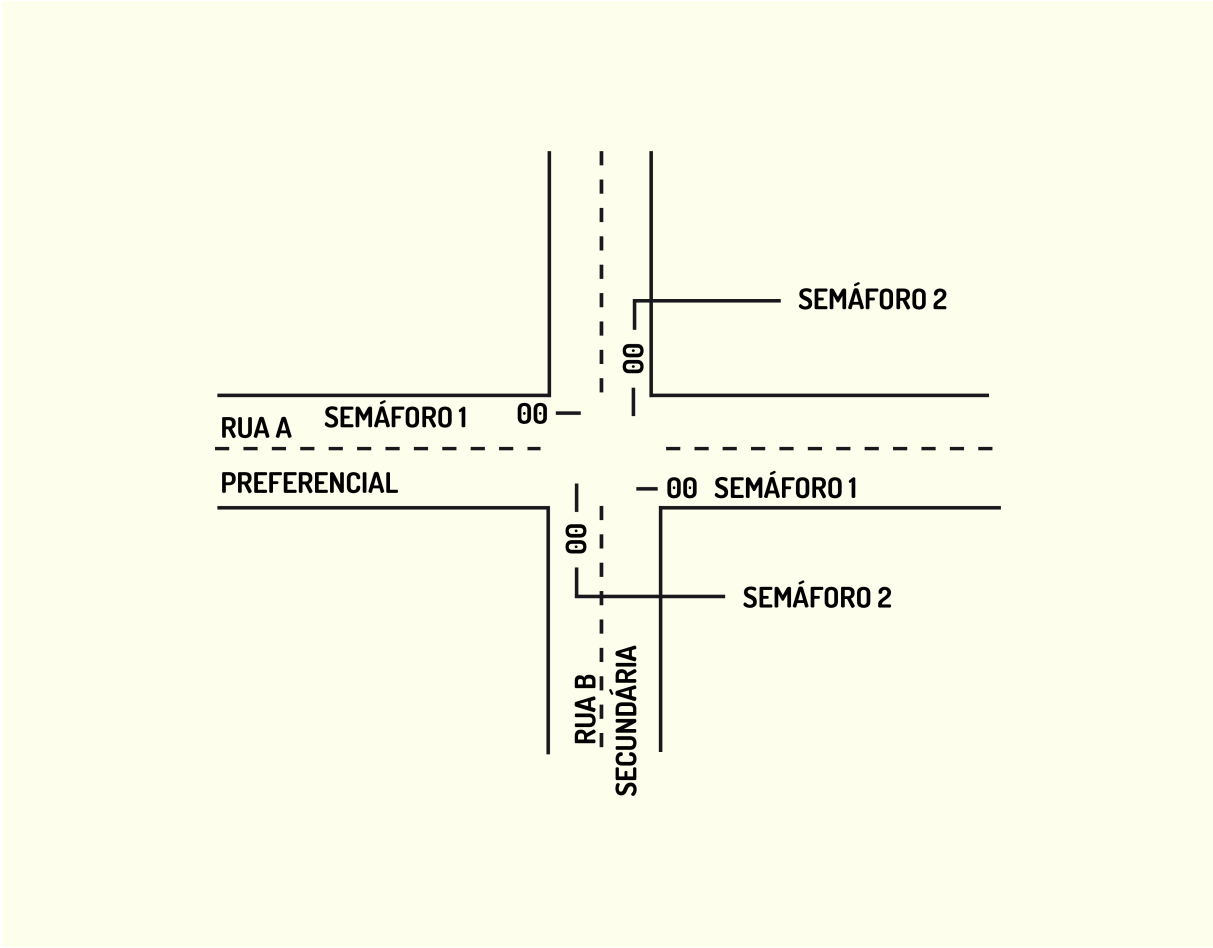
1. Escolher um símbolo para cada variável de entrada e para cada variável de saída;
2. A partir da especificação do problema, determinar a tabela verdade (caso ela já não faça parte da especificação do problema);
3. Obter as equações simplificadas;
4. Mapear o circuito para a biblioteca de portas disponível (se for o caso);
5. Desenhar o circuito final.

A partir desse momento, iremos trabalhar com alguns exemplos de projetos de circuitos combinatórios simples.

Projeto 1

Idoeta (2012) propõe um projeto simples de um sistema automático para semáforos, que depende da existência ou não de trânsito em uma via.

A Figura 2.1 representa o cruzamento das ruas A e B.



2FIGURA 1.4 - Cruzamento das ruas A e B FONTE: Idoeta (2012).

Nesse cruzamento, queremos instalar um sistema automático para os semáforos, com as seguintes características:

a. Quando houver carros transitando somente na Rua B, o semáforo 2 deverá permanecer verde para que estas viaturas possam trafegar livremente;
b. Quando houver carros transitando somente na Rua A, o semáforo 1 deverá permanecer verde pelo mesmo motivo;
c. Quando houver carros transitando nas duas ruas, deveremos abrir o semáforo para a Rua A, pois é preferencial;
d. Quando não houver veículos trafegando em nenhuma das ruas, o semáforo 2 deverá estar aberto.

SITUAÇÃO	*A*	*B*	V_1	V_{M1}	V_2	V_{M2}	*JUSTIFICATIVA*
0	0	0	0	1	1	0	Como não há carros em nenhuma das ruas, tanto faz qual semáforo ficará aceso no verde. Foi convencionado, para isso, o semáforo 2.
1	0	1	0	1	1	0	Agora temos trânsito na rua B, portanto o verde do semáforo 2 deverá ficar aceso.

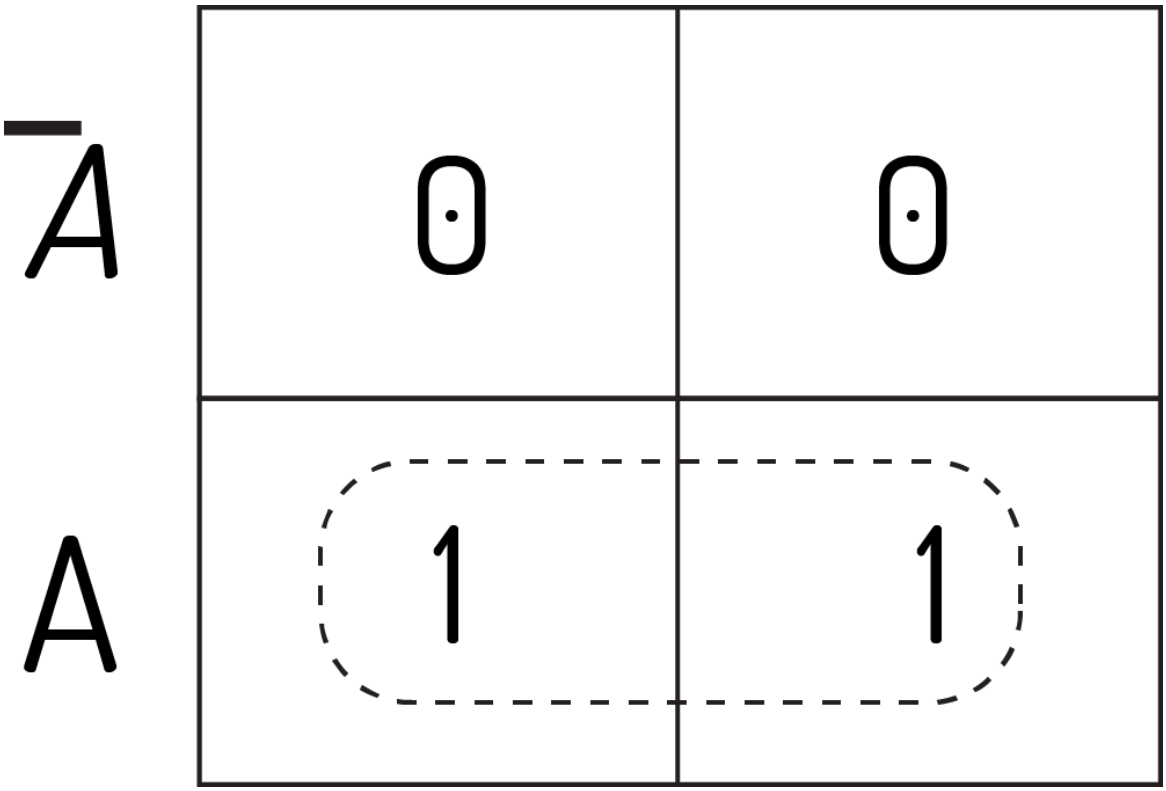
2	1	0	1	0	0	1	Existem carros na rua A, agora o verde do semáforo 1 tem que estar aceso
3	1	1	1	0	0	1	Temos, nesse momento, carros nas duas ruas. Mas a rua A é preferencial, portanto o verde que deverá ficar aceso é o do semáforo 1

2QUADRO 1.1 - Tabela Verdade FONTE: A autora.

O primeiro passo para resolver esse problema é construir a tabela verdade, de acordo com os requisitos fornecidos.

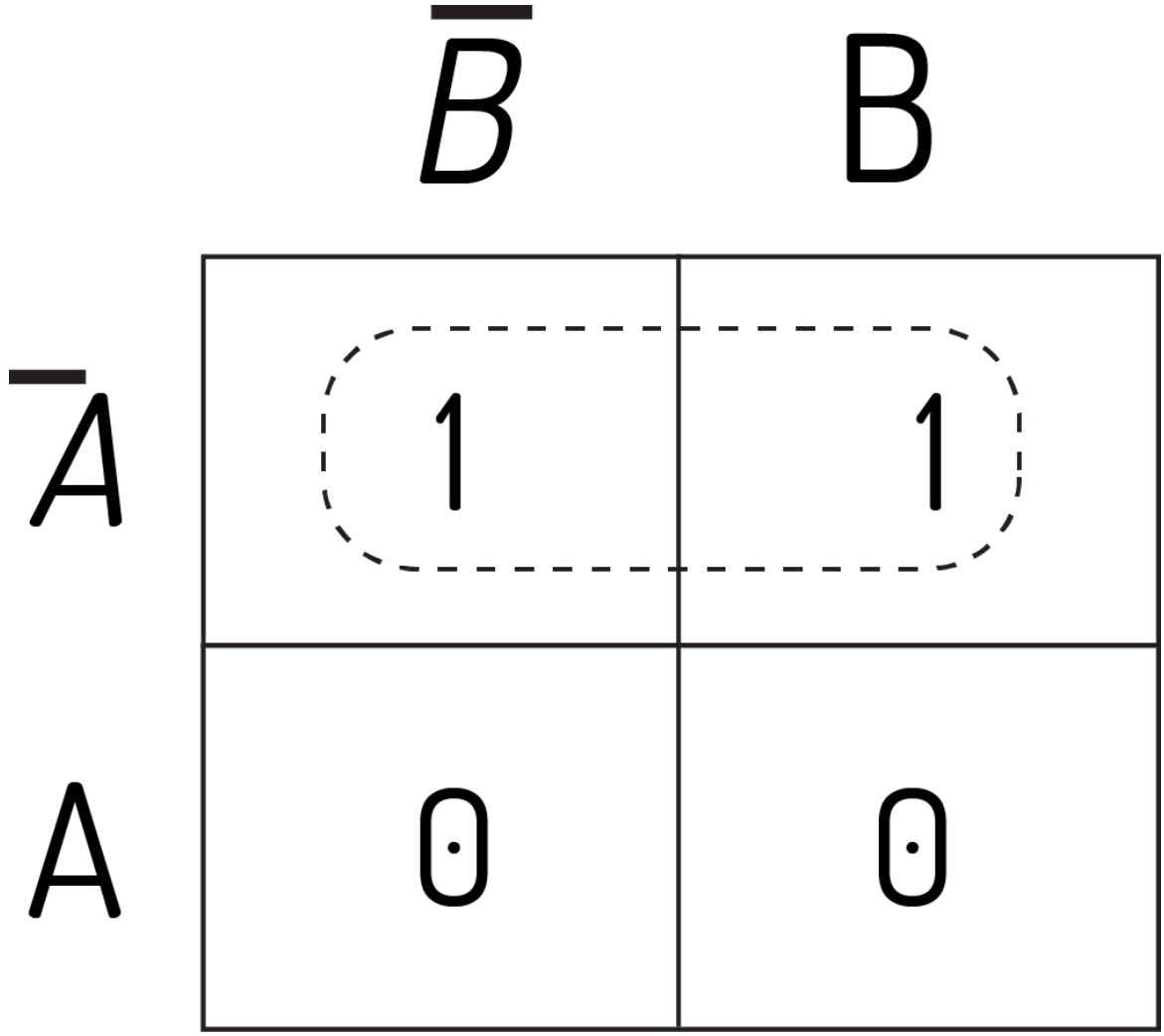
1. Iremos utilizar as variáveis A e B como sendo de entrada e os símbolos V_1, V_2, V_{m1} e V_{m2} para as saídas.
2. Precisamos agora determinar a tabela verdade. Para isso, vamos trabalhar com as seguintes convenções:
   a. Existência de carro na rua A: A = 1
   b. Não existência de carro na rua A: A = 0 ou $\bar{A}$ = 1
   c. Existência de carro na rua B: B = 1
   d. Não existência de carro na rua B: B = 0 ou B = 1
   e. Verde do sinal 1 aceso: $V_1 = 1$
   f. Verde do sinal 2 aceso: $V_2 = 1$
   g. Quando $V_1 = 1$:
      - Vermelho do semáforo 1 apagado ($V_{m1} = 0$);
      - Verde do semáforo 2 apagado ($V_2 = 0$);
      - Vermelho do semáforo 2 aceso ($V_{m2} = 1$).
   h. Quando $V_2 = 1$:
      - Vermelho do semáforo 2 apagado ($V_{m2} = 0$);
      - Verde do semáforo 1 apagado ($V_1 = 0$);
      - Vermelho do semáforo 1 aceso ($V_{m1} = 1$).
   i. Construindo a tabela verdade (Tabela 2.1), de acordo com as convenções estabelecidas.
3. O próximo passo é obter as equações booleanas e simplificá-las. Para isso, vamos usar os mapas de Karnaugh para cada um dos sinais de saída.

V_1 (sinal verde do semáforo 1)



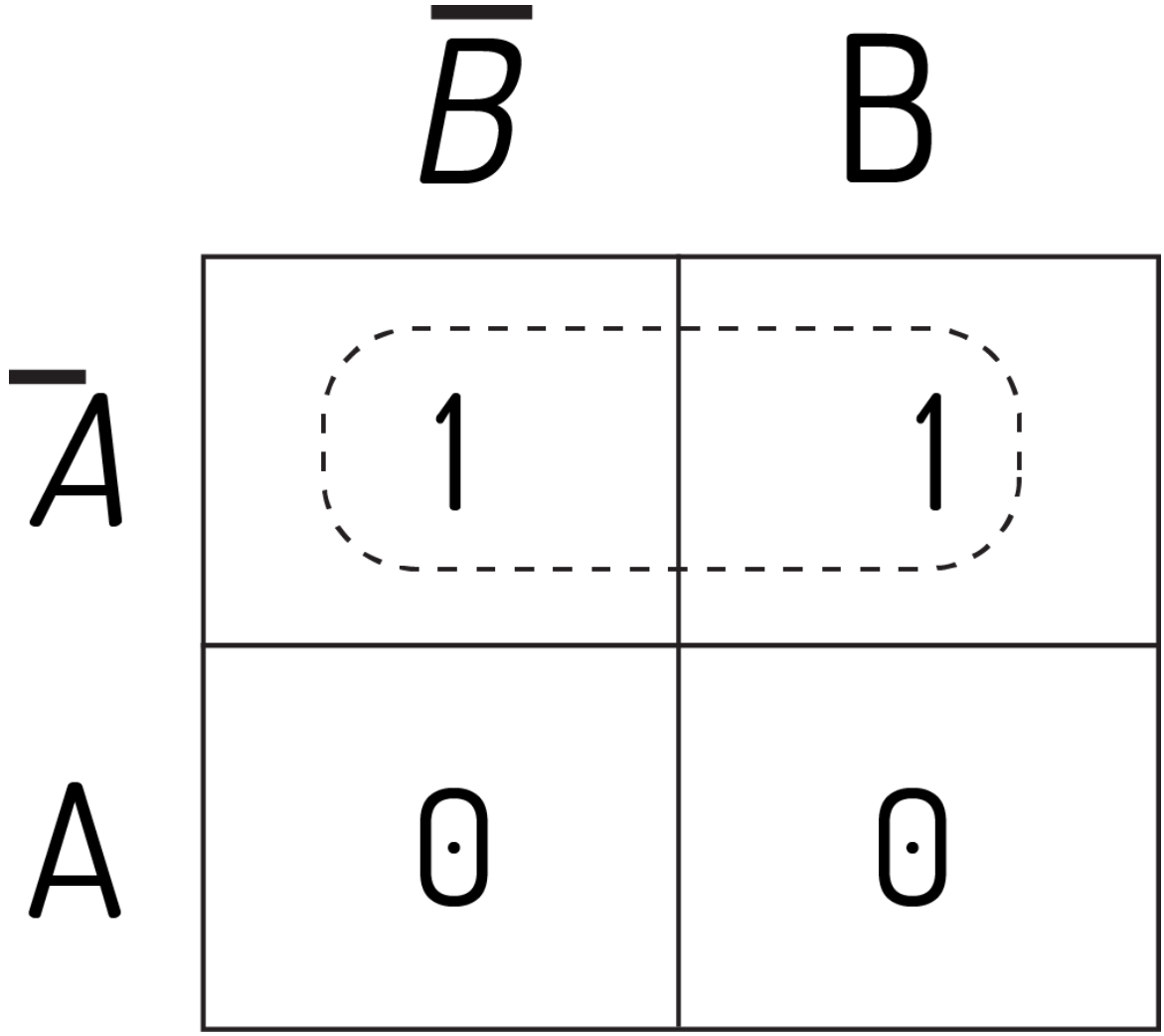
Agrupando-se os valores 1, obtemos um par. Portanto: $V_1 = A$

V_{m1} **(sinal vermelho do semáforo 1)**



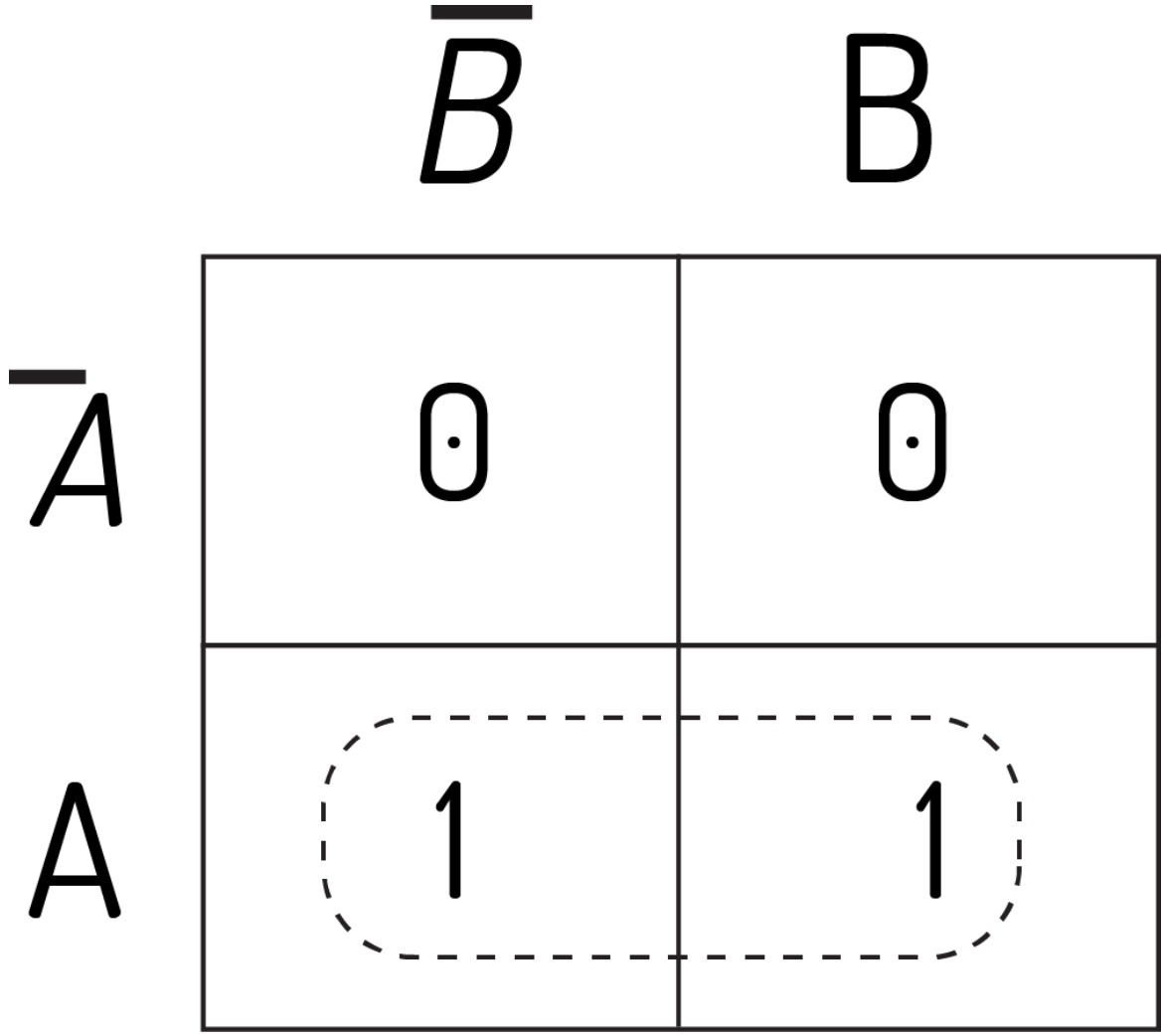
Agrupando-se os valores 1, também obtemos um par. Assim: $V_2 = \bar{A}$

V_2 (sinal verde do semáforo 2)



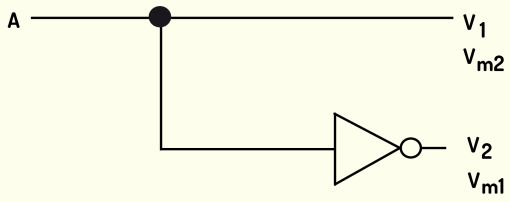
Novamente, o maior agrupamento é um par: $V_{m1} = \bar{A}$

V_{m2} **(sinal vermelho do semáforo 2)**



E outro par: $V_{m2} = A$

4. O circuito obtido é composto, então, somente por uma porta inversora, visto todo o controle depender somente da presença de carros na rua A. Esse desenho é apresentado na Figura 2.2.



2FIGURA 2.4 - Circuito lógico obtido FONTE: Idoeta (2012).

Projeto 2

Tocci (2011) propõe um projeto que gere um sinal que interrompa o funcionamento de uma máquina copiadora e ative um sinal luminoso quando ocorrem problemas.

Em uma simples máquina copiadora, um sinal de parada, S, é gerado para interromper a operação da máquina e ativar um indicador luminoso sempre que uma das condições a seguir ocorrerem: (1) a bandeja de alimentação de papel estiver vazia; ou (2) as duas microchaves sensoras de papel estiverem acionadas, indicando um atolamento de papel. A presença de papel na bandeja de alimentação é indicada por um nível ALTO no sinal lógico P. Cada uma das microchaves produz sinais lógicos (Q e R) que vão para o nível ALTO sempre que um papel estiver passando sobre a chave, que é ativada. Projete um circuito lógico que gere uma saída S em nível ALTO para as condições estabelecidas.

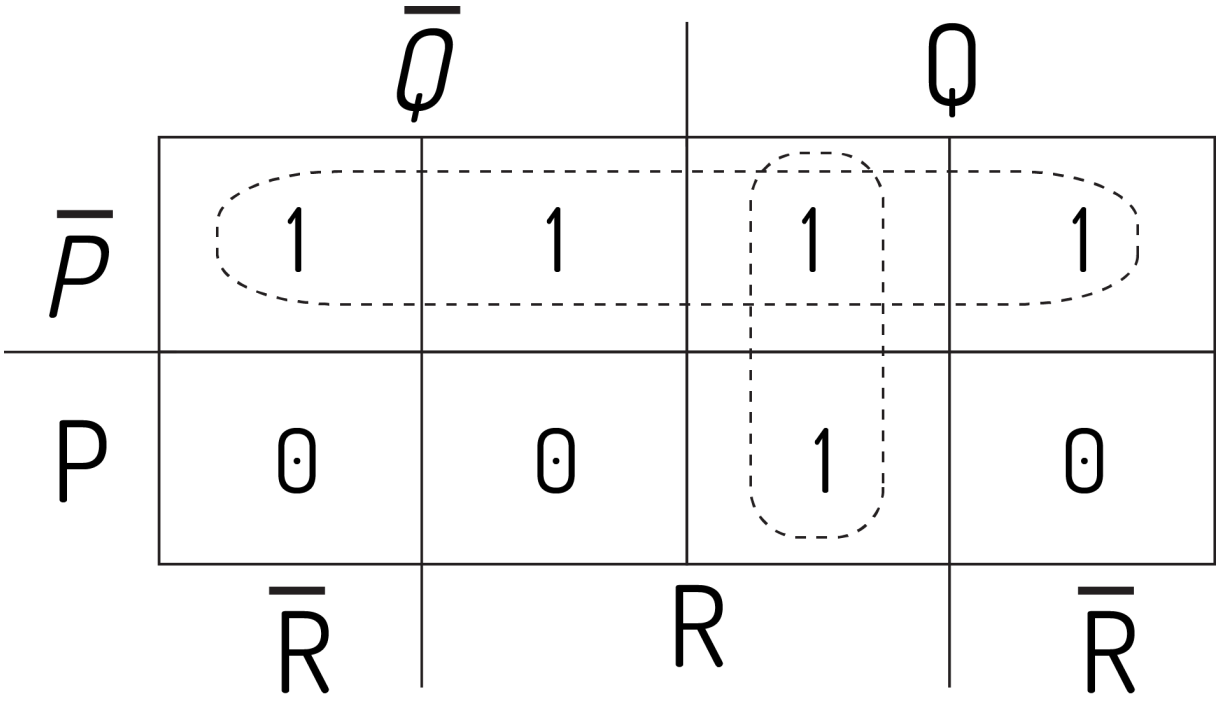
1. Definição das variáveis de entrada: P (presença ou não de papel), Q e R (passagem do papel pelas microchaves):
   Definição da variável de saída: X
2. Determinação da tabela verdade:
   Convenções:
3. Presença de papel na bandeja de alimentação: P = 1
4. Passagem de papel pelas microchaves (atolamento): Q = 1 e R = 1

Pela análise dos dados fornecidos, podemos verificar que a saída X estará em nível alto quando ocorrerem duas condições:

- P = 0 (falta papel na bandeja);
- Q e R = 1 (indica atolamento de papel).

SITUAÇÃO	P	Q	R	X
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

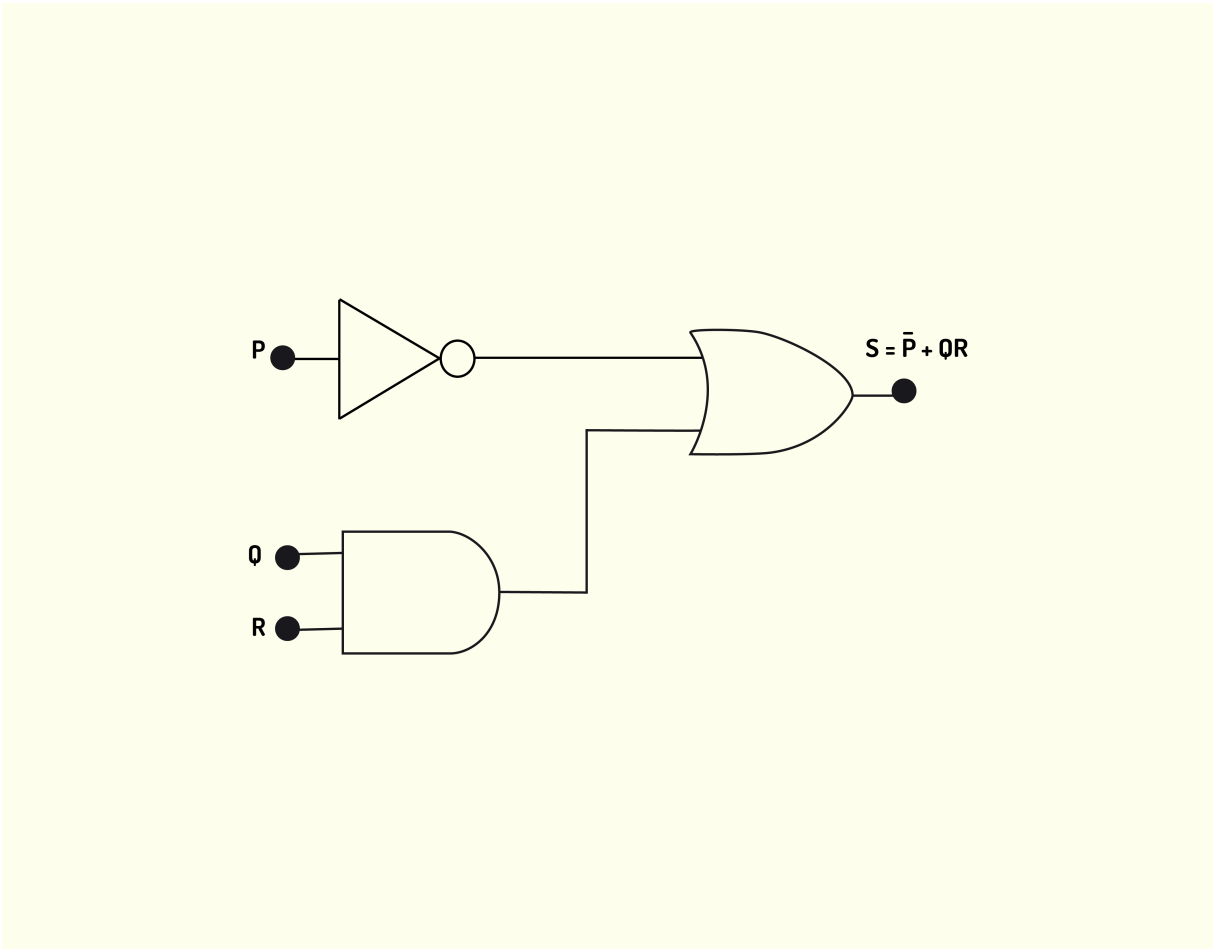
1. Obtenção e simplificação da equação booleana:



Agrupando-se os valores 1, temos:

$$X = \bar{P} + Q \,.\, R$$

4° e 5°) O circuito obtido é mostrado na Figura 2.3.

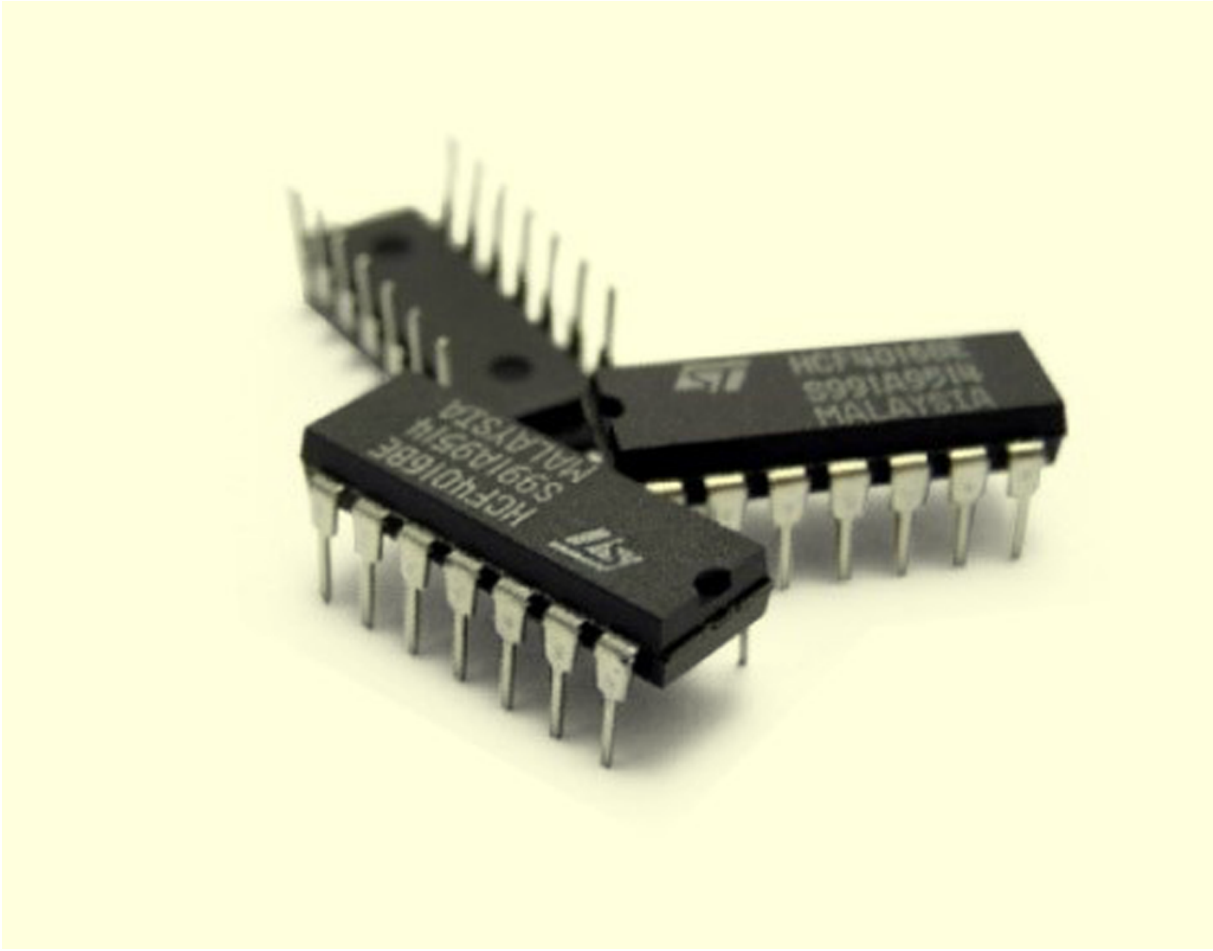


2FIGURA 3.4 - Circuito lógico obtido FONTE: Tocci (2011).

Terminado o projeto de um circuito combinatório, Monteiro (2012) informa: completa-se esta tarefa com a implementação física do circuito por meio da fabricação dos elementos e integrando-os em um só dispositivo (encapsulando-o), com o propósito de redução de espaço e custo, isto é, construindo um circuito integrado.

O mesmo autor complementa: um circuito integrado - CI (IC - *Integrated Circuit*) é um pequeno dispositivo, denominado pastilha (chip) que contém em seu interior centenas e atualmente milhares de componentes eletrônicos: transistores, diodos, resistores, capacitores e suas interligações. Estes componentes são os formadores das portas lógicas que, interligadas, formam um determinado circuito combinatório.

A Figura 2.4 mostra o desenho de pastilhas de circuito integrado.



2FIGURA 4.4 - Circuito integrado FONTE: Google imagens.

Alguns exemplos de circuitos combinatórios destinados a aplicações específicas são:

- Multiplexadores: possuem entradas de dados e controle e uma saída de dados. Esses circuitos podem ser utilizados para selecionar uma determinada entrada ou até mesmo como um conversor de dados de paralelo para serial. São também chamados de seletores.
- Decodificadores: possuem entradas e saídas de dados. É um circuito que pode ser utilizado para selecionar uma das linhas de saída.
- Comparadores: são circuitos que comparam duas palavras disponibilizadas na entrada, fornecendo o sinal 1 na saída se as elas forem iguais e 0 se forem diferentes.

Indicação de leitura

Nome do livro: Elementos de Eletrônica Digital

Editora: Érica

Autor: Francisco Gabriel Capuano / Ivan Valeije Idoeta

ISBN: 8571940193

Este livro é muito indicado para os conteúdos estudados nesta unidade, abordando diversos elementos utilizados no sistema digital e contemplando, de forma abrangente, as funções e portas lógicas, álgebra booleana, mapas de Vetch-Karnaugh e circuitos combinacionais. Disponibiliza vários exemplos e exercícios resolvidos, explicados de forma clara e precisa. Também propõe vários exercícios, fornecendo as respostas e expondo a teoria de forma simples e didática. É uma excelente opção para quem deseja estudar mais os conteúdos já trabalhados ou aprofundar seus conhecimentos.

Indicação de leitura

Nome do livro: O homem que calculava

Editora: Record

Autor: Malba Tahan

ISBN: 8501061964

O livro conta as aventuras do calculista persa Beremiz Samir, chamado de o Homem que Calculava. O personagem resolve e explica vários problemas e quebra-cabeças, complementando com curiosidades da matemática. As situações são resolvidas utilizando o raciocínio lógico, fazendo com que o(a) leitor(a) pense para resolvê-los. Como os processos de simplificação exigem bastante raciocínio, é uma forma lúdica e divertida de trabalhar com problemas lógicos.

UNIDADE III

Fundamentos de Sistemas Computacionais

Ana Cláudia de Oliveira Pedro Andrêo

Até o momento, aprendemos os conceitos básicos necessários sobre um sistema computacional, iniciando com os sistemas numéricos e suas conversões.

Na Unidade II, passamos realmente a entender como o hardware (parte física da máquina) é constituído, as portas lógicas que o formam e como projetar e simplificar circuitos combinatórios.

Nesta unidade, trabalharemos os fundamentos de Organização de Computadores.

Iniciaremos com os conceitos básicos de um sistema computacional, como é o funcionamento simplificado de um computador e quais suas funções elementares, além das operações que podem ser realizadas e como elas são repassadas para o computador, para que possam ser executadas.

Em seguida, será apresentado um breve histórico da evolução dos computadores, o que nos ajudará a compreender os principais componentes do sistema e a interligação entre eles.

Em seguida, realizaremos um estudo um pouco mais aprofundado da Unidade Central de Processamento (UCP), como ela funciona, quais as suas funções e como seu ciclo de instrução é constituído.

Finalizaremos com alguns conceitos de linguagem de montagem.

Conceitos básicos

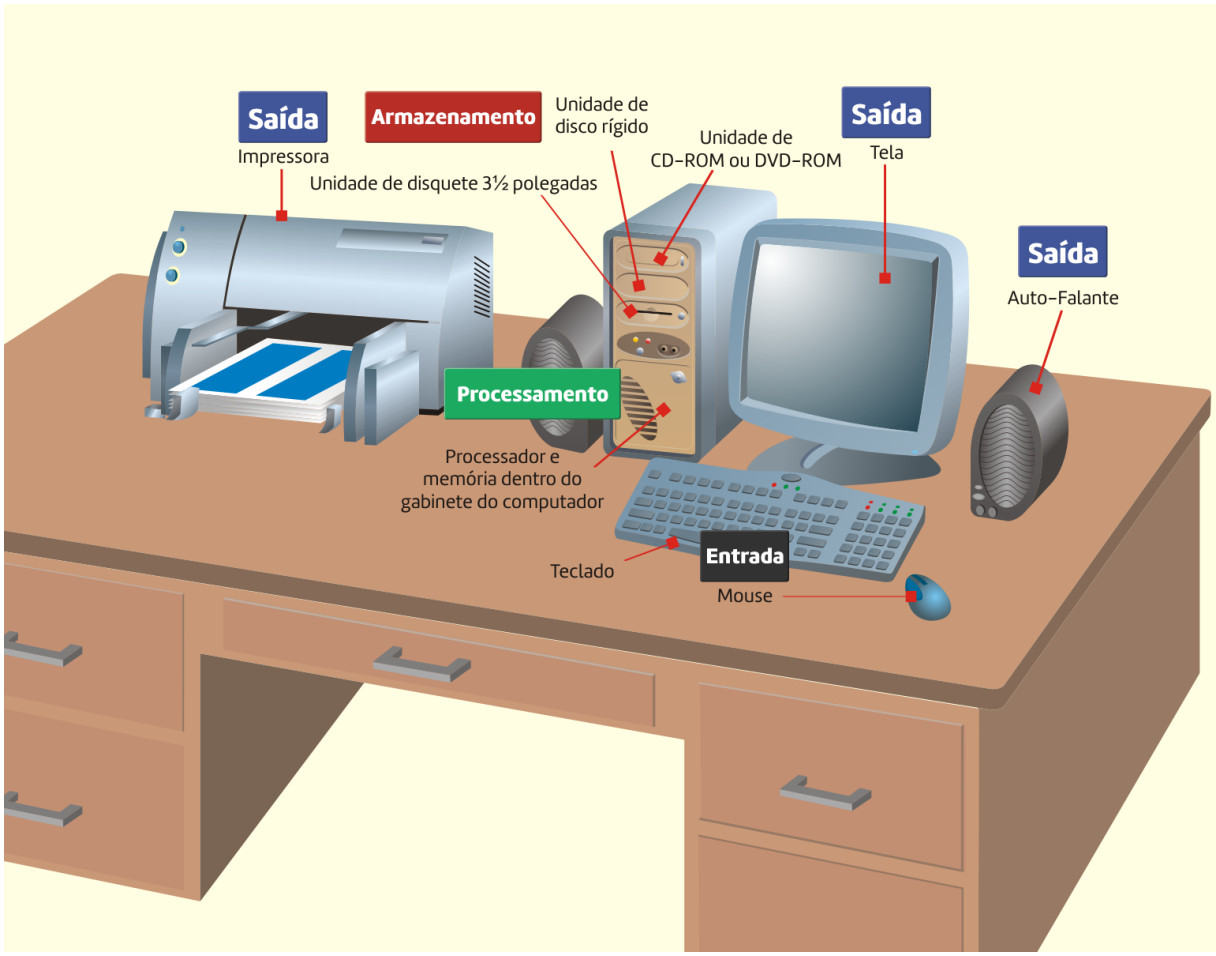
A humanidade, desde seus primórdios, possui uma necessidade muito grande de contar.

Segundo Ramos (2011), civilizações muito antigas já sentiam necessidade de quantificar e de expressar quantitativamente seu cotidiano. Os primeiros registros de quantificação não numérica foram entalhes em ossos, de aproximadamente 30000 a 20000 a.C. Muitas civilizações usaram objetos para auxiliar suas contagens, fazendo marcas e registros no chão, nas paredes ou em ossos; muitos desses instrumentos ou registros utilizavam a contagem um a um.

Nos dias de hoje, temos os computadores, que resolvem de uma maneira muito mais fácil essa necessidade da sociedade, realizando não só cálculos extremamente complexos, como também atividades que não possuem, aparentemente, nenhuma relação com números, como editores de texto, *softwares* de apresentação, programas para manipulação de imagens, entre outros.

O sistema computacional é formado pela combinação do *hardware* e do *software*. O *hardware* é a máquina em si, com seus circuitos digitais, ou seja, o equipamento. O *software* são os programas que comandam a máquina ou fazem com que o computador execute a tarefa a que se propõe, resolvendo problemas de naturezas diversas.

A Figura 3.1 apresenta um sistema computacional com seus principais componentes.



3FIGURA 1.19 - Sistema computacional FONTE: Capron (2004).

Weber (2012, p.27) acrescenta que:

> cada computador tem um conjunto de operações e convenções único para determinar as posições dos dados com os quais a operação será realizada. Os vários computadores diferem nas operações específicas que fornecem e nos métodos que usam para referenciar os dados que serão manipulados por uma operação.

As funções básicas que um computador pode realizar são: processamento de dados, armazenamento de dados, movimentação de dados e controle.

a. Processamento de Dados

Monteiro (2012, p.1) caracteriza que "***processamento de Dados (Data Processing) consiste, então, em uma série de atividades ordenadamente realizadas, com o objetivo de produzir um arranjo determinado de informações a partir de outras obtidas inicialmente".***

O mesmo autor complementa que processar o dado é executar com ele uma ação que produza algum tipo de resultado" (MONTEIRO, 2012, p.173).

Uma instrução é a responsável pela codificação de uma determinada operação.

Quando se tem um conjunto de instruções, colocadas de forma adequada (em sequência) forma-se um programa.

Weber (2012, p.27) define isso de uma maneira muito clara:

> um programa é constituído por uma sequência pré-determinada de instruções, que deve ser seguida para que seja atingido o objetivo computacional. Este programa e os dados correspondentes estão armazenados na memória da máquina; o conjunto de instruções (ou programa) deve ser interpretado para realização do processamento, isto é, a informação codificada correspondente às ações e aos operandos deve ser entendida e então processada.

b. Armazenamento de Dados

O armazenamento de dados e instruções é realizado pela memória de um sistema computacional. Essa memória é organizada em posições, cada uma com seu endereço específico.

c. Movimentação de Dados

A movimentação de dados diz respeito à comunicação do computador com o mundo exterior, o processo de entrada e saída.

Essa comunicação é realizada pelos dispositivos de entrada e saída, como mouses, monitores de vídeo e impressoras.

Exemplo: uma pessoa fornece o número do CEP da sua residência por meio do teclado e o programa, no monitor de vídeo, mostra o nome da rua e o bairro.

d. Controle

O controle é necessário para que se possa gerenciar as funções anteriores e é realizado pela Unidade de Controle, existente dentro do computador. Na seção sobre a Unidade Central de Processamento, detalharemos um pouco mais essa função.

Quando desenvolvemos um programa, geralmente ele é codificado utilizando linguagens chamadas de alto nível, como C, C++, C# ou Java. Todas essas linguagens possuem uma sintaxe específica, com definições e bibliotecas básicas e são mais simples para o programador, pois ele pode utilizar seu raciocínio de forma mais clara. Além disso, elas são independentes do computador para o qual são desenvolvidas. Assim, o mesmo programa pode ser utilizado em máquinas diferentes.

Mas o computador não é projetado para trabalhar com esse tipo de linguagem (lembre-se: ele trabalha somente com 0s e 1s) e sim com a chamada linguagem de máquina, que contém instruções de máquina, muito mais simples do que as utilizadas no alto nível.

As principais atividades que podem ser realizadas por uma instrução de máquina são:

- Operações aritméticas e lógicas;
- Movimentação de bits;
- Controle;
- Desvios na sequência do programa; e
- Comunicação com dispositivos de entrada e saída.

A linguagem de máquina utiliza mnemônicos para indicar a instrução ou operação que será executada, facilitando a programação. Mas esse programa precisa ser traduzido para a linguagem de máquina para ser executado pelo computador. O programa que realiza essa tradução é chamado de montador.

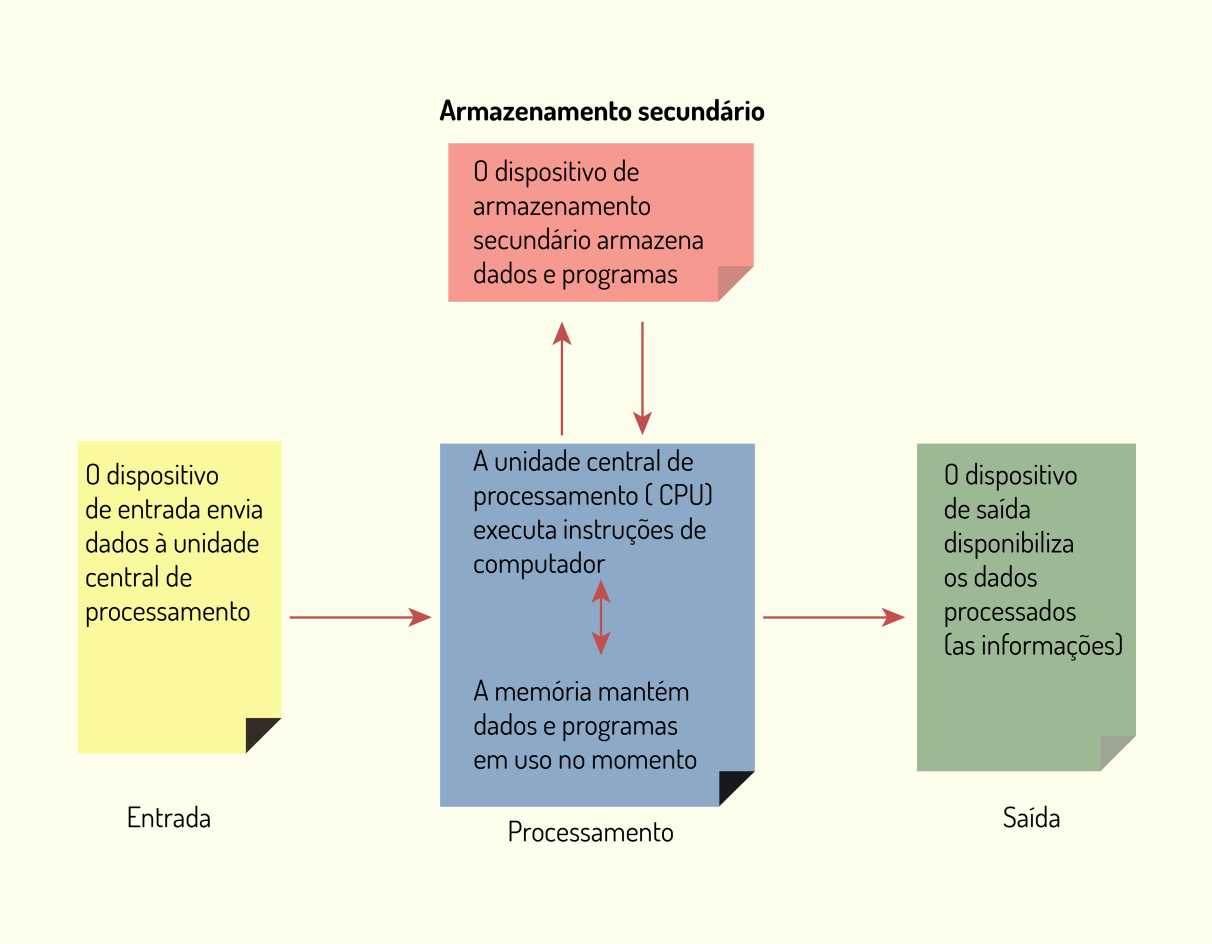
Um montador realiza praticamente apenas uma tradução "um para um" das instruções da linguagem simbólica para instruções de máquina (ao contrário de um compilador, que gera rotinas em linguagem de máquina para cada instrução da linguagem de alto nível e depois otimiza código e alocação de variáveis) (WEBER, 2012).

Existe, também, um outro método para se executar um programa, chamado interpretação que, segundo Monteiro (2012), se caracteriza por realizar as três fases (compilação, ligação e execução), comando a comando, do programa-fonte. Não há, pois, um processo explícito de compilação e ligação. Na realidade, um programa-fonte é diretamente executado (interpretado) por outro programa (o interpretador) e produz o resultado.

Funcionamento básico de um computador

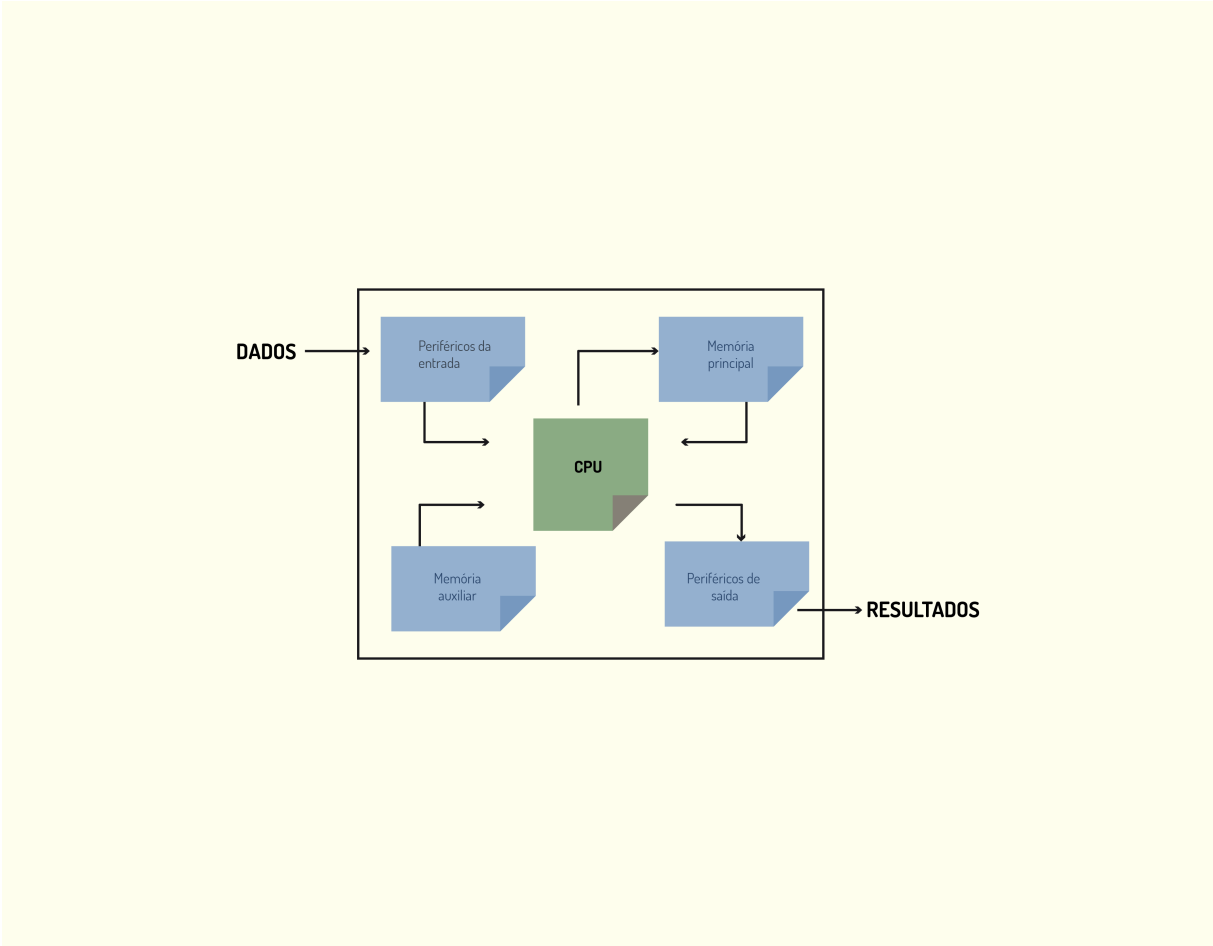
Complementando as informações anteriores, o computador é uma máquina que pode ser programada para aceitar dados (entrada), transformá-los em informações (saída) útil e armazená-los (em um dispositivo de saída secundário) para proteção ou reutilização. O processamento da entrada para a saída é conduzido pelo *software*, mas realizado pelo hardware (CAPRON, 2008).

A Figura 3.2 ilustra os quatro componentes principais de um computador.



3FIGURA 2.19 - Componentes principais de um computador FONTE: Capron (2004).

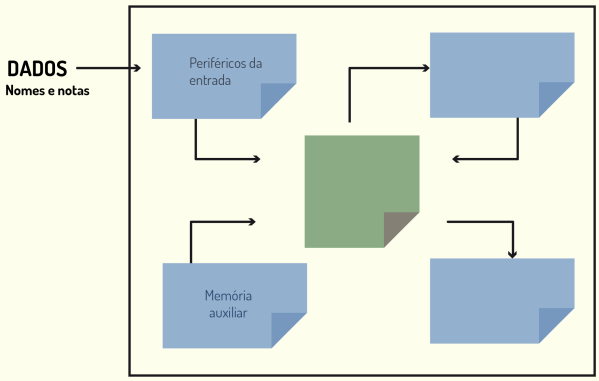
Para mostrar o funcionamento geral de um computador, vamos utilizar o esquema apresentado na Figura 3.3.



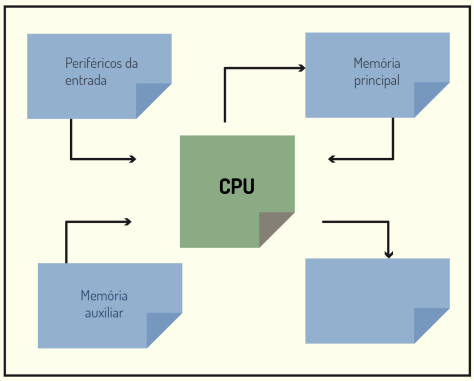
3FIGURA 3.19 - Esquema geral de um computador FONTE: A autora.

Vamos utilizar como exemplo o cálculo da média de notas de uma turma. Os passos necessários para esse cálculo podem ser divididos em:

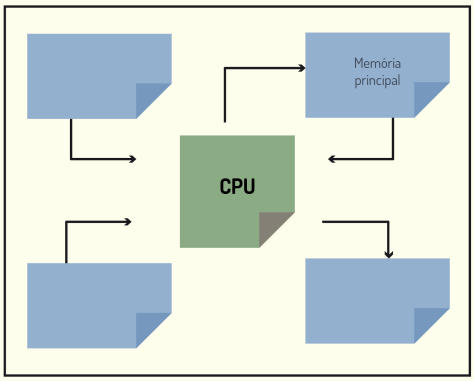
1. Os periféricos de entrada do computador recebem o nome e a média de cada um dos alunos da turma, conforme visto na Figura 3.4. Os dados podem vir também da memória auxiliar, onde estão armazenados.
2. A UCP transfere as informações recebidas no periférico de entrada para a memória principal, processo ilustrado na Figura 3.5.
3. A UCP trabalha com os dados armazenados na memória principal, neste caso calculando a média da turma, conforme mostra a Figura 3.6. Todos os valores intermediários de cálculo vão sendo armazenados na memória principal.
4. Quando os cálculos foram todos realizados, a UCP transfere o valor final para os periféricos de saída, representada na Figura 3.7.
5. E, por último, por meio do periférico de saída é mostrado o resultado final (neste caso, a média da turma) ao usuário, o que pode ser visto na Figura 3.8.



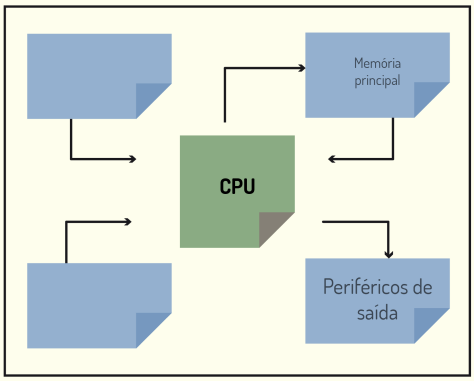
3FIGURA 4.19 - Processo de entrada FONTE: A autora.



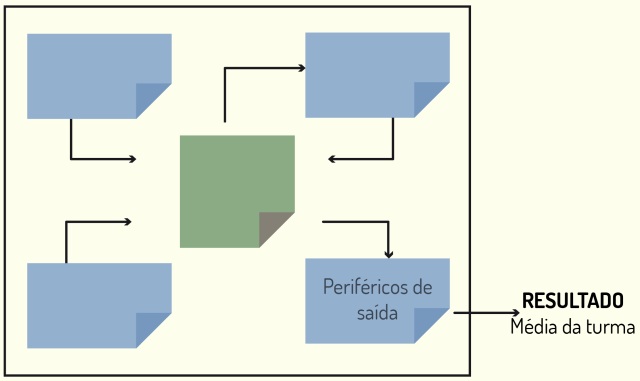
3FIGURA 5.19 - Transferência das informações FONTE: A autora.



3FIGURA 6.19 - Etapa de cálculos FONTE: A autora.



3FIGURA 7.19 - Transferência para os periféricos de saída FONTE: A autora.



3FIGURA 8.19 - Apresentação do resultado final FONTE: A autora.

Evolução Histórica dos Computadores

Atualmente, temos os computadores sendo utilizados nas mais diversas aplicações, seja na educação, no comércio, nos transportes, em controles financeiros, na agricultura, na medicina, dentre vários outros usos.

Mas a maior evolução pode ser observada na utilização de computadores por pessoas comuns, na realização de tarefas triviais do seu dia a dia. Como escrever um texto, comunicar-se com nossos amigos ou realizar uma pesquisa sobre um determinado assunto? Até quando precisamos de uma nova receita de doce consultamos a internet, que nos traz inúmeras opções.

Não podemos nos esquecer, também, que os celulares modernos também são um tipo de computador. Como viver sem eles?

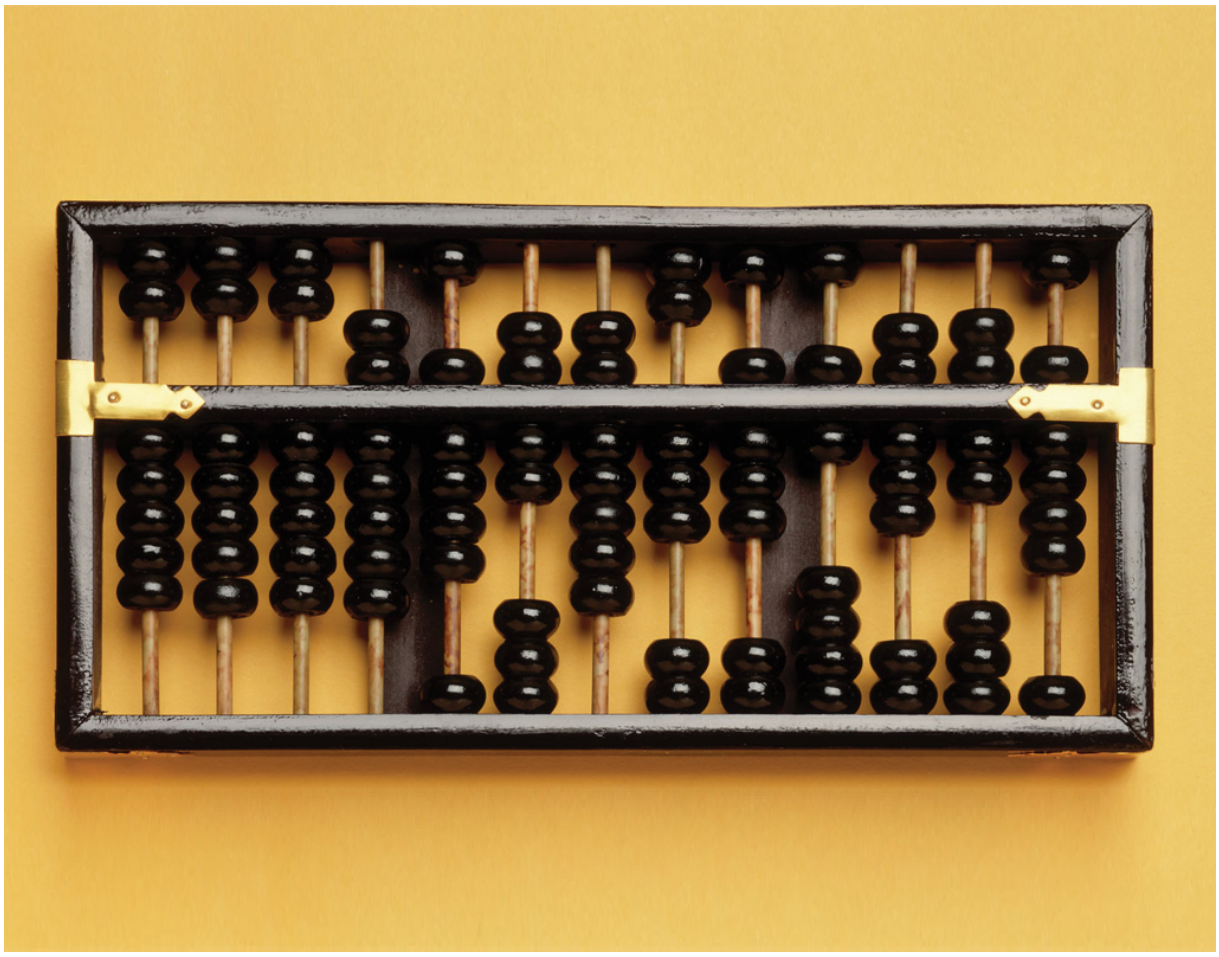
Pode-se caracterizar as principais evoluções dos computadores pensando em seu aumento de velocidade, diminuição do tamanho dos componentes, aumento da capacidade de memória e capacidade e velocidade do sistema de entrada e saída.

O aumento da velocidade está diretamente ligado à diminuição do tamanho dos componentes, pois isso reduz a distância entre eles e aumenta a velocidade.

O que há de notável a respeito da era da computação é que grande parte ocorreu em um tempo muito curto. Demos um salto ao longo de quatro gerações de tecnologia em aproximadamente 55 anos – um intervalo de tempo cujos eventos ainda estão na memória de muitas pessoas. As três primeiras "gerações" de computadores estão intimamente ligadas a três desenvolvimentos tecnológicos: a válvula a vácuo, o transistor e o circuito integrado. Cada um modificou drasticamente a natureza dos computadores. Definimos o tempo de ocorrência de cada geração de acordo com o início do fornecimento comercial da tecnologia de *hardware*. Definir as gerações seguintes não foi fácil, pois a indústria inteira tornou-se mais complexa (CAPRON, 2008).

Antes de falarmos das gerações de computadores, vamos relembrar a época dos dispositivos mecânicos e eletromecânicos. Pode-se falar que o primeiro computador inventado foi o ábaco, que é uma máquina de cálculo utilizada até os dias de hoje, principalmente no desenvolvimento do raciocínio matemático. Nele, os números são representados por símbolos de madeira que mudam de valor conforme a posição que ocupam no instrumento.

A Figura 3.9 mostra um ábaco.



3FIGURA 9.19 - O ábaco FONTE: Google imagens

Em 1642, o filósofo e matemático francês Blaise Pascal inventou a chamada primeira calculadora do mundo: a Pascaline, que foi uma evolução do ábaco. Essa calculadora era um contador mecânico formado por rodas e engrenagens (daí também vem o nome de rodas dentadas de Pascal) e realiza operações de soma e subtração.

A máquina, embora rudimentar, era eficaz para sua época, sendo inteiramente mecânica e não automática (funcionava por comando de uma manivela acionada manualmente). A linguagem de programação PASCAL foi assim chamada em homenagem a esse cientista pelo seu trabalho pioneiro em matemática e também devido a sua invenção (MONTEIRO, 2012).

Após algum tempo, Gottfried Leibniz construiu uma calculadora mais avançada que a de Pascal, que realizava, além das operações de soma e subtração, também as de multiplicação e divisão. O filósofo e matemático alemão introduziu mais dois conjuntos de rodas ao dispositivo de Pascal.

Em 1823, Charles Babbage passou a desenvolver sua máquina analítica. Sua intenção era de que ela possuísse a capacidade de modificar suas operações e realizasse diferentes cálculos, ou seja, possuísse um programa que pudesse ser alterado. O projeto da máquina de Babbage previa os mesmos elementos que um computador atual: memória, processador e entrada/saída.

Infelizmente, essa máquina nunca foi colocada em prática, por ser muito avançada para a época.

Herman Hollerith, em 1889, criou o cartão perfurado, utilizado para guardar dados, juntamente com uma máquina tabuladora mecânica. Essa máquina era acionada por um motor elétrico, que era capaz de contar, classificar e ordenar as informações do cartão perfurado. Portanto, essa já era uma máquina eletromecânica. O sucesso foi tanto que o Bureau of Census dos EUA contratou o cientista para apurar os dados do censo de 1890. Essa apuração demorou dois anos e meio, mas foi uma grande inovação.

Segundo Monteiro (2012), o problema dos computadores mecânicos e eletromecânicos residia em dois fatos: baixa velocidade de processamento, devido à parte mecânica de seus elementos, e falta de confiabilidade dos resultados, já que seu armazenamento e movimento interno eram realizados por engrenagens, incapazes de realizar sempre o mesmo tipo de movimento, principalmente com o desgaste causado pelo tempo.

Em 1937, o matemático inglês Alan Turing desenvolveu a máquina de Turing, que possuía componentes eletrônicos. De acordo com o matemático, se essa máquina fosse adequadamente instruída, poderia simular qualquer comportamento. Esse é um dos princípios da Inteligência Artificial, área na qual Turing foi um dos grandes precursores.

Recentemente, com a divulgação de documentos militares do governo britânico, antes sigilosos, é que se tomou conhecimento de que o primeiro computador verdadeiramente eletrônico foi colocado em operação em 1943, com o propósito de quebrar códigos militares secretos de comunicação dos alemães. Esta máquina, construída por Alan Turing com válvulas eletrônicas, foi denominada Colossus, provavelmente devido a seu tamanho. Sua grande desvantagem residia no fato de não ser uma máquina de emprego geral, pois não podia resolver outros problemas a não ser a quebra de códigos militares. Ela era, então, um sistema de computação com programa único (MONTEIRO, 2012).

¶ Reflita

A guerra é péssima para a humanidade, mas excelente para o desenvolvimento da tecnologia. Assim, a partir da Segunda Guerra Mundial, começou o desenvolvimento dos computadores eletrônicos.

A evolução desses computadores pode ser dividida em gerações, de acordo com as inovações que trouxeram.

Primeira geração – Válvulas (1945-1955)

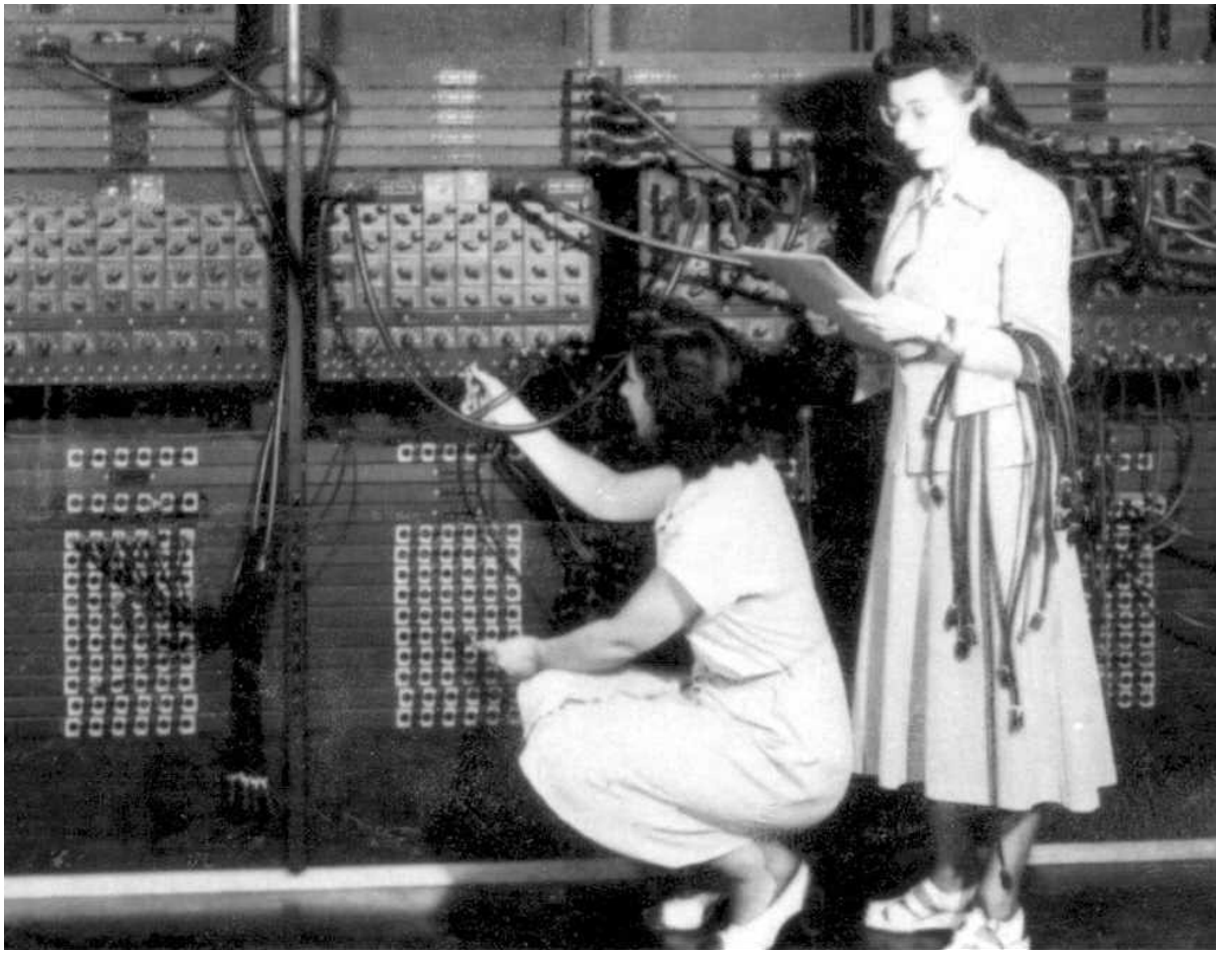
Os computadores dessa geração funcionavam com válvulas a vácuo, aproximadamente com o tamanho de uma lâmpada, e que eram utilizadas como componentes internos do computador e que permitiam ou não a passagem de corrente elétrica.

Segundo Souza Filho (2014), as instruções eram programadas diretamente em linguagem de máquina e gravadas em cartões perfurados, o que tornava o seu funcionamento lento e sua programação difícil de ser executada.

Um dos exemplos de máquina dessa geração é o ENIAC, que era uma máquina decimal, em vez de binária. Ele foi concluído em 1946 e, como não pôde mais ser utilizado na Segunda Guerra, ajudou a determinar a viabilidade da bomba de hidrogênio.

Essa máquina, de acordo com Stallings (2010), era enorme, pesava 30 toneladas, ocupando 1500 pés quadrados de superfície e contendo mais de 18000 válvulas. Quando estava em operação, ela consumia 140 kilowatts de potência. Ela também era mais rápida que qualquer computador eletromecânico, capaz de realizar 5000 adições por segundo.

A Figura 3.10 mostra o processo de programação do ENIAC.



3FIGURA 10.19 - Programando o ENIAC FONTE: **columbia.edu. <http://www.columbia.edu/cu/computinghistory/eniac.html>**

Essa geração também marcou o surgimento da máquina de Von Neumann, que será discutida mais a seguir.

Segunda geração – Transistores (1955-1965)

A segunda geração de computadores é caracterizada pela substituição das válvulas pelos transistores. Segundo Stallings (2010), o transistor é menor, mais barato e dissipa menos calor que uma válvula, mas pode ser usado da mesma forma que uma válvula para construir computadores. Diferente da válvula, que exige fios, placas de metal, uma cápsula de vidro e um vácuo, o transistor é um dispositivo de estado sólido, feito de silício.

Também fazem parte das evoluções dessa geração o surgimento das unidades aritméticas e lógicas mais complexas e da unidade de controle, além do armazenamento em disco e fita magnética.

Como se não bastassem todas essas inovações, a linguagem Assembly passou a substituir a linguagem de máquina e iniciou-se o uso de linguagens de programação de alto nível, como o FORTRAN.

Terceira geração – Circuitos integrados (1965-1980)

A invenção do circuito integrado de silício por Robert Noyce em 1958 permitiu que dezenas de transistores fossem colocados em um único chip. Esse empacotamento possibilitava a construção de computadores menores, mais rápidos e mais baratos do que seus precursores transistorizados (TANENBAUM, 2007).

Monteiro (2012) também cita outras evoluções dessa geração:

a. O conceito de família de computadores, em vez de máquina individual, como até então. Este conceito permite que o fabricante ofereça o mesmo tipo de máquina (arquitetura igual, linguagem de máquina semelhante etc.) com diferentes capacidades e preços, o que garante uma maior quantidade de clientes;
b. A utilização de unidade de controle com microprogramação em vez das tradicionais unidades de controle no *hardware*;
c. O emprego de uma técnica chamada multiprogramação, pela qual vários programas compartilham a mesma memória principal e dividem o uso da UCP, dando a impressão ao usuário de que estão sendo executados simultaneamente;
d. A elevada capacidade de processamento (para a época), com palavra de 32 bits e ciclo de instrução de até 250 nanossegundos, bem como a grande capacidade de armazenamento na memória principal, 16 Mbytes;
e. Memória principal orientada a byte, isto é, cada célula de MP armazena oito bits de informação, independentemente do tamanho de bits definido para a palavra de dados, característica utilizada até hoje;
f. O lançamento de um programa (conjunto de programas é o melhor termo) gerenciador dos recursos de *hardware*, de modo mais integrado e eficaz, o sistema operacional OS/360.

Além disso, os computadores também passaram a utilizar linguagens de alto nível, como FORTRAN e Cobol.

Quarta geração – Integração em escala muito grande (1980-dias atuais)

A quarta geração de computadores é caracterizada, principalmente, pelo surgimento do microprocessador, o que proporcionou o desenvolvimento dos computadores pessoais.

O termo VLSI (*Very Large Scale Integration*), integração em larga escala, caracteriza uma classe de dispositivos eletrônicos capazes de armazenar, em um único invólucro, milhares e até milhões de diminutos componentes. Este dispositivo, denominado pastilha (chip), vem constituindo a base da estrutura de todos os principais sistemas de computação modernos (MONTEIRO, 2012).

A Tabela 3.1 apresenta os principais marcos no desenvolvimento dos computadores modernos.

ANO	NOME	CONSTRUÍDO POR	COMENTÁRIOS
1834	Máquina analítica	Babbage	Primeira tentativa de construir um computador digital
1936	Z1	Zuse	Primeira máquina de calcular com relés
1943	COLOSSUS	Governo britânico	Primeiro computador eletrônico
1944	MarkI	Aiken	Primeiro computador norte-americano de uso geral
1946	ENIAC	Eckert/Mauchley	A história moderna dos computadores começa aqui
1949	EDSAC	Wilkes	Primeiro computador com programa armazenado
1951	Whirlwind I	M.I.T.	Primeiro computador de tempo real
1952	IAS	von Neumann	A maioria das máquinas atuais usa esse projeto
1960	PDP-I	DEC	Primeiro minicomputador (50 vendidos)
1961	1401	IBM	Máquina para pequenos negócios de enorme popularidade
1962	7094	IBM	Dominou a computação científica no início da década de 1960
1963	B5000	Burroughs	Primeira máquina projetada para uma linguagem de alto nível
1964	360	IBM	Primeira linha de produto projetada como uma família
1964	6600	CDC	Primeiro supercomputador científico
1965	PDP-8	DEC	Primeiro minicomputador de mercado de massa (50 mil vendidos)
1970	PDP- 11	DEC	Dominou os minicomputadores na década de 1970
1974	8080	Intel	Primeiro computador de uso geral de 8 bits em um chip
1974	CRAY-I	Cray	Primeiro supercomputador vetorial
1978	VAX	DEC	Primeiro superminicomputador de 32 bits
1981	IBM PC	IBM	Deu início à era moderna do computador pessoal
1981	Osborne-I	Osborne	Primeiro computador portátil
1983	Lisa	Apple	Primeiro computador pessoal com uma GUI
1985	386	Intel	Primeiro ancestral de 32-bits da linha Pentium
1985	MIPS	MIPS	Primeira máquina comercial RISC
1987	SPARC	Sun	Primeira estação de trabalho RISC baseada em SPARC
1990	RS6000	IBM	Primeira máquina superescalar
1992	Alpha	DEC	Primeiro computador pessoal de 64 bits
1993	Newton	Apple	Primeiro computador palmtop

3QUADRO 1.2 - Alguns marcos do desenvolvimento do computador digital moderno FONTE: Tanenbaum (2007).

Máquina de Von Neumann

A máquina de Von Neumann foi concebida durante a primeira geração de computadores, em 1946, mas sua arquitetura continua sendo usada até hoje.

John Von Neumann, segundo Tanenbaum (2007), era um gênio, da mesma estirpe de Leonardo da Vinci. Falava muitos idiomas, era especialista em ciências físicas e matemática e guardava na memória tudo que já ouvira, vira ou lera. Conseguia citar, sem consulta, palavra por palavra, o texto de livros que tinha lido havia anos. Na época em que se interessou por computadores, já era o mais eminente matemático do mundo.

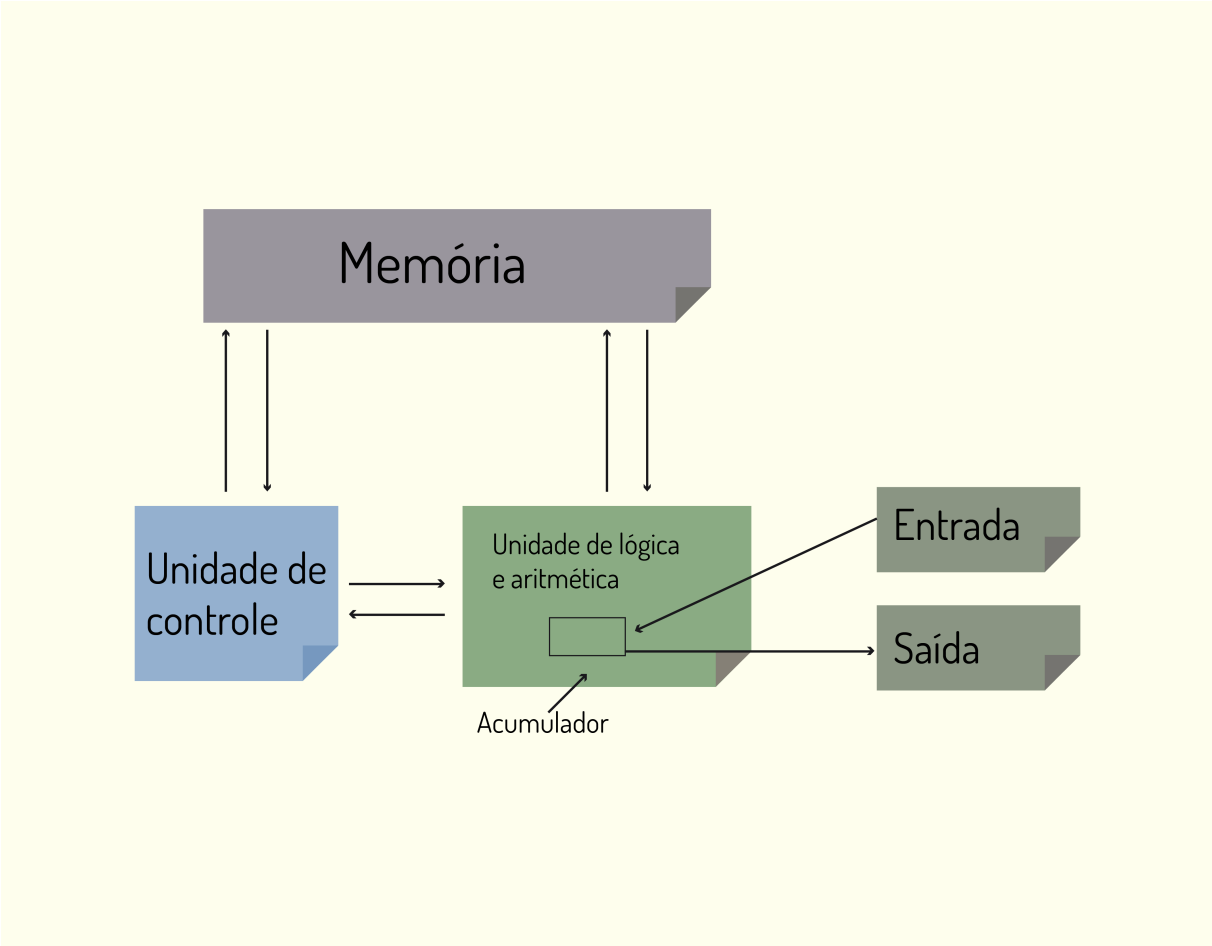
O autor complementa que:

> uma das coisas que ficou óbvia para ele foi que programar computadores com quantidades imensas de interruptores e cabos era uma tarefa lenta, tediosa e mecânica. Ele percebeu que o programa podia ser representado em forma digital na memória do computador, junto com os dados. Também viu que a desajeitada aritmética decimal usada pelo ENIAC, com cada dígito representado por 10 válvulas (1 acesa e 9 apagadas), podia ser substituída usando aritmética binária paralela .
>
> (TANENBAUM, 2007, p.10)

A arquitetura concebida por Von Neumann era composta por:

a. A UCP, formada pela Unidade Aritmética e Lógica e a Unidade de Controle.
   Esses componentes possuíam a mesma função utilizada até hoje: a unidade aritmética e lógica era responsável por realizar operações sobre dados agora binários e a unidade de controle por interpretar e executar as instruções armazenadas na memória.
b. A Memória Principal, responsável por armazenar dados e instruções (essa foi uma grande inovação para a época).
c. O equipamento de entrada e saída, controlado pela unidade de controle.
d. O acumulador, que é um registrador interno especial.

A Figura 3.11 mostra um esboço da arquitetura de Von Neumann.



3FIGURA 11.19 - Máquina original de Von Neumann FONTE: Tanenbaum (2007).

Componentes de um Sistema de Computação

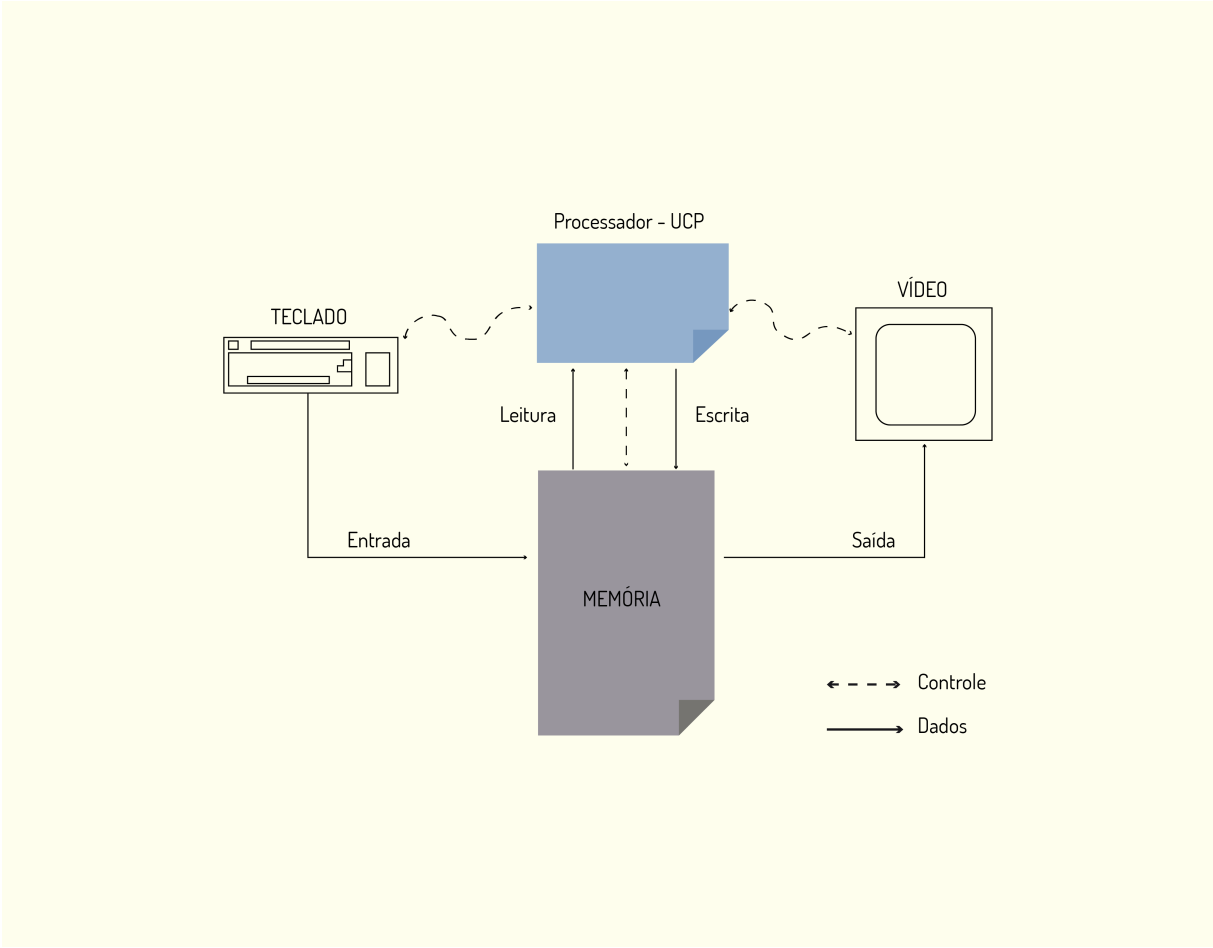
Como foi dito no início da nossa unidade, os computadores funcionam utilizando programas, que são um conjunto de instruções que efetuam ações ou operações visando a um objetivo comum.

Eles são formados por vários elementos acoplados, chamados componentes que, juntos, funcionam como um sistema só.

Os componentes principais de um sistema computacional são:

- A Unidade Central de Processamento (UCP);
- A Memória;
- Os dispositivos de entrada; e
- Os dispositivos de saída.

A Figura 3.12 mostra estes componentes e sua interligação.



3FIGURA 12.19 - Componentes de um sistema de computação FONTE: Monteiro (2012).

Unidade Central de Processamento

A Unidade Central de Processamento (UCP) ou, em inglês, *Central Processing Unit* (CPU) ou processador, é o "cérebro" do sistema computacional.

Para Monteiro (2012), uma UCP ou processador é constituída de milhões de minúsculos circuitos e componentes eletrônicos (transistores, resistores etc.), cujas funções básicas são ler e interpretar instruções de máquina e realizar as operações matemáticas (ou outras) definidas após a interpretação de uma determinada instrução.

Por ser um componente tão importante, teremos, logo a seguir, um item dedicado somente a ela.

Memória

Segundo Souza Filho (2014), um espaço de memória pode conter uma instrução de um programa ou um dado qualquer, que serão endereçados na memória pela unidade de controle da UCP. Os dados que serão processados pela ULA ficam na memória e a unidade de controle endereça estes dados. Isto permite que a ULA identifique onde estão os dados a serem

processados, execute as operações necessárias, e a unidade de controle pode definir onde armazenar os dados resultantes do processamento. A memória que recebe esse endereçamento é usada para receber as informações da unidade de entrada e as processadas pelo computador é a memória RAM.

A ULA (*Arithmetic Logic Unit*) ou ULA (Unidade Aritmética e Lógica) é a responsável, dentro do processador, pelas operações matemáticas e lógicas.

¶ Reflita

A arquitetura de Von Neumann prevê a possibilidade de uma máquina digital armazenar os programas e os dados no mesmo espaço de memória, e estes serão processados por uma unidade de processamento central (CPU) composta por uma unidade controle e uma unidade aritmética e lógica (ULA). Os dados são fornecidos mediante dispositivos de entrada e retornados por meio dos dispositivos de saída (RAINER, 2012).

Os dispositivos de memória podem ser de quatro tipos: registradores, memória cache, memória principal e memória secundária. Eles serão estudados com mais detalhes na nossa próxima unidade.

A memória é formada por elementos armazenadores de informação. Uma memória está dividida em palavras. Cada palavra é identificada por um endereço. O conteúdo armazenado nas palavras da memória tanto pode representar dados como instruções (WEBER, 2012).

Ou seja, para que se possa armazenar ou localizar uma informação na memória, precisamos de um endereço, que define um local de forma precisa e única.

Existem dois tipos de ações que podem ser realizadas na memória:

a. Armazenamento de informações, que é a operação de escrita e gravação; e
b. Recuperação de informação, que consiste na operação de leitura.

Assim, podemos definir os princípios de operação da memória como sendo:

1. Selecionar o endereço de memória que está sendo acessado por uma operação de leitura ou escrita;
2. Selecionar uma operação de escrita ou de leitura para ser executada;
3. Fornecer os dados de entrada para serem armazenados na memória durante a operação de escrita;
4. Manter os dados de saída vindos da memória durante uma operação de leitura;
5. Habilitar (ou desabilitar) a memória de modo que ela responda (ou não) às entradas de endereço e ao comando de leitura e escrita.

As memórias diferem pela sua tecnologia de fabricação.

Tipos de memória RAM (*Random Access Memory*)

a. Estáticas (SRAM - *static* RAM): são formadas por *flip-flops*, que permanecem em um determinado estado indefinidamente, desde que a alimentação do circuito seja mantida.
b. Dinâmicas (DRAM - *dynamic* RAM): armazenam os sinais em capacitores, possuindo alta capacidade e baixo consumo. Como os capacitores possuem uma fuga de carga, há a necessidade de recargas periódicas das células de memória (*refresh*). Este é o tipo de memória utilizada nos computadores atuais.

Tipos de memória ROM (*Read Only Memory*)

A ROM é programada pelo fabricante, mas pode ser alterada, podendo ser classificada em:

a. PROM (ROMs programáveis)

A PROM pode ser programada pelo usuário, mas somente uma vez. O processo geralmente é realizado por um programador de PROM, que também verifica se os dados armazenados estão corretos.

b. EPROM (ROM programável e apagável – *Erable Programable* ROM)

Uma EPROM pode ser programada pelo usuário e também ser apagada e reprogramada quantas vezes for desejado. Uma vez programada, a EPROM é uma memória não volátil que mantém indefinidamente os dados armazenados (TOCCI, 2011).

A EPROM pode ser apagada mediante a exposição à luz ultravioleta, que apaga todas as células ao mesmo tempo.

c. EEPROM (PROM apagável eletricamente – *Eletrically Erasable* PROM)

A EEPROM pode ser apagada eletricamente no próprio circuito, permitindo que se apague e reescreva bytes individualmente.

d. Flash

A memória Flash também pode ser apagada eletricamente, sem a remoção do circuito. Seu nome é derivado do tempo curto de apagamento e escrita. Oferecem um modo de apagamento por setor, no qual setores específicos podem ser apagados de cada vez.

Dispositivos de entrada e saída

Os dispositivos de entrada e saída são utilizados para permitir a comunicação entre o computador e o mundo externo. Esses dispositivos são também chamados de dispositivos de I/O ou dispositivos periféricos. Podem ser divididos em duas categorias: dispositivos que são utilizados como memória secundária e dispositivos que servem para interface homem-máquina.

Os dispositivos utilizados como memória secundária, como discos rígidos e pen drives, se caracterizam por um custo relativamente baixo, porém o tempo de acesso à memória secundária é bem maior que o acesso à memória principal.

Os dispositivos que servem de comunicação homem-máquina são os que realmente conhecemos como sendo de entrada e saída, que realizam a interface entre as pessoas e o sistema computacional, podendo ser chamados também de periféricos.

Dispositivos de entrada

Como o próprio nome diz, os dispositivos de entrada são responsáveis pela inserção de dados no sistema para que possam ser processados.

Os principais periféricos de entrada que encontramos comercialmente são:

- O teclado, que é o meio mais comum de comunicação entre o homem e a máquina;
- O mouse, que é um dispositivo utilizado para apontar e selecionar informações no vídeo.

Atualmente, podemos citar, também, o joystick, a caneta ótica, scanners, leitores de código de barras e as telas sensíveis ao toque.

A Figura 3.13 mostra alguns dispositivos de entrada, como o teclado, mouse e um leitor de código de barras.

3FIGURA 13.19 - Dispositivos de entrada FONTE: Capron (2004).

Dispositivos de saída

O monitor de vídeo é um dispositivo utilizado para mostrar ao usuário os dados por ele digitados e os resultantes das operações da UCP. É o meio mais utilizado para a saída dos dados.

Já as impressoras são os dispositivos mais usados para apresentar os resultados das operações do computador no papel.

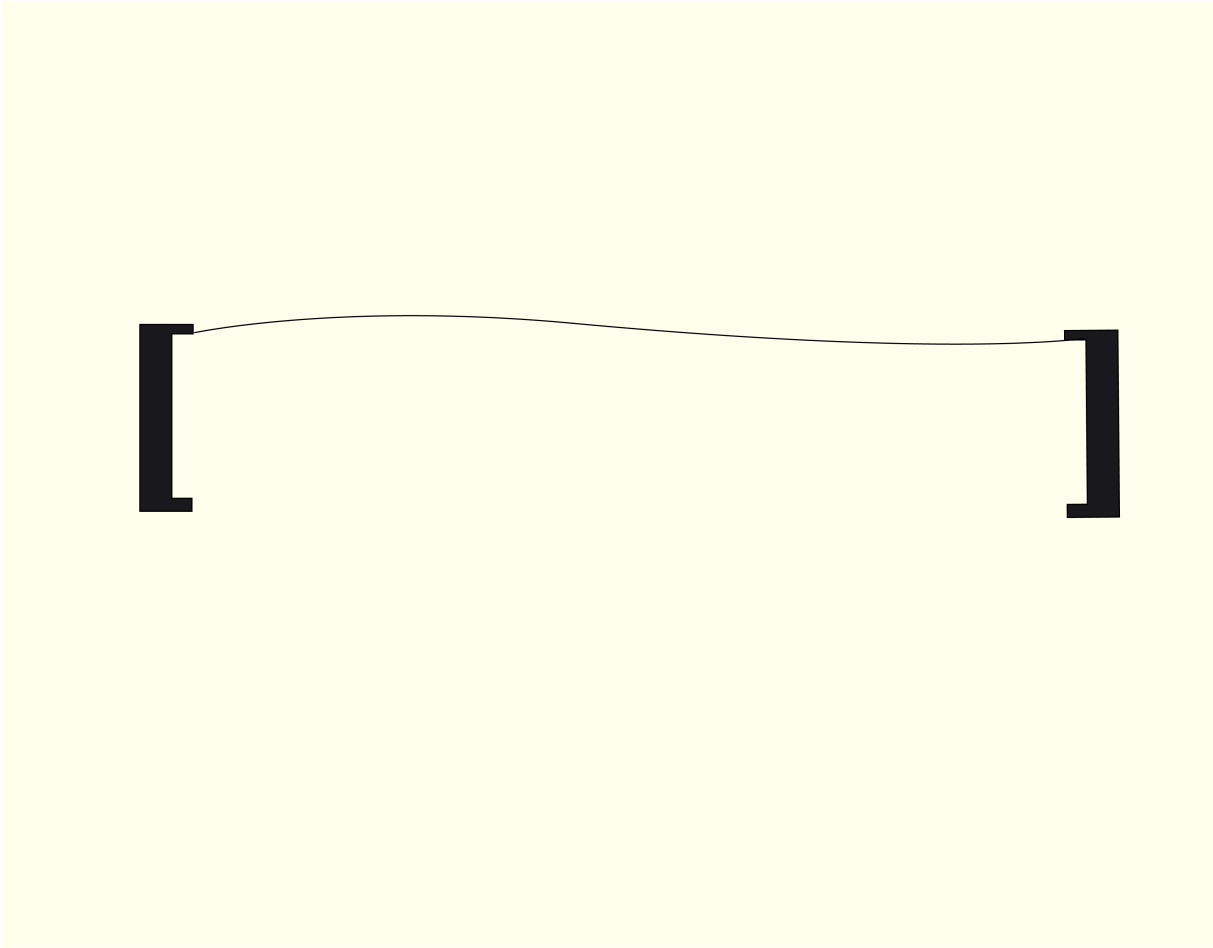
Podemos ter, também, como periférico de saída, as caixas de som.

Interconexão entre os componentes

Mas para que todos os componentes de um sistema de computação trabalhem corretamente, é necessário uma ligação entre eles, o chamado barramento.

Um barramento é um caminho de comunicação que conecta dois ou mais dispositivos. Uma característica-chave de um barramento é que ele é um meio de transmissão compartilhado. Múltiplos dispositivos se conectam ao barramento, e um sinal transmitido por qualquer dispositivo está disponível para recepção por todos os outros dispositivos conectados ao barramento. Se dois dispositivos transmitirem durante o mesmo período, seus sinais serão sobrepostos e ficarão distorcidos. Assim, somente um dispositivo de cada vez pode transmitir com sucesso (STALLINGS, 2010).

Monteiro (2012) apresenta, na Figura 3.14, o exemplo de um cabo de ligação (apresentação física de um barramento) entre um periférico e um processador, constituído de diversos fios paralelos bem próximos um dos outros, cada um conduzindo um bit da informação que está sendo transferida (ou um sinal de controle diferente).



3FIGURA 14.19 - Exemplo de um barramento FONTE: Monteiro (2012).

Os barramentos podem ser classificados em:

- Barramento local: interliga a UCP aos dispositivos de maior velocidade, como a memória cache e a memória principal. Esse barramento geralmente funciona na mesma frequência da UCP, sendo o de maior velocidade de transferência de dados;
- Barramento do sistema: em alguns modelos de computador, a UCP não acessa diretamente a memória principal e sim a memória cache. O barramento do sistema realiza a ligação entre a memória cache e a memória principal;
- Barramento de expansão: é utilizado para interligar os dispositivos de entrada e saída.

Representação das informações

Toda informação introduzida em um computador (sejam dados que serão processados ou instruções de um programa) precisa ser entendida pela máquina, para que possa corretamente interpretá-la e processá-la (MONTEIRO, 2012).

Como discutimos em nossa Unidade II, o computador não "pensa". Ele apenas realiza operações com os sinais recebidos e armazenados, que podem ser representados por sinais elétricos.

A partir desses conhecimentos, conseguimos entender por que o sistema computacional trabalha com o sistema binário e não o decimal. Isso ficou bem claro quando falamos do ENIAC em nosso histórico. Ele utilizava o sistema decimal. Veja o tamanho dele e a potência que era consumida.

O sistema binário é composto por somente dois dígitos, 0 e 1, chamados bits (*binary digit*). O bit é a menor quantidade de informação que podemos ter (lembra das palavras cruzadas? Isso sempre aparece).

Mas somente um bit não representa praticamente nada. Assim, utilizamos agrupamentos deles para poder representar as informações. Assim, temos, primeiramente, o byte, que é composto por 8 bits.

Como os principais códigos de representação de caracteres utilizam grupos de oito bits por caractere, os conceitos de byte e caractere tornam-se semelhantes, e as palavras, quase sinônimas. O termo caractere é mais empregado para fins comerciais (propaganda, apresentações a pessoas não familiarizadas com o jargão de computação), enquanto o termo byte é empregado mais na linguagem técnica dos profissionais da área (MONTEIRO, 2012).

Pensando bem, 8 bits também é muito pouco para expressarmos quantidades de informação, somente um caractere.

Assim, surgiram os múltiplos dos bytes, que podem ser vistos na Tabela 3.2.

TERMO	SÍMBOLO	NÚMERO APROXIMADO DE BYTES
Quilobyte	K (ou KB)	um mil
Megabyte	MB	um milhão
Gigabyte	GB	um bilhão
Terabyte	TB	um trilhão
Petabyte	PB	um quatrilhão

3QUADRO 2.2 - Capacidades de armazenamento FONTE: Capron (2004).

¶ Reflita

Os múltiplos utilizados na informática são praticamente iguais aos do sistema métrico, mas existe uma diferença fundamental. Quando falamos em 1 Kg, estamos falando em 10^3 gramas, ou mil gramas. No sistema binário, como trabalhamos com potências de 2 (essa é a base nesse sistema), passamos a ter, em 1 KB, 2^{10} bytes, o equivalente a 1024 bytes.

Classificação de sistemas de computação

Os sistemas computacionais podem ser classificados em: microcomputadores, estações de trabalho, minicomputadores, computadores de grande porte e supercomputadores.

a. Microcomputadores (desktops, laptops, notebooks, palmtops)

Os microcomputadores são também chamados de computadores pessoais, ou PCs. É o tipo de sistema de computação mais difundido no mercado, possuindo, inclusive, a possibilidade de serem ligados em rede.

Temos várias categorias de microcomputadores, que variam de acordo com seu tamanho e portabilidade, como os desktops (computadores de mesa), laptops (portáteis de tamanho razoável), notebooks (portáteis de tamanho menor) e os palmtops, que são quase do tamanho de uma palma da mão (vem daí o seu nome).

A Figura 3.15 ilustra o desktop e o notebook.

3FIGURA 15.19 - Microcomputadores FONTE: Google imagens.

Os Tablets, iPads e Smartphones também se enquadram nesta categoria de computadores pessoais.

b. Estações de trabalho (*workstations*)

As estações de trabalho são pequenas o bastante para caberem em uma escrivaninha e contam com grande capacidade de processamento de dados, superior a muitos computadores de grande porte (CAPRON, 2008).

Elas são utilizadas para fins específicos, como na área científica ou industrial, geralmente para se trabalhar com programas de imagem e projeto.

c. Minicomputadores

Segundo Capron (2004), os computadores midrange (anteriormente chamados de minicomputadores) são computadores multiusuário projetados para atender às necessidades das organizações de porte médio. Centenas ou, às vezes, milhares de usuários podem estabelecer conexão com um computador midrange por meio de terminais ou PCs ligados em rede para acessar aplicativos em toda a empresa, como entrada de pedidos e controle de estoque.

d. Computadores de grande porte (*mainframes*)

Monteiro (2012, p.35) explica que:

> os computadores de grande porte são sistemas projetados para manusear considerável volume de dados e executar simultaneamente programas de uma grande quantidade de usuários. Essas máquinas podem interagir com centenas de usuários em um dado instante, como, por exemplo, um sistema de reserva de passagens aéreas, onde há necessidade de armazenamento em grande escala (todos os dados de vôos e das reservas realizadas), bem como uma contínua solicitação de processamento por parte dos incontáveis terminais conectados diretamente ao sistema, aos quais o computador tem que atender e responder em poucos segundos.

A Figura 3.16 mostra um *mainframe* fabricado pela IBM.

3FIGURA 16.19 - Mainframe FONTE: Google imagens

e. Supercomputadores

É o tipo de computador mais poderoso e rápido.

Segundo Capron (2004), podem ser encontrados em aplicações prevalecentes tão variadas quanto análise de ações, design de automóveis, efeitos especiais cinematográficos e até mesmo ilustrações gráficas sofisticadas. Entretanto, durante muitos anos, seus clientes foram um grupo exclusivo – órgãos do Governo Federal -, que os utilizavam para tarefas que requerem uma gigantesca manipulação de dados, como a previsão meteorológica que engloba o mundo inteiro e pesquisas de armamento.

Desempenho dos computadores

Os computadores têm se desenvolvido cada vez mais rapidamente, seja com a utilização de novas tecnologias para a fabricação de seus componentes ou modificações na forma como o ciclo de instruções é realizado. Mas ainda temos os mesmos princípios básicos de arquitetura concebidos por Von Neumann.

Na busca do aumento de desempenho, verifica-se que a medida geral deste desempenho depende fundamentalmente da capacidade e velocidade de seus diferentes componentes, da velocidade com que estes componentes se comunicam entre si e do grau de compatibilidade que possa existir entre eles (por exemplo, se a velocidade da UCP de um sistema é muito maior

que a da memória, então esse sistema tem um desempenho inferior ao de um outro em que a UCP e a memória têm velocidades mais próximas) (MONTEIRO, 2012).

Isto significa que o computador deve se desenvolver como um todo, pois as diferentes velocidades existentes nele afetam o desempenho global do sistema.

Segundo Ricarte (2016), como há um grande número de alternativas para a organização de um computador, é preciso ter mecanismos que permitam realizar a avaliação de cada arquitetura. Algumas medidas básicas de avaliação são necessárias para tanto. O desempenho está usualmente associado à velocidade de execução de instruções básicas (taxas MIPS e FLOPS) ou à velocidade de execução de programas representativos das aplicações (*benchmarks*). Lembrando que MIPS significa milhões de instruções por segundo e FLOPS, operações de ponto flutuante por segundo.

Em essência, o termo *benchmark* utilizado na computação tem o mesmo significado do *benchmark* utilizado no mundo corporativo, por exemplo, já que visa a comparação de mecanismos, processos, objetos e resultados. Geralmente, na computação, o termo "*benchmarking*" é associado com avaliação de características de desempenho de um *hardware*, mas também pode ser aplicado a *software*, desde que se leve em consideração dados apenas técnicos (CANALTECH, 2016).

¶ Fique por dentro

Medidas de desempenho

Medidas de desempenho dependem do ponto de vista individual. Um usuário de computador está mais preocupado com o tempo de resposta: o tempo total para o sistema realizar uma tarefa. Administradores de sistema estão mais preocupados com vazão: quantas tarefas concorrentes o sistema pode realizar sem afetar de forma adversa seu tempo de resposta. Estes dois pontos de vista são inversamente relacionados (ANDRADE, 2016).

Temos várias medidas de desempenho:

a. Tempo de resposta: é o intervalo de tempo entre a requisição do usuário e a resposta do sistema. É uma medida relacionada ao desempenho do sistema como um todo e não de seus componentes individuais.
b. Tempo de acesso: é relativo ao uso da memória, quando são realizadas operações com a mesma (escrita e leitura). Este tempo depende da velocidade da UCP, da memória e da interligação entre elas.
c. Vazão (*throughput*), ou taxa de processamento, ou ainda taxa de serviço: é a quantidade executada de serviço por unidade de tempo.

Almeida (2009) apresenta mais algumas métricas para o desempenho:

- Eficiência: capacidade útil/capacidade nominal;
- Utilização: porcentagem de tempo que o recurso está executando o serviço;
- Custo-benefício = custo/desempenho;
- Métricas específicas: porcentagem de perda de pacotes, qualidade do sinal.

Unidade Central de Processamento (UCP) e Linguagem de Montagem

Tanenbaum (2007, p.29) define:

> a CPU (Central Processing Unit – unidade central de processamento) é o "cérebro" do computador. Sua função é executar programas armazenados na memória principal buscando suas instruções, examinando-as e então executando-as uma após a outra. Os componentes são conectados por um barramento, que é um conjunto de fios paralelos que transmitem endereços, dados e sinais de controle. Barramentos podem ser externos à CPU, conectando-a à memória e aos dispositivos de E/S, mas também podem ser internos à CPU.

Na realidade, a UCP é responsável pela realização de qualquer operação realizada por um computador. Isto quer dizer que a UCP comanda não somente as ações efetuadas internamente, como também, em decorrência da interpretação de uma determinada instrução, ela emite os sinais de controle para os demais componentes do computador agirem e realizarem alguma tarefa (MONTEIRO, 2012).

Como acabamos de ver, a UCP é responsável por uma grande quantidade de atividades. Elas podem ser divididas em duas categorias: função processamento e função controle.

Função processamento

A função processamento se encarrega de realizar as atividades relacionadas com a efetiva execução de uma operação, ou seja, processar (MONTEIRO, 2012).

Algumas das operações que podem ser realizadas por essa parte da UCP são:

- Aritméticas (soma, subtração, multiplicação e divisão);
- Lógicas (as mesmas realizadas pelas portas lógicas);
- Movimentação de dados (por exemplo, entre memória e UCP);
- Desvios (alterações na sequência normal do programa);
- Entrada e saída.

A parte da UCP que realiza a função processamento é formada pela UAL (Unidade Aritmética e Lógica) ou ULA (Unidade Lógica e Aritmética) ou, ainda, em inglês, ALU (*Arithmetic Logic Unit*) e pelos registradores, interligados pelo barramento interno.

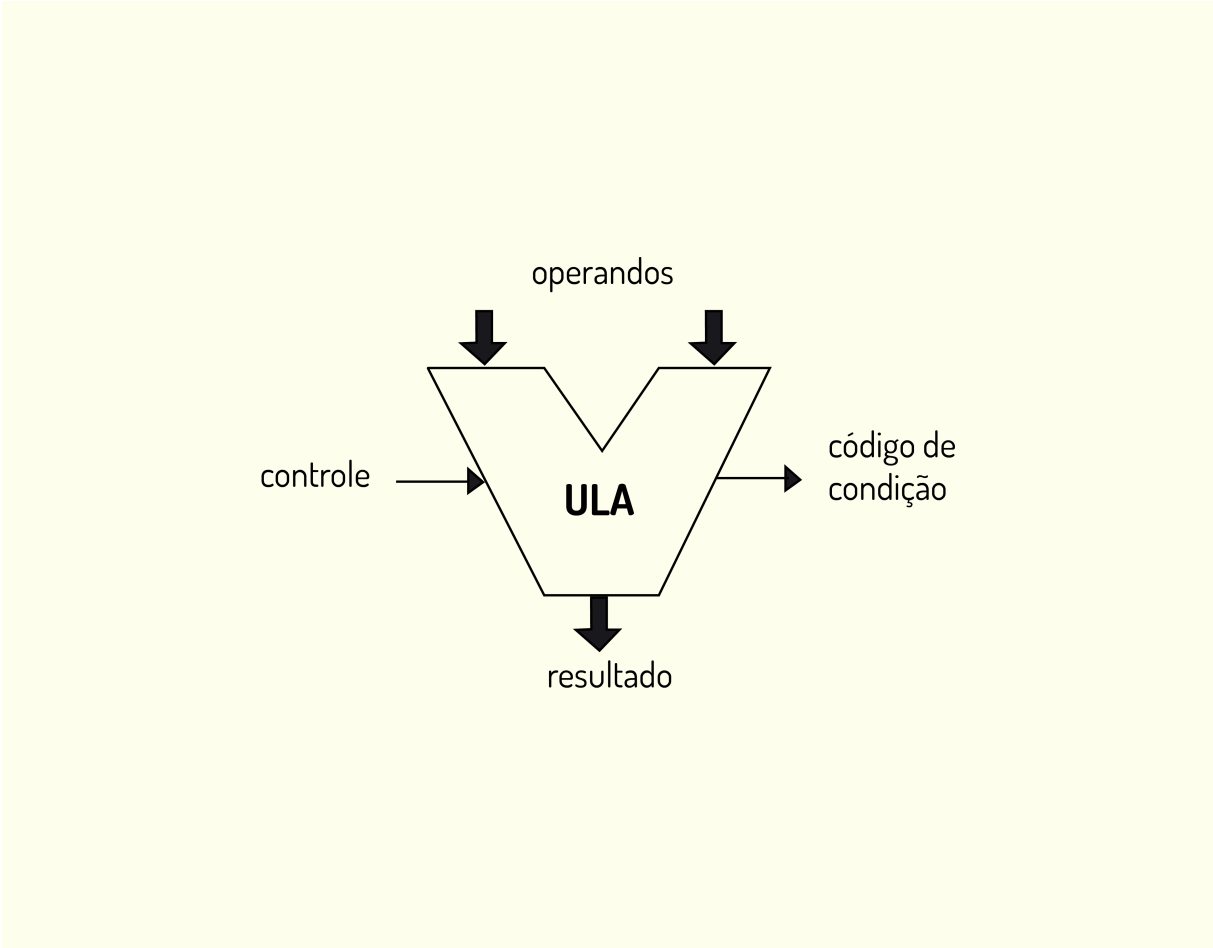
Unidade Aritmética e Lógica

Como o próprio nome diz, a Unidade Aritmética e Lógica é responsável pelas operações matemáticas e lógicas que podem ser realizadas pelo computador, citadas anteriormente.

A UAL possui duas entradas, pois pode realizar operações com um ou dois valores. Por exemplo, as operações de soma e subtração necessitam de dois valores na entrada, mas a de complemento somente um.

Quando o resultado é obtido, por meio da passagem dos sinais pelos circuitos lógicos que compõem a UAL, estes dados são repassados à saída pelo barramento interno de dados.

A Figura 3.17 apresenta o desenho esquemático de uma UAL.



3FIGURA 17.19 - Modelo estrutural da UAL FONTE: Weber (2012).

Registradores

Os registradores são um tipo de memória interna à UCP, extremamente rápidos, mas presentes em pequena quantidade. São áreas de armazenamento temporário, para guardar dados, instruções e endereços, que estão sendo utilizados na operação que está sendo executada.

Os principais registradores presentes na UCP são:

a. ACC (Acumulador): serve de ligação entre a UCP e os outros dispositivos dela. É quem armazena os resultados obtidos na UAL.
b. RDM (Registrador de Dados da Memória): contém o dado lido da memória ou que deverá ser escrito nela. Em inglês é chamado de *Memory Buffer Register* (MBR).
c. REM (Registrador de Endereços da Memória): contém o endereço do dado que deverá ser lido ou escrito na memória. Em inglês é chamado de *Memory Address Register* (MAR).
d. RI (Registrador de Instrução): contém a instrução que foi lida mais recentemente. Em inglês, é chamado de *Instruction Register* (IR).
e. CI (Contador de Instrução): contém o endereço da próxima instrução que será lida. Em inglês, é chamado de *Program Counter* (PC).
f. RST (Registrador de Estado): armazena os bits de estado, referentes ao resultado de operações realizadas, como os de sinal, *overflow*, zero, vai 1 e paridade. Seu nome, em inglês, é *Program Status Word* (PSW), ou palavra de estado.
g. Registradores de uso geral: registradores genéricos presentes na UCP que a auxiliam em suas atividades.

Função controle

A função controle é exercida pelos componentes da UCP que se encarregam das atividades de busca, interpretação e controle da execução das instruções, bem como do controle da ação dos demais componentes do sistema de computação (memória, entrada/saída) (MONTEIRO, 2012).

O principal elemento da função controle é a Unidade de Controle, que é considerado o dispositivo mais complexo da UCP.

Unidade de Controle

Capron (2008, p.94) define:

> a unidade de controle contém circuitos que usam sinais elétricos para coordenar o computador inteiro para realizar ou executar instruções armazenadas de um programa. Como um maestro de orquestra, a unidade de controle não executa instruções de programa: ao contrário, ela comanda outras partes do sistema para isso. A unidade de controle deve comunicar-se tanto com a unidade lógica e aritmética quanto com a memória.

Assim, a unidade de controle gera todos os sinais de controle para comandar a máquina (microeventos ou micro-operações), emitidos de tempos em tempos, coordenados pelo relógio (*clock*).

Os sinais para estes microeventos ou micro-operações podem ser gerados de duas formas: programados no *hardware* (*hardwired control*) ou por microprogramação.

A diferença básica entre os dois métodos, segundo Monteiro (2012), está no processo de controle da realização do ciclo de instrução. No primeiro caso (*hardwired*), cada etapa é realizada segundo uma lógica preestabelecida, implementada fisicamente no *hardware* da área de controle. No caso de controle microprogramado, a interpretação e as consequentes etapas do ciclo de instrução são realizadas passo a passo por um programa, denominado microprograma.

¶ Fique por dentro

Funções processamento e controle

Podemos fazer uma analogia com os seres humanos, imaginando que a área de controle é o cérebro que comanda o ato de andar, e a área de processamento são os músculos e ossos das pessoas que realizam efetivamente o ato. Os nervos são análogos ao barramento de interligação entre os diversos elementos (MONTEIRO, 2012).

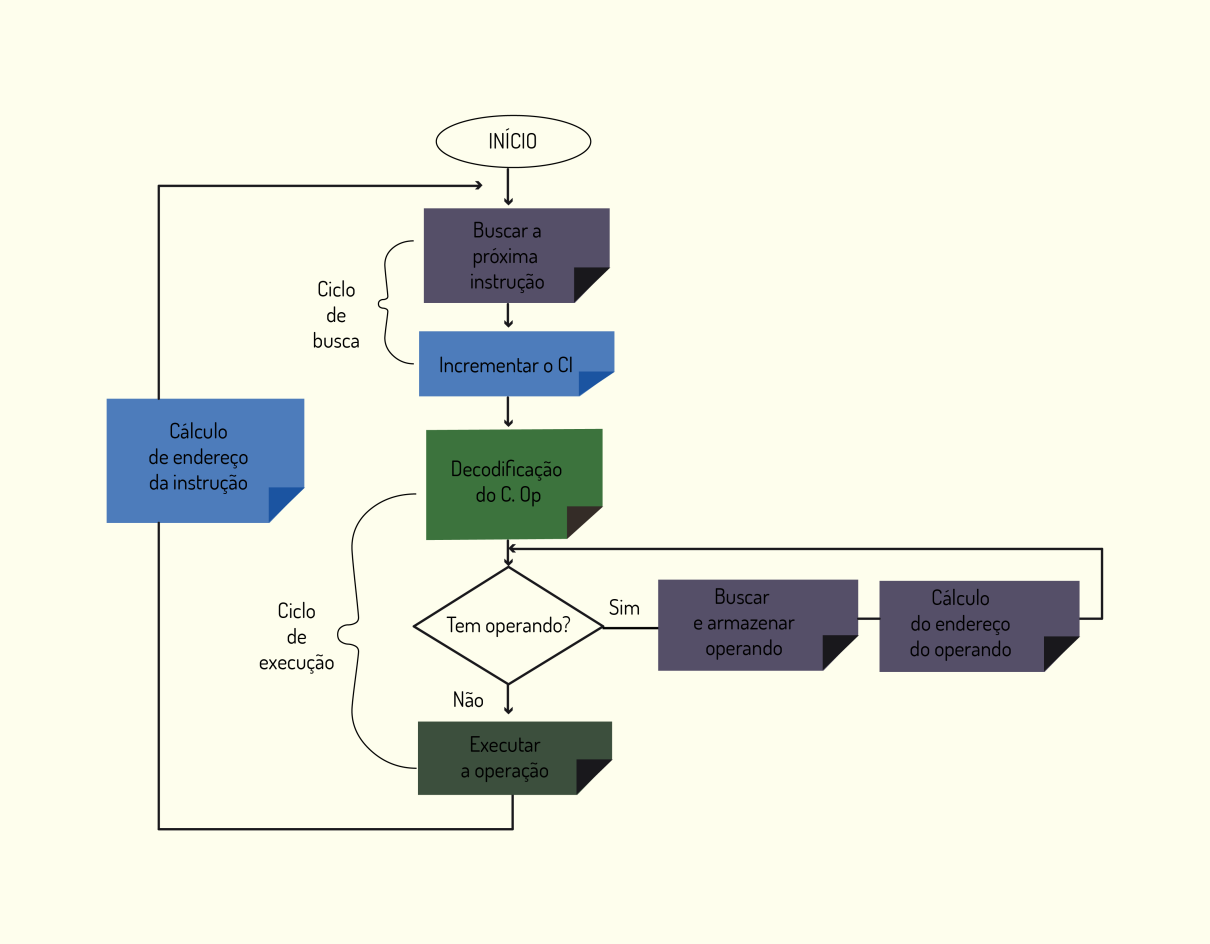
Ciclo de instrução

Conforme vimos anteriormente, a UCP é responsável por executar os programas, que são compostos de instruções. Agora, veremos como isto é feito, o que chamamos de ciclo de instrução. Cada uma destas instruções é dividida em etapas.

Tanenbaum (2007) define as seguintes etapas, chamadas de ciclo buscar-decodificar-executar:

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

A Figura 3.18 ilustra essas etapas.

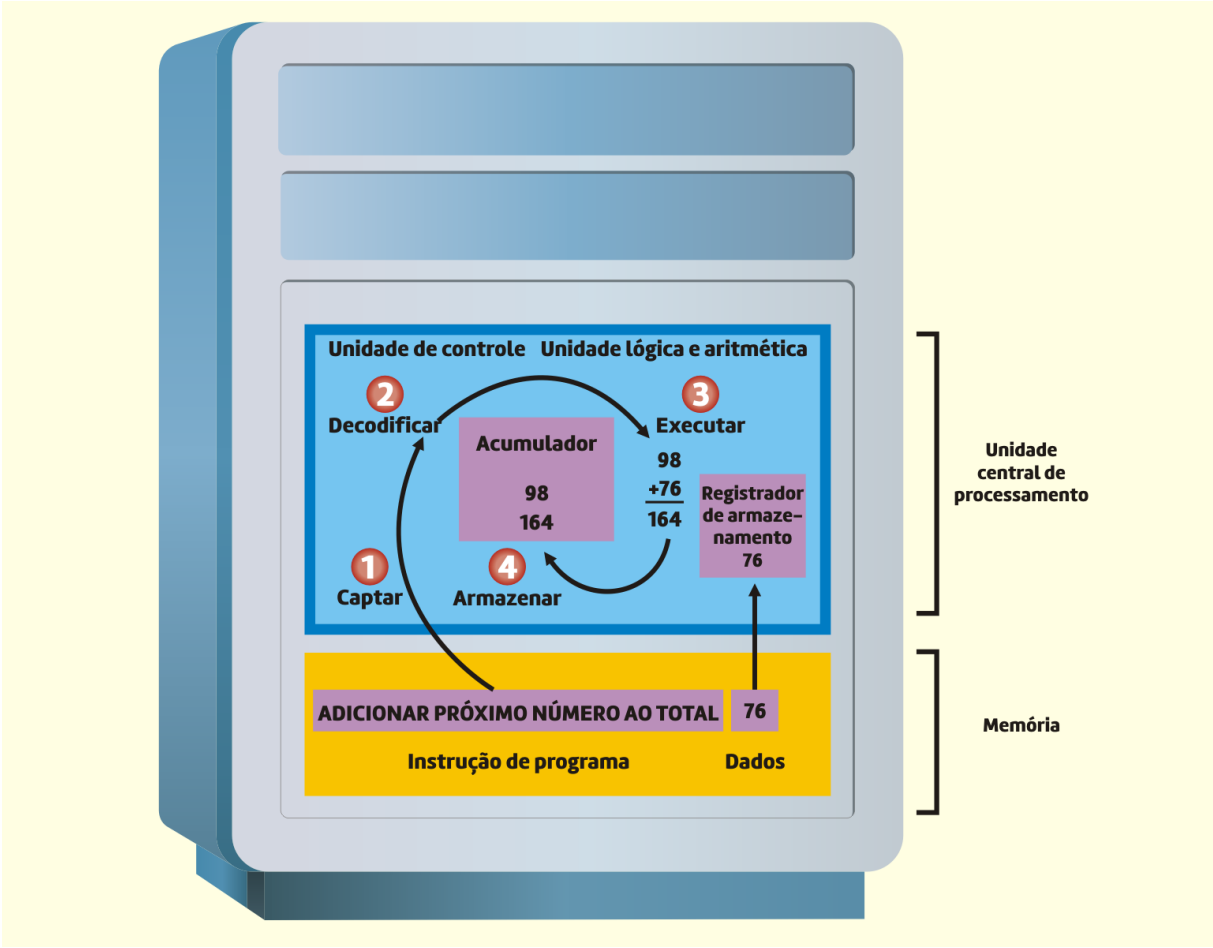


3FIGURA 18.19 - Fluxograma de um ciclo de instrução FONTE: Monteiro (2012).

Capron (2008, p.98) apresenta um exemplo simples do ciclo de instrução:

> suponhamos que um programa queira encontrar a média de cinco pontuações de exame. Para isso, deve totalizar as cinco pontuações e depois dividir o resultado por cinco. O programa inicia fixando o total em 0. Depois, adiciona cada um dos cinco números, um por vez, ao total. Suponhamos que as pontuações sejam 88, 76, 91, 83 e 87. Nesse cálculo, o total foi fixado em 0 e, depois, 88, a primeira pontuação do exame, foi adicionada. Examine agora o ciclo de máquina quando ele adiciona a pontuação seguinte, 76, ao total. Siga as etapas do ciclo de máquina. (1) Captar: a unidade de controle capta a instrução ADD da memória. (2) Decodificar: a unidade de controle decodifica a instrução ADD. Ela determina que a adição deve ser efetuada e fornece instruções para que o número seguinte, 76, seja colocado em um registrador para esse propósito. O total atual, 88, já está no acumulador. (3) Executar: o número seguinte, 76, é colocado no registrador. A ALU faz a adição, aumentando o valor para 164. (4) Armazenar: nesse caso, a ALU armazena o resultado no acumulador, em vez de na memória, porque ainda é necessário adicionar mais números a ela. O novo total, 164, substitui o total anterior, 88 e o processo continua.

Esse exemplo é ilustrado na Figura 3.19.



3FIGURA 19.19 - O ciclo de máquina em ação FONTE: Capron (2004)

Introdução à linguagem de montagem

Conforme falamos anteriormente, o computador é projetado para trabalhar com a linguagem de máquina, que utiliza mnemônicos para indicar a instrução ou operação que será executada.

O projeto de um processador é centrado no conjunto de instruções de máquina que se deseja que ele execute (na realidade, do conjunto de operações primitivas que ele poderá executar). Uma das mais fundamentais análises e decisões do projeto envolve o tamanho e a complexidade do conjunto de instruções. Quanto menor e mais simples o conjunto de instruções, mais rápido é o ciclo de tempo do processador (MONTEIRO, 2012).

Representação de instruções

O formato geral de uma instrução de máquina é:

C. Op.	Op.
Código de operação	Operando

Em que:

Código de operação: identifica a operação que será realizada. Temos um código para cada instrução que pode ser realizada pelo computador.

Operando: indica a localização do dado que será usado durante a operação. Esse campo pode conter mais de um operando.

Exemplos de instruções:

a. Com um operando:

- LDA Op., fornece o resultado Acc←(Op.)

O acumulador recebe o conteúdo do operando.

- ADD Op., fornece o resultado Acc←Acc + (Op.)

O acumulador recebe o seu valor anterior acrescido do conteúdo do operando.

b. Com dois operandos:

- MOVE Op.1, Op.2, fornece o resultado (Op.1)←(Op.2)

O operando 1 recebe o conteúdo do operando 2.

- SUB Op.1, Op.2, fornece o resultado (Op.1)←(Op.1) - (Op.2)

O operando 1 recebe o seu valor anterior subtraído do conteúdo do operando 2.

c. Com três operandos:

- ADD Op.1, Op.2, Op.3, fornece o resultado (Op. 3)←(Op.1) + (Op.2)

O operando 3 recebe a soma entre o conteúdo dos operando 1 e 2.

O campo operando não precisa, necessariamente, indicar o endereço do dado. O modo de endereçamento é que define como este dado será localizado, o que varia de instrução para instrução.

Os modos de endereçamento principais, segundo Tanenbaum (2007), são:

a. Endereçamento imediato

O modo mais simples de uma instrução especificar um operando é a parte da instrução referente ao endereço realmente conter o operando em si em vez de um endereço ou outra informação que descreva onde o operando está. Tal operando é denominado operando imediato porque ele é automaticamente buscado na memória, ao mesmo tempo que a própria

instrução; por conseguinte, ele está imediatamente disponível para uso. O endereçamento imediato tem a virtude de não exigir uma referência extra à memória para buscar o operando. A desvantagem é que somente uma constante pode ser fornecida desse modo.

Exemplo: a instrução MOV B, 22H, que move o valor hexadecimal 22 para o registrador B, utiliza o endereçamento imediato.

b. Endereçamento direto

Um método para especificar um operando na memória é dar seu endereço completo. Esse modo é denominado endereçamento direto. Assim como o endereçamento imediato, o endereçamento direto tem uso restrito: a instrução sempre acessará exatamente a mesma localização de memória. Portanto, embora o valor possa mudar, a localização não pode. Assim, o endereçamento direto só pode ser usado para acessar variáveis globais cujos endereços são conhecidos no momento da compilação. Não obstante, muitos programas têm variáveis globais, portanto, esse modo é amplamente usado.

Exemplo: a instrução ADD A, que soma o valor do endereço A ao acumulador utiliza o endereçamento direto.

c. Endereçamento de registrador

Endereçamento de registrador é conceitualmente o mesmo que endereçamento direto, mas especifica um registrador em vez de uma localização de memória. Como os registradores são tão importantes (devido ao acesso rápido e endereços curtos), esse modo de endereçamento é o mais comum na maioria dos computadores.

Exemplo: a instrução ADD A, B, soma o conteúdo do registrador B ao conteúdo do acumulador, armazenando o resultado no acumulador. Essa instrução utiliza o endereçamento de registrador.

d. Endereçamento indireto de registrador

Nesse modo, o operando que está sendo especificado vem da memória ou vai para a memória, mas seu endereço não está ligado à instrução, como no endereçamento direto. Em vez disso, o endereço está contido em um registrador. Quando um endereço é usado dessa maneira, ele é denominado ponteiro. Uma grande vantagem do endereçamento indireto de registrador é que ele pode referenciar memória sem pagar o preço de ter um endereço de memória completo na instrução.

Exemplo: a instrução ADD (A), que soma ao acumulador do conteúdo de memória apontado pelo endereço representado por A, utiliza o endereçamento indireto de registrador.

e. Endereçamento indexado.

Muitas vezes é útil poder referenciar palavras de memória cujo deslocamento em relação a um registrador é conhecido. Endereçamento indexado é o nome que se dá ao endereçamento de memória que fornece um registrador (explícito ou implícito) mais um deslocamento constante.

Indicação de leitura

Nome do livro: Turing e o computador em 90 minutos

Editora: Le Livros

Autor: Paul Strathern

ISBN: não tem

O livro apresenta um histórico bem completo da evolução das máquinas, desde a era a.C., quando ainda nem se pensava em computadores da maneira que temos hoje. Também contém uma biografia bem completa de Alan Turing, mostrando, também, curiosidades de sua vida até chegar no momento do desenvolvimento da máquina que viria a decifrar a Enigma, responsável pela codificação das mensagens alemãs. Finaliza com algumas datas significativas no desenvolvimento do computador e sugestões de leitura. É um livro curto e fácil de ler, ideal para quem quer saber um pouco mais sobre a história do computador.

O livro está disponível em: **Le Livro - Turing e o Computador em 90 minutos – Paul Strathern <https://lelivros.pro/book/baixar-livro-turing-e-o-computador-em-90-minutos-paul-strathern-em-pdf-mobi-e-epub/>** .

Indicação de leitura

Nome do livro: Introdução à Organização de Computadores

Editora: LTC – Livros Técnicos e Científicos

Autor: Mário Antonio Monteiro

ISBN: 9788521615439

O livro é bastante didático e contém todos os conteúdos estudados nesta unidade, como o funcionamento do computador, seu histórico, componentes e uma explicação bem completa sobre a Unidade Central de Processamento. Apresenta exemplos práticos, comparando a máquina com situações da vida real. É um livro muito indicado para fixar e conhecer mais profundamente os conceitos trabalhados.

UNIDADE IV

Organização de Sistemas Computacionais

Ana Cláudia de Oliveira Pedro Andrêo

Caro(a) aluno(a), estamos quase chegando ao final de nossos estudos de Arquitetura e Organização de Computadores.

Já trabalhamos com os sistemas numéricos, conhecemos o hardware do computador e seu funcionamento, levando em consideração seus principais componentes.

Agora, é hora de complementar esses conhecimentos, trabalhando com o subsistema de memória e as principais características e funcionamento de cada tipo diferente de memória que o compõe. Enfatizaremos o funcionamento e organização das memórias principal e cache.

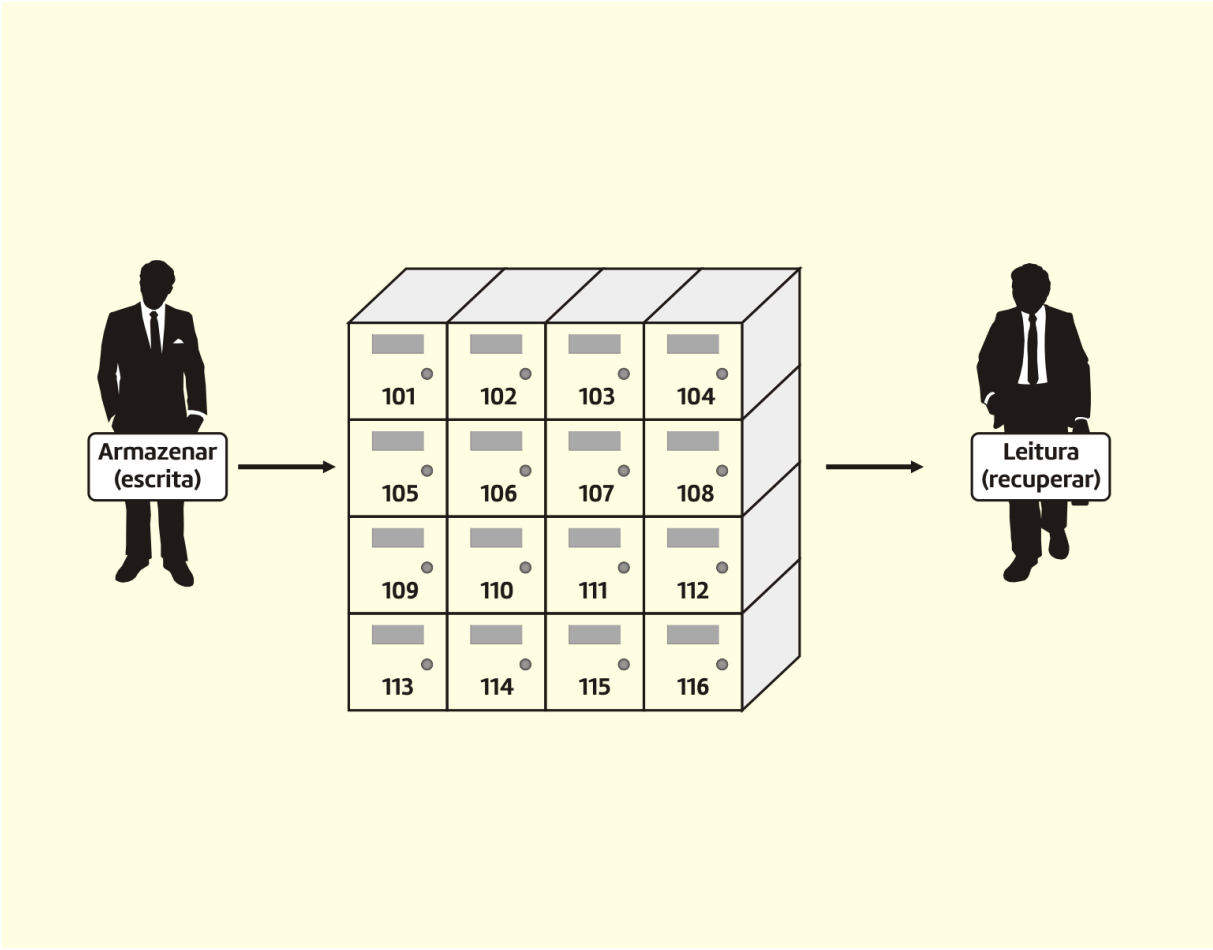
A seguir, falaremos sobre o subsistema de entrada e saída, explicando as transmissões serial e paralela, além do funcionamento das operações de entrada e saída e alguns dispositivos de armazenamento.

Finalizaremos abordando algumas tecnologias usadas para melhorar o desempenho do microprocessador, como as arquiteturas RISC, o pipeline e os sistemas com multiprocessadores.

Subsistemas de memória

Como já estudamos anteriormente, a memória é o componente responsável pelo armazenamento das informações (dados ou instruções) que são utilizadas no sistema computacional. Vamos complementar algumas informações agora.

Monteiro (2012) mostra de maneira simples o esquema conceitual de memória, utilizando como exemplo um depósito, que pode ser utilizado por uma ou mais entidades. Esse exemplo pode ser visualizado na Figura 4.1



4FIGURA 1.23 - Exemplo de um típico depósito que funciona de modo semelhante a uma memória FONTE: Monteiro (2012, p. 109).

Analisando com calma a figura 4.1, podemos perceber que existem outros itens que devemos analisar. Vamos a eles:

1. Cada caixa tem seu endereço.

O endereço é essencial para que se possa acessar qualquer informação na memória. Pense bem: como você poderia chegar à casa de um colega se não soubesse seu endereço?

Na memória, isso funciona da mesma maneira, mas, ao invés de chegar à casa de seu colega, pode-se acessar uma célula (ou bloco ou setor) de memória.

Segundo Monteiro (2012, p. 111), *"uma célula é, então, um grupo de bits tratado em conjunto pelo sistema, isto é, esse grupo é movido em bloco como se fosse um único elemento, sendo assim identificado para efeito de armazenamento e transferência, como uma unidade"*.

Cada uma das células, ou conjunto de bits, possui um endereço único, e a memória é organizada de acordo com esses endereços, que são colocados de forma sequencial.

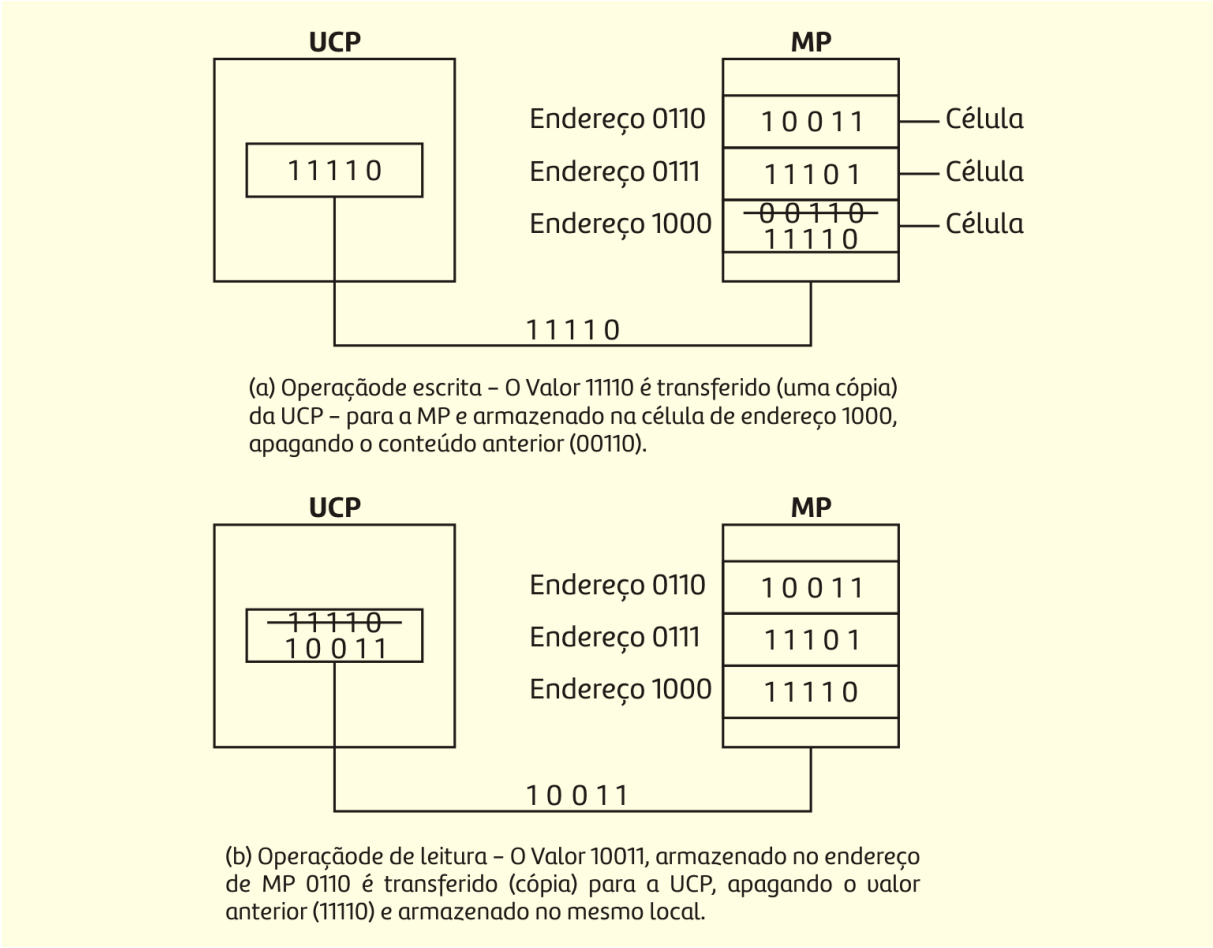
2. Temos duas pessoas acessando o depósito.

Isso representa as duas operações que podem ser realizadas com a memória: a de escrita e a de leitura.

A operação de escrita é caracterizada como o armazenamento de uma informação na memória. Essa operação também pode ser chamada de gravação (*write*). É importante lembrar que a escrita destrói o conteúdo anterior da memória, pois os bits novos serão armazenados no lugar dos originais.

A operação de leitura (*read*) é utilizada para recuperar uma informação na memória (*retrieve*). Essa informação pode ser utilizada para diversos usos: pela UCP, para realizar o ciclo de instrução; como dados para operações aritméticas, para envio a dispositivos de saída; entre vários outros usos no sistema computacional. A operação de leitura, diferentemente da de escrita, não é destrutiva, ou seja, os bits na memória permanecem os mesmos. O que acontece é apenas a realização de uma cópia desse conteúdo.

A figura 4.2 apresenta um exemplo das operações de escrita e leitura.

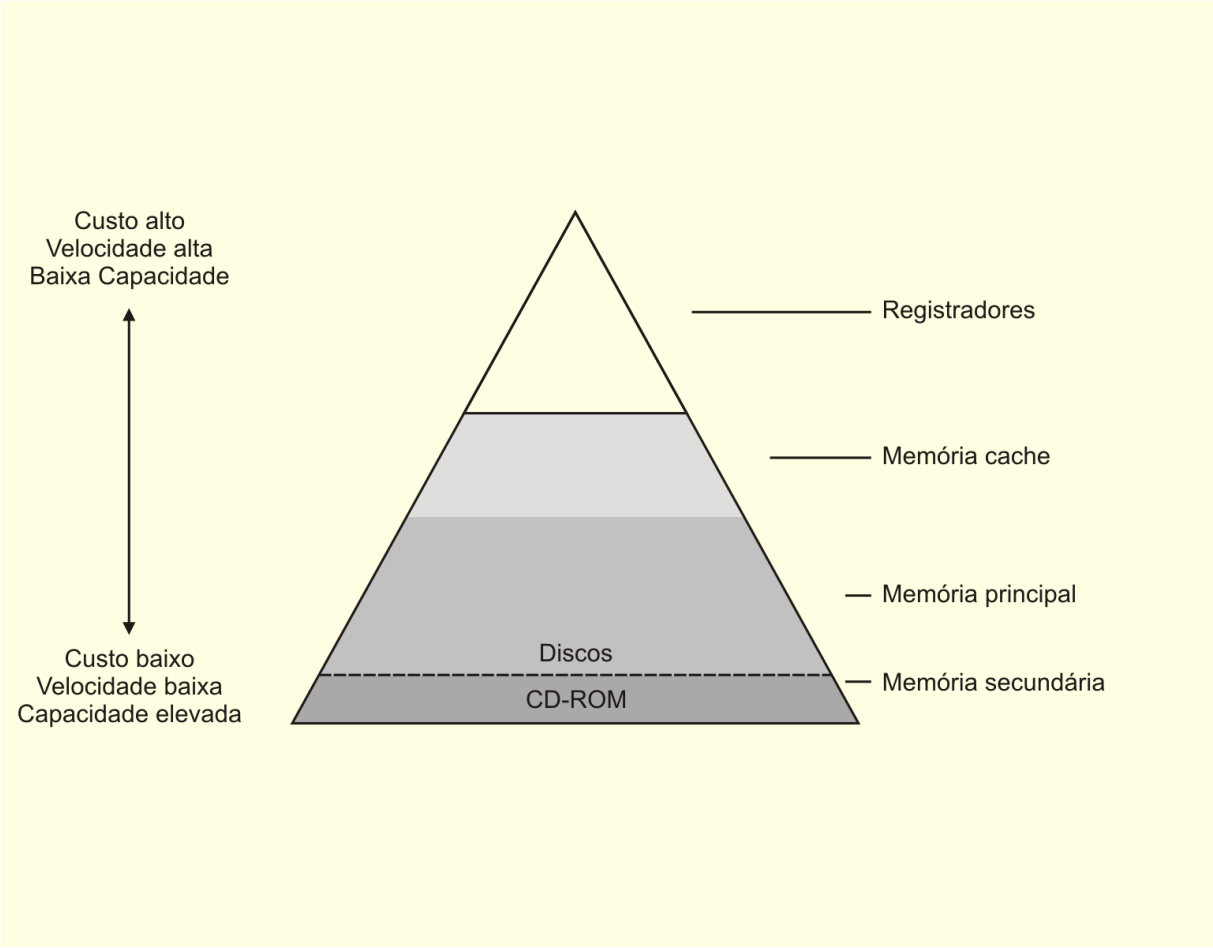


4FIGURA 2.23 - Operação de leitura e escrita na MP FONTE: Monteiro (2012, p. 112).

Mas não temos somente um tipo de memória atuando em um sistema computacional. Devido a diferentes características advindas da tecnologia de construção dessas memórias, como tempo de acesso (que é diretamente ligado à velocidade), capacidade, custo e volatilidade, cada tipo de memória é utilizado para uma função.

Esses diferentes tipos de memória que existem no sistema computacional constituem o subsistema de memória. É importante lembrar que essas memórias funcionam de forma conjunta, sendo interligadas.

Podemos organizar as memórias em uma pirâmide, que representa a hierarquia de memória, conforme mostra a figura 4.3



4FIGURA 3.23 - Hierarquia de memória FONTE: Monteiro (2012, p. 113).

Segundo Hennessy (2003), como uma memória rápida é dispendiosa, uma hierarquia de memória é organizada em vários níveis - cada um deles menor, mais rápido e caro por byte que o nível imediatamente abaixo. O objetivo é fornecer um sistema de memória com custo quase tão baixo quanto o nível de memória mais econômico e com velocidade quase tão grande quanto o nível mais rápido.

Weber (2008, p. 55) complementa: *"um sistema de computação só apresenta ganho na relação custo/desempenho se as memórias na hierarquia apresentarem velocidade, preço e tamanho significativamente diferentes"*.

Antes de falarmos sobre os níveis dessa hierarquia, é importante entender os parâmetros que levam a essa organização, que foram citados anteriormente.

Tempo de acesso

Para que seja realizada uma operação de leitura ou escrita na memória, é imprescindível fornecer o endereço desejado a ela. O tempo de acesso engloba desde o momento em que a UCP envia o endereço para o sistema de memória até o instante em que a informação solicitada foi transferida.

Dependendo da tecnologia utilizada para a fabricação da memória, temos diferentes tempos de acesso. Nas memórias eletrônicas, tipo RAM, ROM, esse tempo não varia de acordo com a localização das informações, ou seja, não importa a distância física entre os locais de acesso.

Já no caso das memórias construídas com dispositivos eletromecânicos, têm-se velocidades diferentes de acordo com a posição em que a informação está ou será armazenada. Isso acontece porque é necessário posicionar o mecanismo de escrita/leitura no local apropriado.

Como você é bem mais jovem do que eu, não deve se lembrar do toca discos ou "vitrola", como é chamado de forma caçoada. Apesar de que, com a moda "retrô", esses dispositivos podem ser encontrados novamente. Mas, se você não conhece, pergunte a seus pais ou avós. Nesses dispositivos, as músicas eram armazenadas em trilhas, divididas em faixas, conforme pode ser visto na figura 4.4, que mostra um toca discos e um disco de vinil.

4FIGURA 4.23 - Toca discos com disco de vinil FONTE: . Acesso: em 03 dez. 2016.

Para que se pudesse acessar cada uma das faixas, era necessário movimentar o braço que contém a cabeça de leitura, como ainda ocorre nos leitores de CDs/DVDs e discos rígidos (HDs).

Utilizando esse exemplo, podemos visualizar o tempo gasto para se ter acesso a esse tipo de memória.

¶ Fique por dentro

Velocidade x tempo de acesso

É importante perceber que os conceitos de velocidade e tempo de acesso são inversamente proporcionais. Quanto menor o tempo de acesso, maior a velocidade e, quanto mais tempo é gasto para trabalhar com uma informação (maior tempo de acesso), menor é a velocidade da memória.

Capacidade

A capacidade da memória representa a quantidade de informação que pode ser armazenada. Hoje utilizamos como medidas para essa capacidade o byte e seus múltiplos, conforme vimos anteriormente e está sendo mostrado novamente na figura 4.5.

Termo	Símbolo	Número aproximado de bytes
Quilobyte	K (ou KB)	um mil
Megabyte	MB	um milhão
Gigabyte	GB	um bilhão
Terabyte	TB	um trilhão
Petabyte	PB	um quatrilhão

4FIGURA 5.23 - Capacidades de armazenamento FONTE: Capron (2004, p. 101).

Dependendo do tipo de memória que estamos utilizando, usamos múltiplos diferentes. Por exemplo, quando nos referimos à memória principal, falamos em megabytes e, quando nos referimos à memória secundária, falamos em gigabytes e terabytes.

Tecnologia de fabricação

Temos diversas tecnologias para fabricação das memórias. As mais utilizadas atualmente são:

- Memórias de semicondutores: utilizadas para implementação dos registradores, memória principal e memória cache. São também chamadas de memórias eletrônicas, pois são construídas usando-se circuitos eletrônicos.
- Memórias de meio magnético: utilizadas nos discos rígidos (HDs) e nos antigos disquetes, que você provavelmente só conhece de nome. Sorte sua! Eles eram grandes, incômodos e extremamente inseguros. Nessas memórias, as informações

são armazenadas na forma de campos magnéticos.

- Memórias de meio ótico: utilizadas para armazenamento de informações nos CDs e DVDs. São usados feixes de luz para gravar os bits no dispositivo.

Uma nova tecnologia que está sendo muito utilizada atualmente para a fabricação de discos rígidos é o SSD (*Solid-State Drive*). Segundo Techtudo, 2016:

> [...] esse tipo de armazenamento de massa não possui nenhum disco. O dispositivo é todo formado por circuitos integrados e em seu interior não há partes móveis, o que o torna absolutamente silencioso, mais rápido e menos propenso a danos físicos do que o HD. O sistema de gravação e leitura, naturalmente, é diferente dos discos rígidos. A maior parte dos SSDs atuais armazena os dados em células de memória flash, que também são do tipo não volátil. É essa a versão usada em smartphones, tablets e computadores mais novos.

Custo

O custo de uma memória está diretamente ligado à sua tecnologia de fabricação, o que irá proporcionar maior ou menor velocidade.

É importante lembrar que, quando analisamos o custo de uma memória, devemos fazer o cálculo do valor por byte, e não levando em consideração o valor dos dispositivos.

Se fizermos isso, podemos chegar a conclusões errôneas, como afirmar que as memórias secundárias, como os discos rígidos, são mais caras que a memória principal (RAM). Veja bem a capacidade de armazenamento dos HDs e da memória RAM!!!

Volatilidade

A volatilidade diz respeito à capacidade de uma memória manter ou não suas informações na falta de alimentação elétrica. Memórias voláteis perdem essas informações e as não voláteis as guardam. Mais à frente, iremos especificar a classificação de cada tipo de memória.

Voltando, agora, à nossa hierarquia de memória. Na nossa pirâmide, podemos ver quatro tipos diferentes de memória compondo o subsistema de memória: os registradores, a memória principal, a memória cache e a memória secundária.

Registradores

Capron (2004, p. 96) afirma que *"registradores são áreas de armazenamento temporário de alta velocidade que servem a propósitos especiais e destinam-se a instruções ou dados [...] são áreas especiais de armazenamento temporário localizadas dentro da própria CPU que oferecem a alta velocidade como vantagem"*.

As instruções armazenadas são executadas, e os dados podem ser utilizados pela Unidade Aritmética e Lógica em suas operações.

Podemos dizer que quanto mais alta a posição da memória na pirâmide de hierarquia, citada anteriormente, mais próxima ela está do processador.

Como os registradores trabalham na mesma velocidade do processador, ou seja, possuem o menor tempo de acesso, estão localizados no topo da pirâmide. Também são o tipo de memória com menor capacidade, devido a seu alto custo, ligado à tecnologia de fabricação, que é a mesma da UCP.

Quanto à volatilidade, é uma memória volátil, pois perde seu conteúdo na falta de alimentação elétrica.

Memória Cache

Em nossa pirâmide, a memória cache se encontra logo abaixo dos registradores. Assim, ela possui um tempo de acesso maior (velocidade menor) do que esses, mas menor do que a memória principal. Sua capacidade também aumenta, sendo inferior, também, à da memória principal, devido a seu custo ainda alto.

Elas são fabricadas com tecnologia semelhante à da UCP e são voláteis.

Memória Principal

A memória principal localiza-se entre a memória cache e a memória secundária, o que significa que possui uma velocidade maior do que a primeira e menor do que a segunda. Quanto à capacidade, essa é maior do que a encontrada na memória cache, mas ainda é muito inferior à da memória secundária.

Seu custo é intermediário, sendo que sua tecnologia de fabricação possui elementos dinâmicos, como discutido em unidades anteriores.

É uma memória volátil também, pelo menos em sua grande parte.

Memória Secundária

A memória secundária localiza-se na base da pirâmide, portanto é a memória que possui mais capacidade de armazenamento, mas menor velocidade. Seu custo é relativamente baixo, comparado com as memórias anteriores, e é uma memória não volátil, sendo capaz de guardar as informações mesmo quando desligamos o computador, o que é o principal fator para sua presença no subsistema de memória.

As memórias principal, cache e secundária serão vistas com mais detalhes no decorrer de nossa unidade.

Memória Principal

Na pirâmide que caracteriza os níveis de hierarquia de memória, a memória principal se encontra em um nível intermediário, logo abaixo da memória cache. Ela é volátil e, em relação à capacidade, possui uma maior do que a cache, com um custo menor. Mas, em contrapartida, sua velocidade é bem menor.

Em uma visão mais simples, como a de usuários leigos, Capron (2004, p. 96) define que

> a memória também é conhecida por armazenamento primário, memória primária, armazenamento principal e memória principal: o pessoal da área de informática usa esses termos de forma alternada. Os fabricantes geralmente usam o termo RAM, que é a sigla de *random-access memory* (memória de acesso aleatório).

Uma definição mais precisa, segundo o mesmo autor, é que

> memória é a parte do computador que mantém dados e instruções a serem processados. Embora esteja estreitamente relacionada com a UCP, a memória encontra-se em um lugar distinto. A memória armazena instruções de programa ou dados apenas enquanto o programa ao qual eles pertencem estiver em execução .
>
> (CAPRON, 2004, p. 96)

Justamente por armazenar temporariamente instruções e dados a serem utilizados pela UCP, essa memória é também chamada de memória de trabalho.

Na realidade, o uso do nome RAM como sinônimo de memória principal não é totalmente correto. Como a RAM é volátil, não é possível armazenar as instruções iniciais, que fazem o computador funcionar.

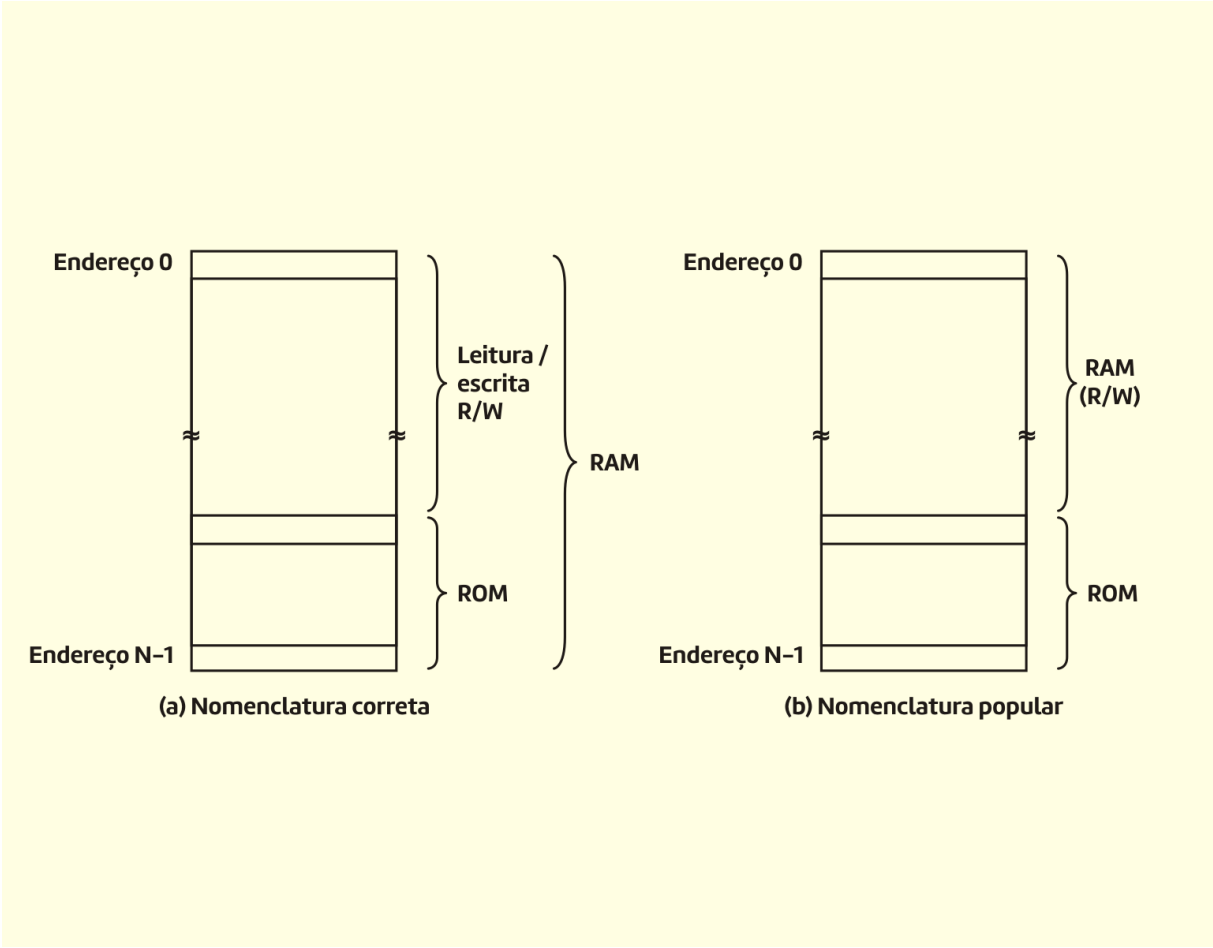
Esses programas são armazenados na memória ROM (que já vimos anteriormente), que não é volátil e funciona somente para leitura, e são chamados de *firmware*. Os principais são a BIOS e o SETUP.

A BIOS (*Basic Input Output System* – Sistema Básico de Entrada/Saída), como o próprio nome diz, é o programa responsável pelo reconhecimento dos dispositivos básicos de entrada e saída.

Já o SETUP, segundo Techtudo (2015, *on-line*),

> é um sistema operacional bem rudimentar, responsável por colocar o computador em funcionamento assim que você liga a máquina. É esse sistema que confere os dispositivos instalados para saber se há memória no PC, se o processador está sendo mantido em temperaturas seguras, se os discos rígidos estão funcionando e prontos para carregar o sistema operacional.

A figura 4.6 mostra a distribuição das memórias RAM e ROM: como são usadas diariamente e a forma correta de representá-las.

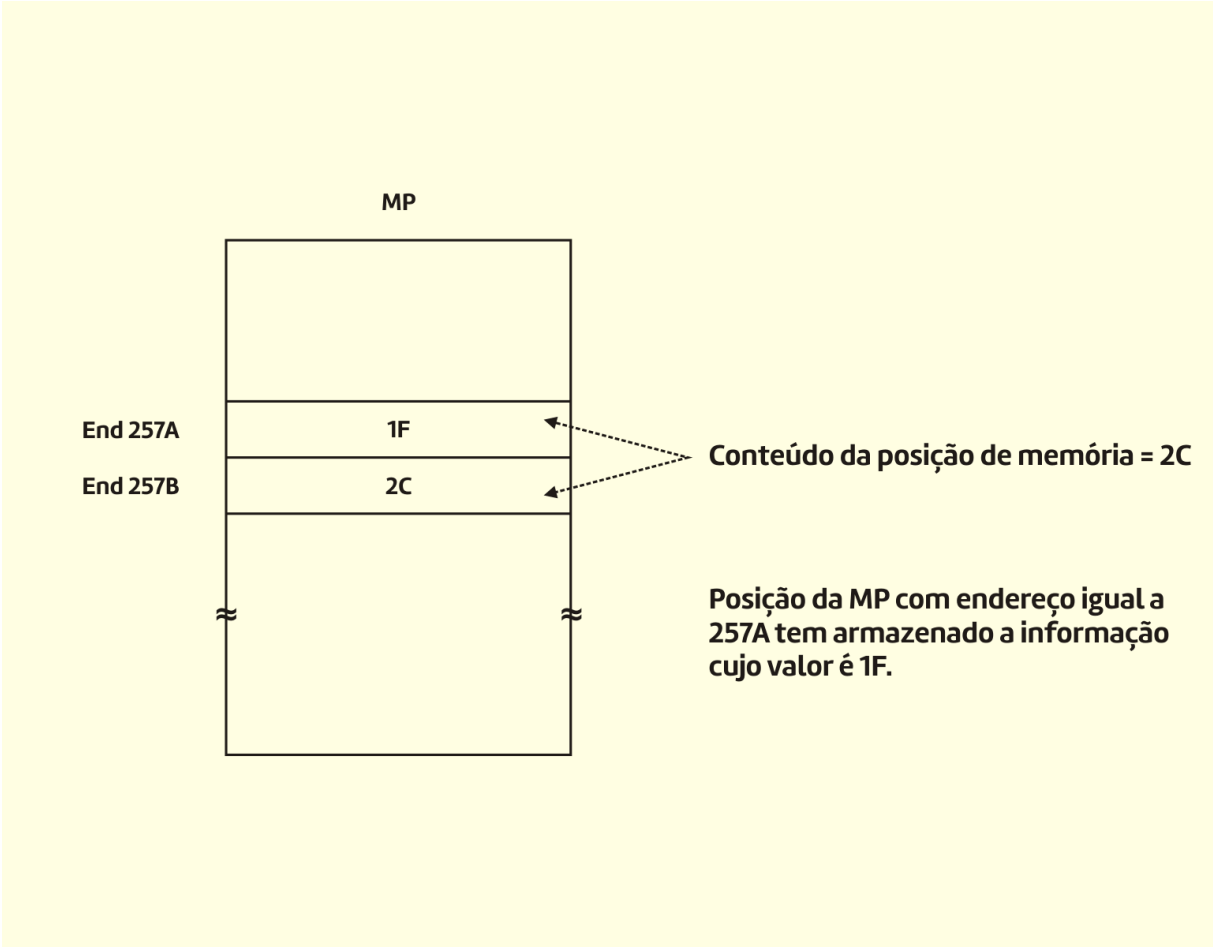


4FIGURA 6.23 - Conção da memória principal (MP) de um microcomputador do tipo PC FONTE: Monteiro (2012, p. 138).

Monteiro (2012, p. 122) define

> a memória principal de qualquer sistema de computação é organizada como um conjunto de N células sequencialmente dispostas a partir da célula de endereço igual a 0 até a última, de endereço igual a N – 1. Cada célula é construída para armazenar um grupo de M bits, que representa a informação propriamente dita e que é manipulado em conjunto (como se fosse uma única unidade) em uma operação de leitura ou escrita.

Quando se estuda a organização de uma memória, é muito importante diferenciar três conceitos: endereço, conteúdo e posição da MP. Já tratamos disso, mas a figura 4.7 apresenta essa diferenciação de forma bem clara, complementando nossos conhecimentos.



4FIGURA 7.23 - Significado dos valores de endereço e conteúdo na MP FONTE: Monteiro (2012, p. 123).

Além desses conceitos, o termo *palavra* é muito importante para compreensão da organização da memória principal.

Palavra

De acordo com Monteiro (2012, p. 122), *palavra "é a unidade de informação do sistema UDP/MP que deve representar o valor de um número (um dado) ou uma instrução de máquina".*

Levando em consideração essa definição, conclui-se que o ideal é que a memória principal fosse composta por células com o tamanho da palavra. Mas, isso não é possível, pois cada fabricante possui projetos diferentes e, como já falamos, quanto maior a quantidade de bits que o computador utiliza de uma vez, maior é o seu desempenho.

Assim, padronizou-se o tamanho de cada célula de memória em 8 bits, e a UCP acessa várias delas sequencialmente.

No início desta unidade, falamos sobre as operações de leitura e escrita que podem ser realizadas com a memória. Vamos detalhá-las um pouco mais, mostrando o papel dos registradores existentes no sistema computacional.

Relembrando nossa unidade anterior:

Os principais registradores presentes na UCP são:

a. RDM (registrador de dados da memória): contém o dado lido da memória ou que deverá ser escrito nela. Em inglês, é chamado de *Memory Buffer Register* (MBR).

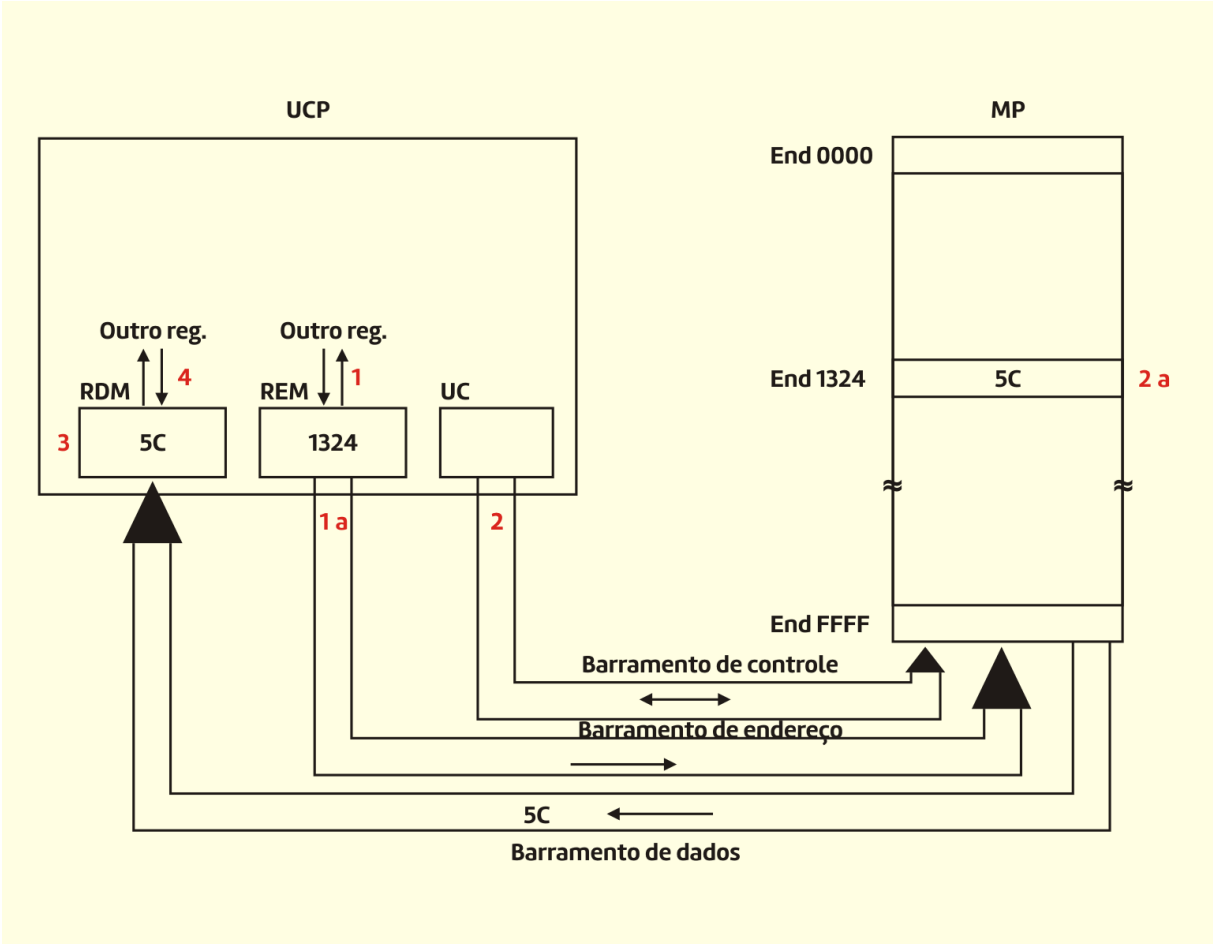
b. REM (registrador de endereços da memória): contém o endereço do dado que deverá ser lido ou escrito na memória. Em inglês, é chamado de *Memory Address Register* (MAR).

Também participam das operações de leitura/gravação os seguintes barramentos:

1. Barramento de dados: conecta o RDM à memória principal, sendo bidirecional, ou seja, os sinais podem transitar nos dois sentidos. Esse barramento transfere as informações e os dados.
2. Barramento de endereços: conecta o REM à memória, fornecendo o endereço da informação que será escrita/lida na memória. É unidirecional, pois somente a UCP envia o endereço desejado. A memória apenas recebe. Esse barramento é capaz de transferir o mesmo número de bits que os que representam o endereço.
3. Barramento de controle: conecta a unidade de controle (que faz parte da UCP) à memória. É o responsável pela passagem de sinais de controle tanto da UCP para a MP (por exemplo, os sinais que indicam se a operação é de leitura ou de escrita) quanto da MP para a UCP (às vezes, a memória precisa enviar um sinal solicitando que a UCP aguarde a finalização de uma determinada transferência). Portanto, é um barramento bidirecional.

Monteiro (2012, p. 127) mostra um exemplo de operação de leitura de um dado que está armazenado na memória principal no endereço 1324 (cujo valor é 5C).

A figura 4.8 mostra os passos necessários para que essa operação ocorra.



4FIGURA 8.23 - Exemplo de operação de leitura FONTE: Adaptada de Monteiro (2012, p. 128).

Em linguagem de transferência de registradores (LTR), temos, para os números indicados na figura 4.8:

1. REM←de outro registrador da UCP.
2. O endereço é colocado no barramento de endereços.

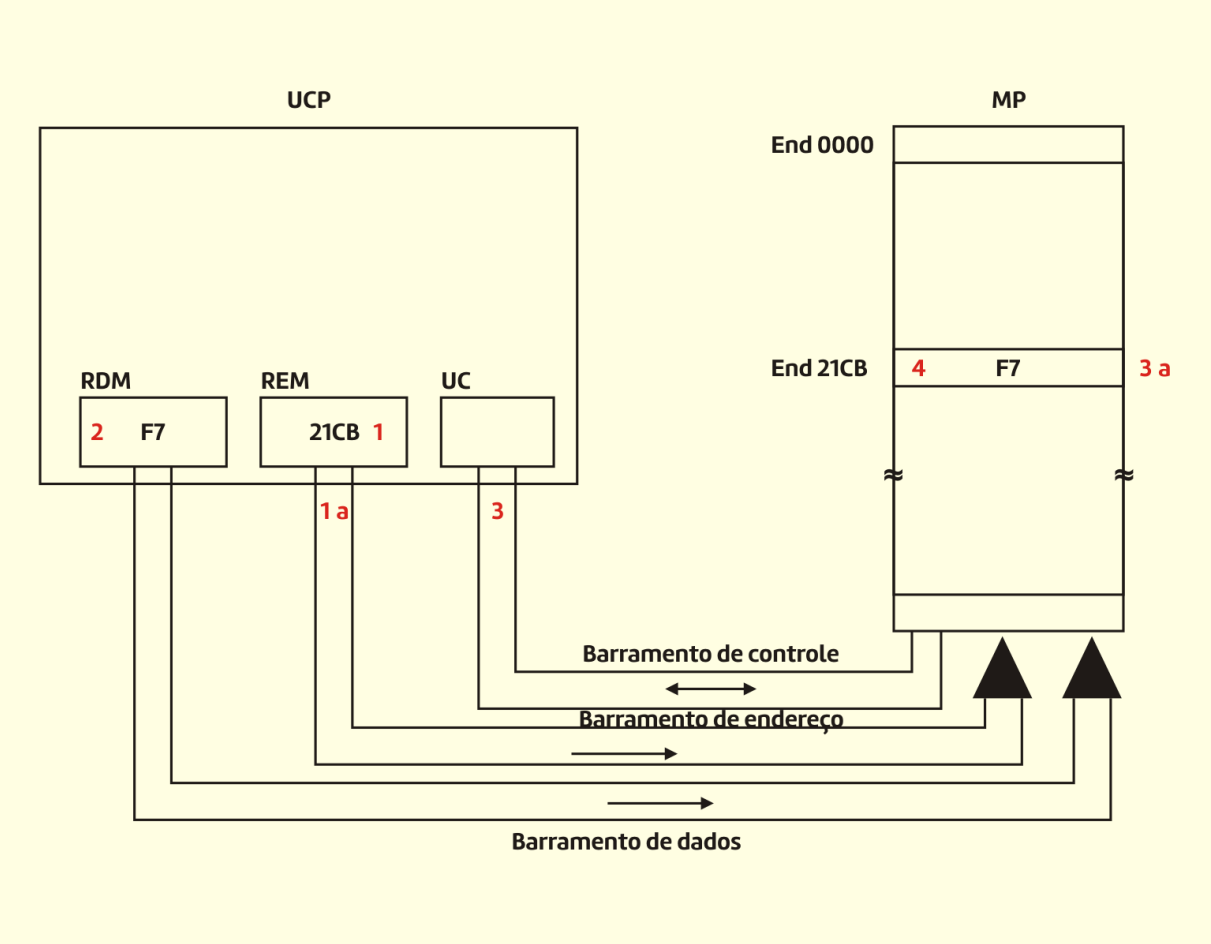
1. Sinal de leitura é colocado no barramento de controle.
2. Decodificação do endereço e localização da célula.

1. RDM←MP (REM)
2. Para outro registrador da UCP←RDM.

Os parênteses no passo 3 indicam que quem será transferido para o registrador de endereços da memória é o conteúdo do endereço enviado através do barramento de endereços.

Para a operação de escrita, os passos são praticamente os mesmos, mas primeiro precisamos carregar o dado a ser gravado na memória no registrador de dados antes de enviar o sinal de escrita (write) à memória. O sentido em que o dado transita também é inverso, indo da UCP para a memória.

A figura 4.9 mostra um exemplo de operação de escrita.



4FIGURA 9.23 - Exemplo de operação de escrita FONTE: Adaptada de Monteiro (2012, p. 129).

Assim como no caso da operação de leitura, para a escrita, podemos ter, de acordo com os números indicados na figura 4.9:

1. (REM)←(outro registrador).
2. O endereço é colocado no barramento de endereços.

1. (RDM)←(outro registrador).
2. Sinal de escrita é colocado no barramento de controle.
3. Decodificação do endereço e localização da célula.

1. (MP (REM))←(RDM).

Capacidade da memória principal

Segundo Monteiro (2012, p. 130),

capacidade de memória refere-se genericamente à quantidade de informação que nela pode ser armazenada em um instante de tempo. Tratando-se de um computador, cuja unidade básica de representação de informação é o bit, pode-se imaginar este elemento como unidade de medida de capacidade.

Para podermos calcular a capacidade de uma memória RAM, precisamos saber quantas células ela possui e qual é a quantidade de bits que cada uma dessas células pode armazenar. Comercialmente, padronizou-se a quantidade de bits em 8, ou seja, 1 byte.

O número de células que ela possui está diretamente ligado ao endereçamento de cada posição de memória ou ao seu tamanho de endereço. Para uma memória que possui E bits, para representar o endereço, temos:

$$N \text{ (número de endereços)} = 2^E$$

Ou seja, se tivermos 9 bits para endereçamento, o número de células que podemos acessar é:

$$N = 2^E = 2^9 = 512 \text{ células, cada um com seu endereço correspondente.}$$

Cada uma das células possui um número máximo de bits que pode armazenar. Falamos em padronização de 8 bits, mas essa célula, dependendo do projeto da memória, poderia armazenar uma quantidade M de bits.

Assim, o total de informação que uma memória pode armazenar é:

$$T = N \times M, \text{ ou seja, número total de endereços (ou total de células) x a capacidade de cada célula}$$

Exemplo: Calcule a capacidade de uma memória hipotética que possui 32 K células, cada uma delas com 8 bits. Qual será o tamanho do REM para que se possa acessar essa memória?

A primeira parte da pergunta solicita a capacidade total de memória. Assim, temos:

$$T = N \times M = 32 \text{ K} \times 8 = 256 \text{ Kbits (atenção nesse ponto! Como estamos trabalhando com M em bits, a resposta também será em bits).}$$

Vamos agora à segunda parte da pergunta: o tamanho do registrador de endereços da memória está diretamente ligado à quantidade de células que ele pode endereçar. Portanto:

$$N = 2^E$$

$$32k = 32 \times 10^3 = 2^{15} = 2^{\text{números de bits}}$$

Assim, o número de bits é 15, ou seja, o REM deverá possuir o tamanho de 15 bits.

¶ Reflita

"Na realidade, sabemos que não é possível armazenar 2 (dois) ou mais valores em uma célula de memória, ou seja, em um único endereço somente um valor (um dado) poderá ser localizado e identificado. Isto porque se fossem

> um único endereço somente um valor (um dado) poderá ser localizado e identificado. Isto porque se fossem armazenados dois valores em um endereço (célula), o sistema não saberia identificar qual dos dois seria o desejado em uma certa operação de leitura ou escrita (precisar-se-ia, então, de uma identificação a mais – um endereço dentro de um endereço), com todos os óbvios inconvenientes" (MONTEIRO, 2012, p. 130).

Memória Cache

A memória cache está localizada, em nossa pirâmide, entre os registradores e a memória principal. Curiosamente, essa também é a sua localização física. Atualmente, ela está dentro do processador.

É uma memória muito rápida, que trabalha praticamente na mesma velocidade da UCP, mas tem capacidade pequena.

Seu uso justifica-se por dois pontos: a grande diferença de velocidade entre a UCP e a MP e o princípio de localidade.

Já vimos que, para realizar suas operações, o processador deve acessar a memória pelo menos uma vez a cada instrução, quando esta é trazida da memória. Mas, o acesso à memória é extremamente lento, se comparado à velocidade em que o processador trabalha. Isso acaba causando um atraso grande no processamento, em que a UCP fica ociosa.

O outro ponto é o princípio de localidade. As instruções dos programas ficam armazenadas na memória de forma sequencial, ordenada. Assim, ao se acessar uma parte da memória, têm-se praticamente todas as instruções de um programa (lógico que isso depende do tamanho dele). Assim, por que não colocar essas instruções em uma memória mais rápida?

É justamente essa a função da memória cache. Ela serve como uma "ponte" entre a memória principal e a UCP. Sempre que o processador precisa de alguma informação, ele procura primeiramente na cache, que permite um acesso bem mais rápido.

Para Capron (2004, p. 110.),

> no jargão da informática, cache é uma área de armazenamento temporário projetada para acelerar a transferência de dados dentro do computador [...]. A memória cache, um tipo de memória muito rápido, é um bloco relativamente pequeno de memória desenvolvido com o propósito específico de acelerar a transferência de dados e instruções do software.

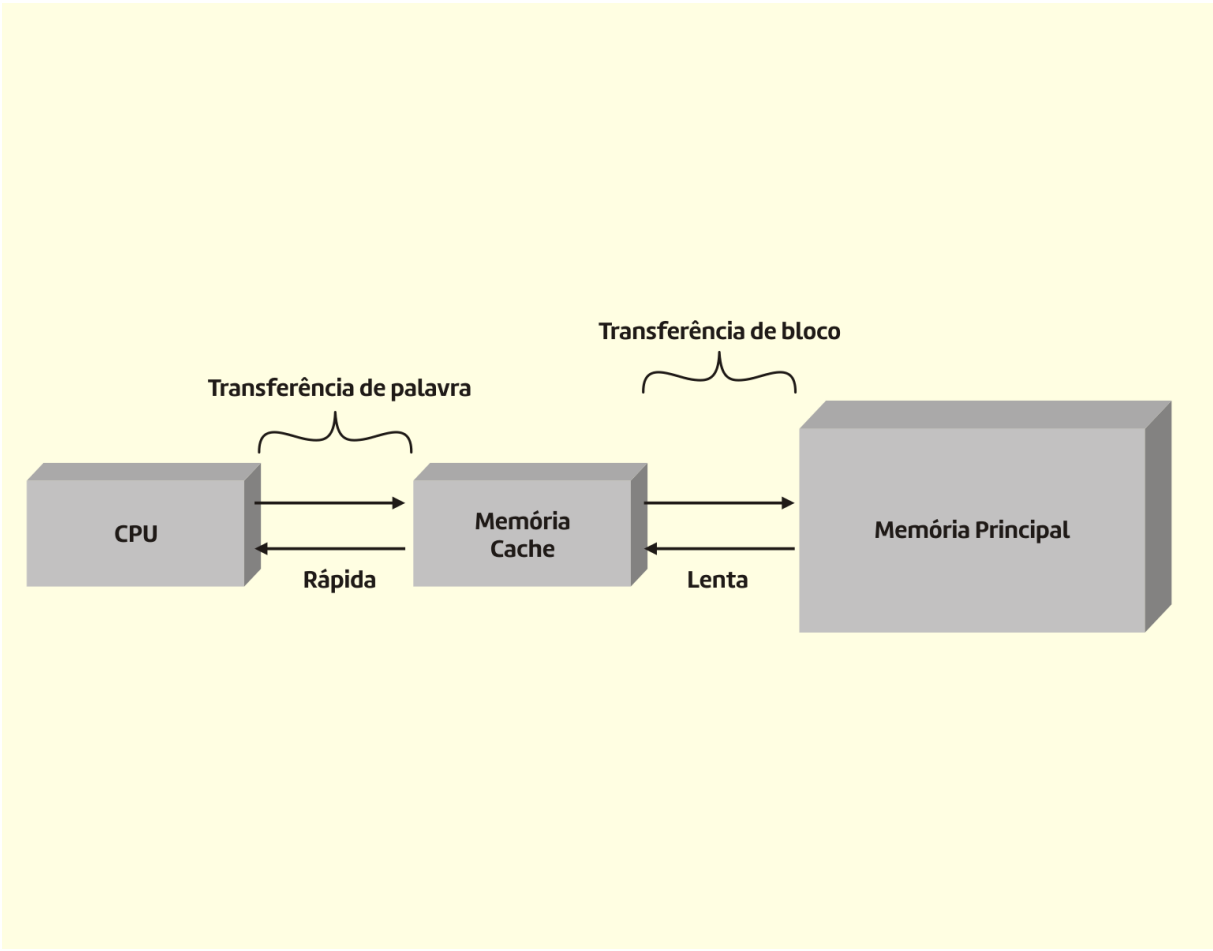
Você nunca ouviu ou falou: "o programa carregou rápido porque já está na cache"?

Isso ocorre porque essas instruções foram usadas mais recentemente e já estão carregadas nessa memória. Se o programa ainda não tivesse sido utilizado após você ter ligado o computador (lembre-se que a cache é volátil, perde as informações quando o computador é desligado), ou tivesse usado vários outros programas após ele, esse processamento não seria tão rápido, já que as instruções teriam que ser buscadas na memória principal.

Mas, você pode se perguntar: se o acesso à memória principal é tão lento, porque esse problema não aparece quando há a transferência de informações entre a memória principal e cache?

Isso novamente nos leva ao princípio da localidade. A transferência entre a memória cache e a principal se dá por blocos de informação e não por palavras, como é a maneira que a UCP trabalha. Assim, quando é realizado o acesso à memória cache, transfere-se uma quantidade razoável de informação. Essa quantidade depende, é claro, da capacidade da memória cache.

A figura 4.10 ilustra o subsistema de memória com a inserção da memória cache.



4FIGURA 10.23 - Cache e memória principal FONTE: Stallings (2010, p. 96).

Isso leva a um novo questionamento: se eu aumentar cada vez mais o tamanho da memória cache, isso não aumentaria o desempenho? Então, por que não usar somente ela?

Lembre-se de nosso outro parâmetro de comparação entre as memórias. Como a memória cache é fabricada com tecnologia semelhante à do microprocessador, seu custo é muito elevado. Além disso, como ela já está embutida na UCP, temos uma limitação de tamanho.

Portanto, os fabricantes procuram uma melhor relação custo-benefício. Uma quantidade de memória cache que não aumente muito o preço do sistema mas que, ao mesmo tempo, possa possibilitar um bom desempenho.

Pensando de forma simplificada, o funcionamento do sistema com a inserção da memória cache funciona da seguinte maneira: quando o processador precisa de uma informação, ela busca primeiro na cache.

Para Henessy (2014, p. 325), *"se a cache informa um acerto, a máquina continua seu processamento, usando a informação requisitada como se nada tivesse ocorrido, ou seja, como se a informação tivesse sido obtida na memória principal".*

O problema é quando essa informação não está lá. Aí, diz-se que ocorreu uma falta ou falha. As faltas são mais difíceis de serem trabalhadas. Primeiramente, o processamento deve ser "congelado", ou seja, o processador parado e todo o conteúdo de seus registradores armazenado.

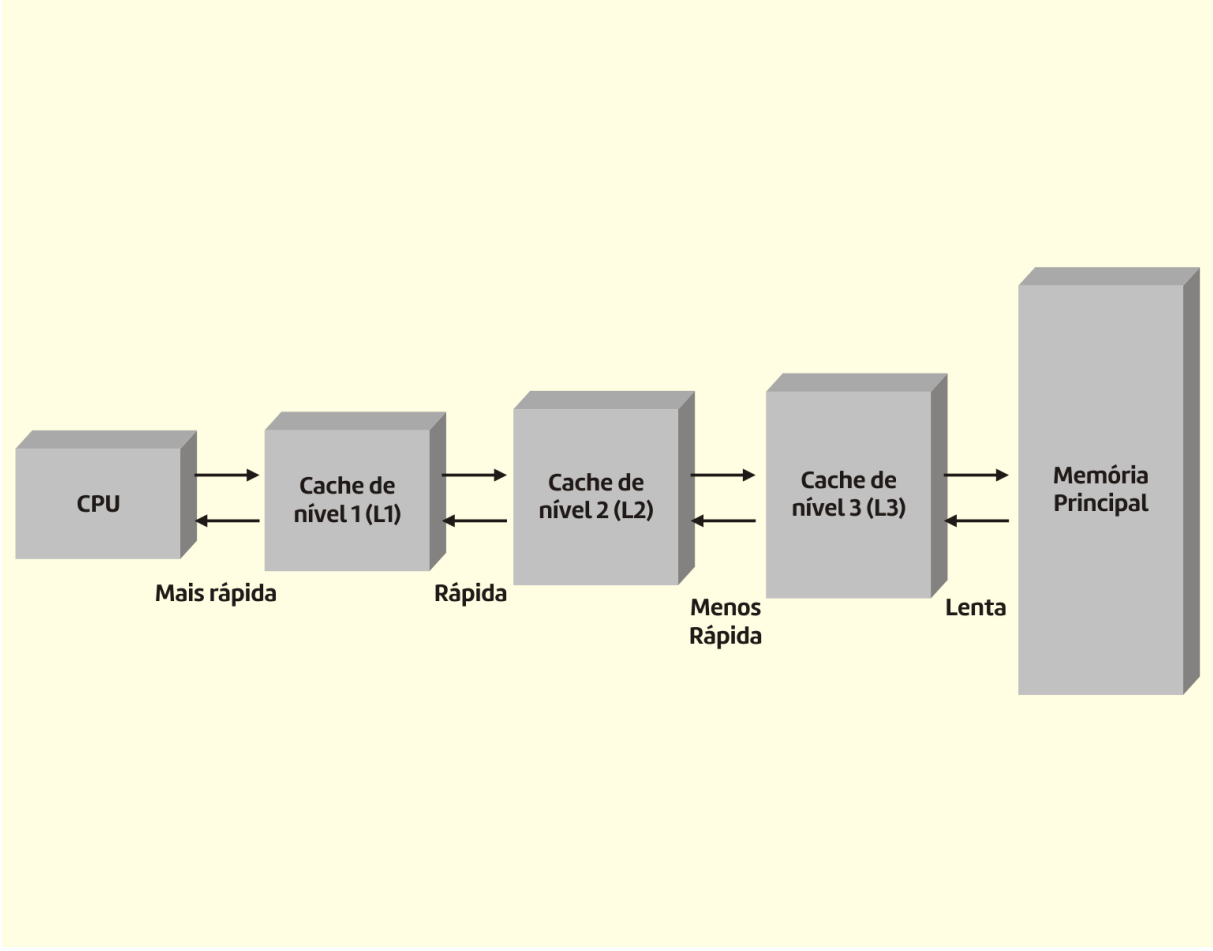
Após isso, um outro controlador busca na memória as novas informações. Quando isso é finalizado, todo o processamento é "descongelado" e volta-se a trabalhar como se nada tivesse acontecido.

Na realidade, não temos somente uma cache no sistema, e sim três. A cache L1 é a mais rápida e menor, seguida pela L2, que possui velocidade e tamanho intermediários e finalizando com a L3, que tem maior capacidade, mas menor velocidade.

Para Stallinggs (2010, p. 95),

> a CPU recebe instruções da cache L1 a velocidades muito rápidas. Quando a CPU precisa de uma instrução que não está na cache L1 (falha de cache), ela vai para a cache L2 procurá-la. Isto leva mais tempo. Se ela não encontra o dado na cache L2, ela precisa ir para a cache L3 ou mesmo à DRAM para recarregar a cache. Isso leva muito tempo.

A figura 4.11 mostra a relação entre a cache e a memória principal, agora com os níveis de cache.

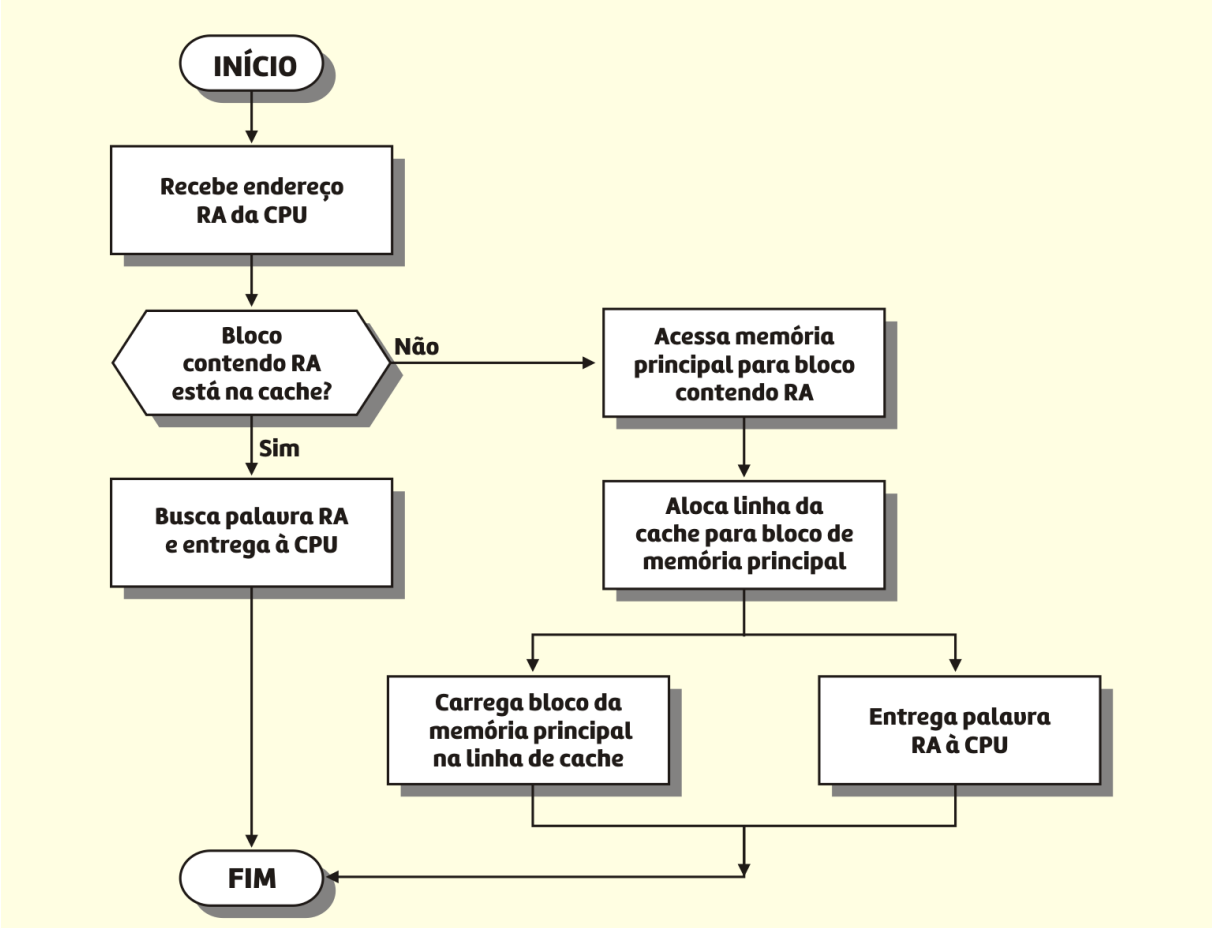


4FIGURA 11.23 - Organização de cache em três níveis FONTE: Stallings (2010, p. 96).

Tocci (2011, p. 733) compara a utilização da memória cache com a maneira que um cozinheiro trabalha:

> os ingredientes estão próximos à mão. Se algo mais é necessário, ele vai até a despensa, onde o estoque local está armazenado. Se o ingrediente não está ali, ele tem que pedir para um fornecedor de fora, e por aí adiante. Estoques são mantidos em cada nível para aumentar a eficiência enquanto gerencia-se o custo. Estas são as mesmas razões pelas quais utilizamos a memória cache em um computador.

A operação de leitura na cache é mostrada no fluxograma apresentado na figura 4.12.



4FIGURA 12.23 - Operação de leitura da cache FONTE: Stallings (2010, p. 97).

¶ Reflita

"Para haver realmente algum aumento de desempenho de um sistema de computação, com a inclusão da cache, é

Para haver realmente algum aumento de desempenho de um sistema de computação, com a inclusão da cache, é necessário que haja muito mais acertos do que faltas. Isto é, a memória cache somente é produtiva (aumento de desempenho do sistema) se a UCP puder encontrar uma quantidade apreciável de palavras na cache, suficientemente grande para sobrepujar as eventuais perdas de tempo com faltas que redundam em transferência de um bloco de palavras da MP para a cache" (MONTEIRO, 2012, p. 145).

¶ Fique por dentro

Utilização da memória cache

A utilização da memória cache se aplica a programas estruturados, que não envolvam desvios constantes. Lembra-se de sua professora de Algoritmos falando para você evitar programas que utilizam o GO TO? Isso porque, como a cache armazena blocos de informação sequencial, se tivermos muitos desvios, ela vai ter que acessar muito a memória principal para a busca de novas instruções, o que acaba inviabilizando o seu uso.

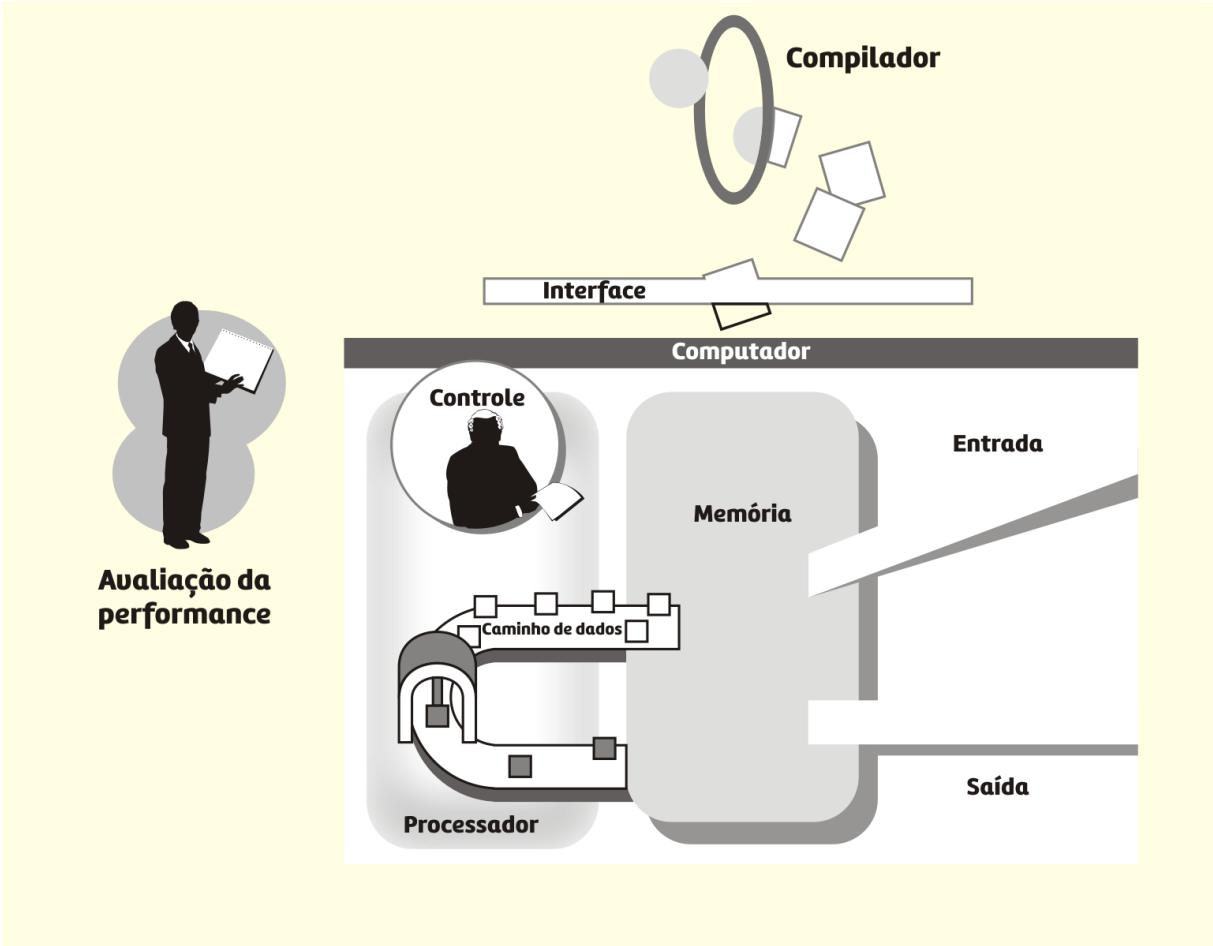
Subsistema de entrada e saída

Assim como temos o subsistema de memória em um computador, temos também o subsistema de entrada e saída, chamado também de E/S ou I/O (*input/output*).

Quando falamos sobre a arquitetura de Von Neumann, vimos que um computador era composto por:

- UCP, com sua unidade aritmética e lógica, unidade de controle, registradores (inclusive o acumulador);
- Memória principal;
- Equipamento de entrada e saída, controlado pela unidade de controle.

A figura 4.13 mostra os componentes clássicos de um computador.



4FIGURA 13.23 - Componentes de um computador FONTE: Hennessy (2014, p. 374).

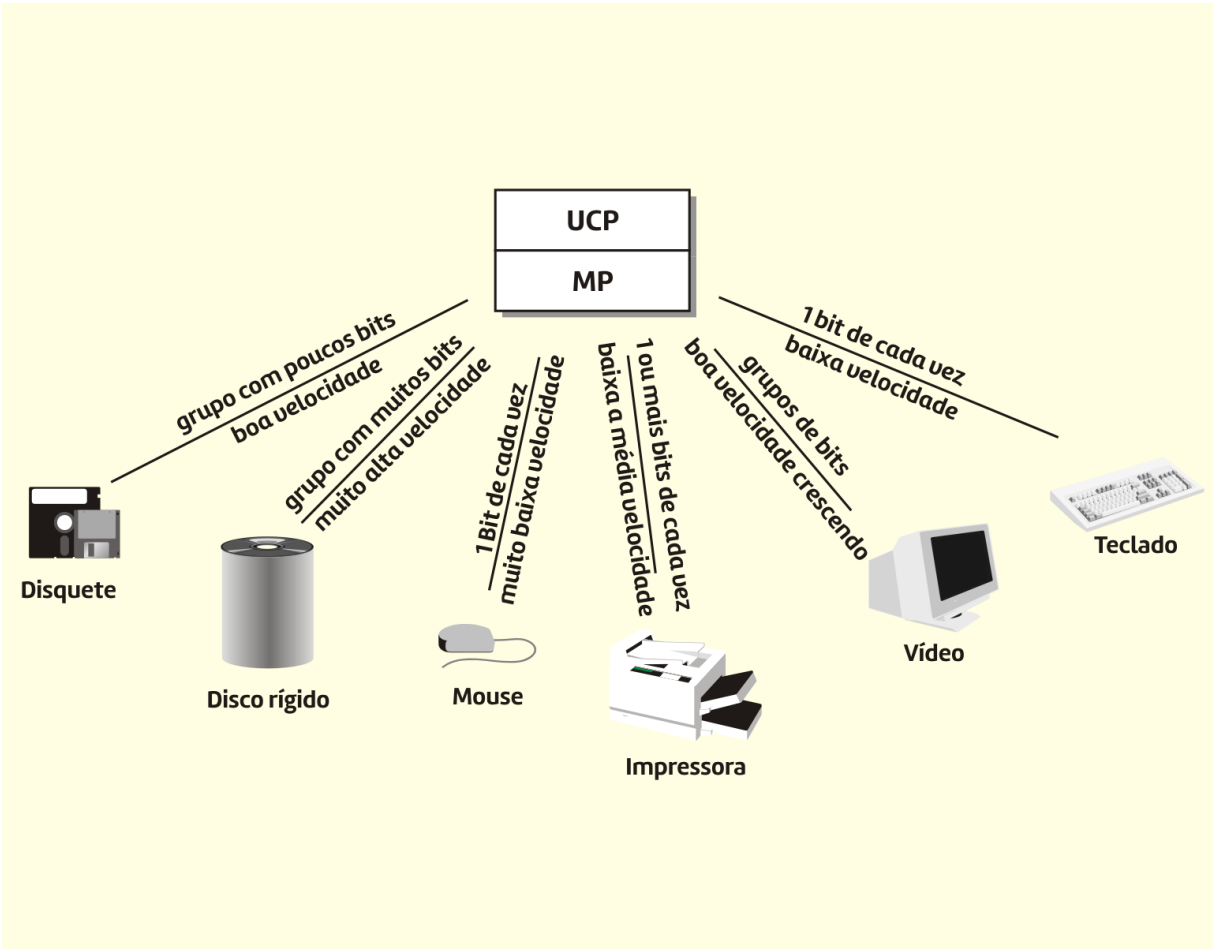
O subsistema de entrada e saída é composto pelos dispositivos (ou periféricos) de entrada, de saída e, ainda, de entrada/saída, além das interfaces de E/S, que também podem ser chamadas de controlador (de disco, de vídeo etc.), processador de periférico, canal, adaptador, "placa", entre outros.

Segundo Stallings (2010), os dispositivos de entrada e saída podem ser divididos em três categorias:

- Legíveis ao ser humano: adequados para comunicação com usuários de computadores;
- Legíveis à máquina: adequados para a comunicação com equipamentos, máquinas e internamente ao computador;
- Comunicação: adequados para a comunicação com dispositivos remotos.

As interfaces são extremamente importantes na interligação que é realizada entre a UCP e o dispositivo. Isso porque os equipamentos trabalham de forma bem particular, havendo diferenças na velocidade entre eles, mesmo na maneira de transmitir a informação. Por isso, há necessidade de um dispositivo para compatibilizar essas diferenças. Uma mesma interface pode controlar mais de um periférico.

Na figura 4.14, podemos ver as diferentes características de transmissão dos periféricos.



4FIGURA 14.23 - Exemplo de comunicação direta UCP/MP e periféricos, indicando-se as diferentes características de transmissão de cada um FONTE: Monteiro (2012, p. 375).

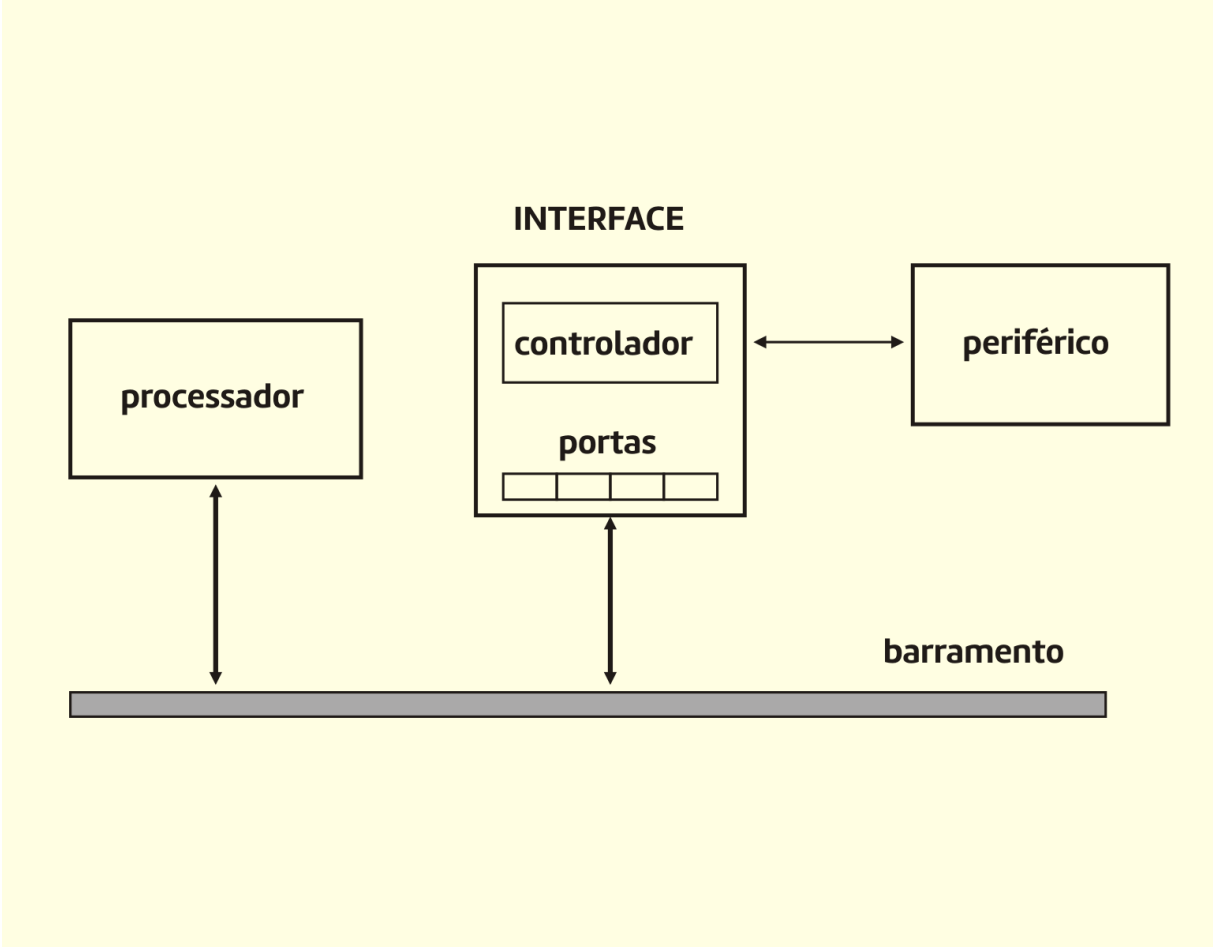
Segundo Monteiro (2012, p. 377),

> uma interface ou controlador de E/S é, em geral, responsável pelas seguintes tarefas:
> a) controlar e sincronizar o fluxo de dados entre a UCP/MP e o periférico;
> b) realizar a comunicação com a UCP, inclusive interpretando suas instruções ou sinais de controle para o acesso físico ao periférico;
> c) servir de memória auxiliar para o trânsito de informações entre os componentes (*buffer* de dados);
> d) realizar algum tipo de detecção e correção de erros durante as transmissões.

Assim, podemos concluir que a comunicação entre uma interface de E/S com o processador e os dispositivos periféricos é realizado da seguinte maneira:

1. O processador interroga o módulo de E/S para verificar o estado do dispositivo conectado.
2. O módulo de E/S retorna o estado do dispositivo.
3. Se o dispositivo estiver operacional e pronto para transmitir, o processador solicita a transferência de dados por meio de um comando ao módulo de E/S (STALLINGS, 2010, p. 179).

A figura 4.15 ilustra os diversos elementos de um sistema de E/S e sua posição.



4FIGURA 15.23 - Esquema de entrada e saída FONTE: Weber (2008, p. 82).

Tipos de transmissão de dados entre a UCP e os dispositivos de E/S

Conforme vimos na figura 4.14, as informações podem ser transmitidas dos/para dispositivos de E/S (ou interfaces) em grupos de bits ou 1 bit de cada vez. Isso é que caracteriza os dois tipos de transmissão de dados que temos:

a) Transmissão serial: os bits são transmitidos individualmente, um em seguida do outro, sendo, geralmente, mais lenta que a paralela e, portanto, utilizada em dispositivos com velocidade menor.

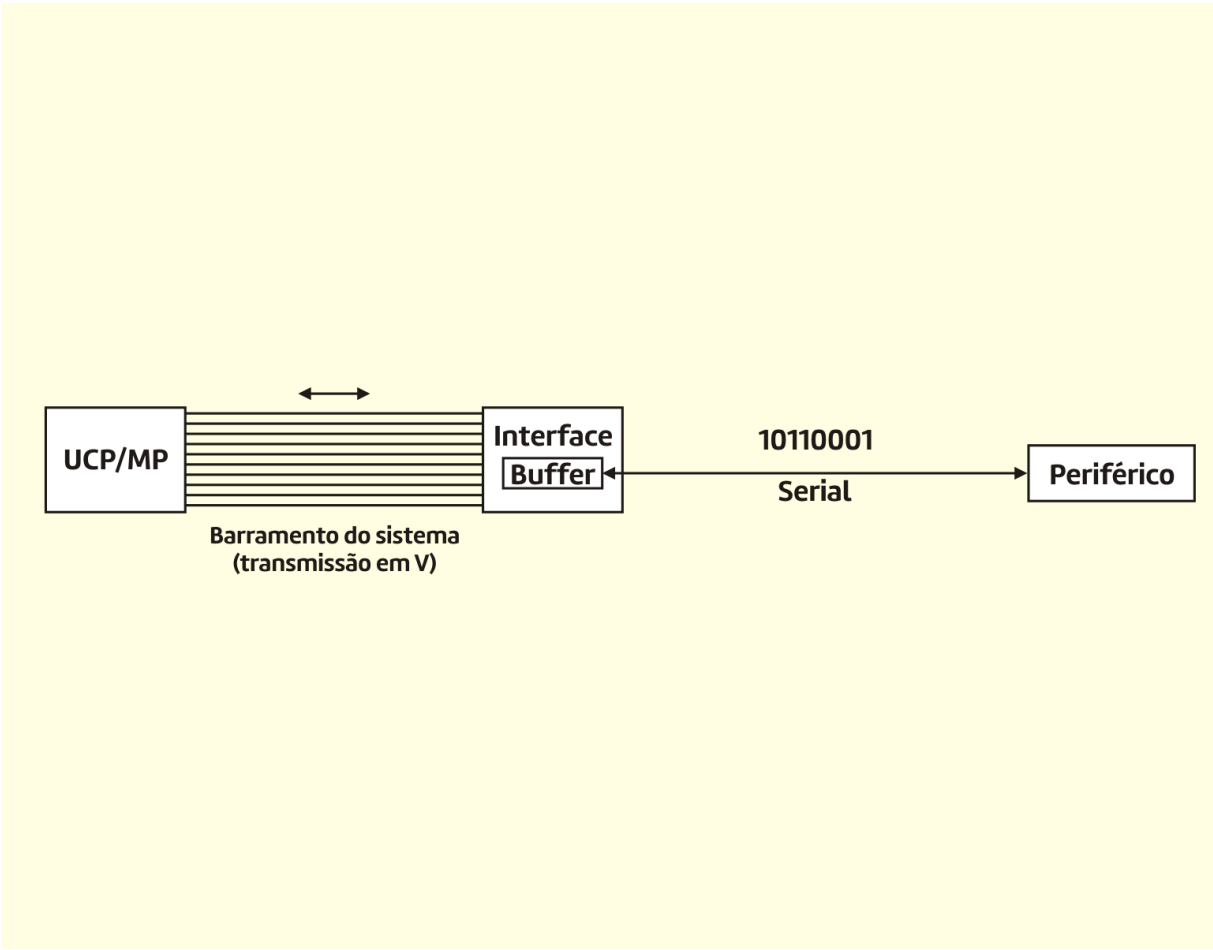
Para que a transmissão serial seja eficiente, tanto o transmissor quanto o receptor devem trabalhar na mesma velocidade, ou seja, têm que estar sincronizados com todos os bits, possuindo a mesma duração. É necessária, também, uma definição de como o bit inicial do caractere (grupo de bits com um significado) a ser transmitido/recebido a esse grupo será identificado. Esses são os caracteres de controle.

Nas palavras de Monteiro (2012, p. 383),

entre os interfaces mais modernos desenvolvidos para controlar a transmissão dos dados na forma serial existe o que se denomina USB – Universal Serial Bus. Este interface serve para interligar ao processador dispositivos do tipo: joysticks, teclados, scanners, telefones, impressoras, sem que se necessite de interfaces separados para cada um [...] permitindo a conexão de até 127 periféricos, ligando-se a uma única porta de saída no microcomputador.

O padrão USB também suporta velocidades bem altas.

A figura 4.16 mostra um exemplo de transmissão serial.



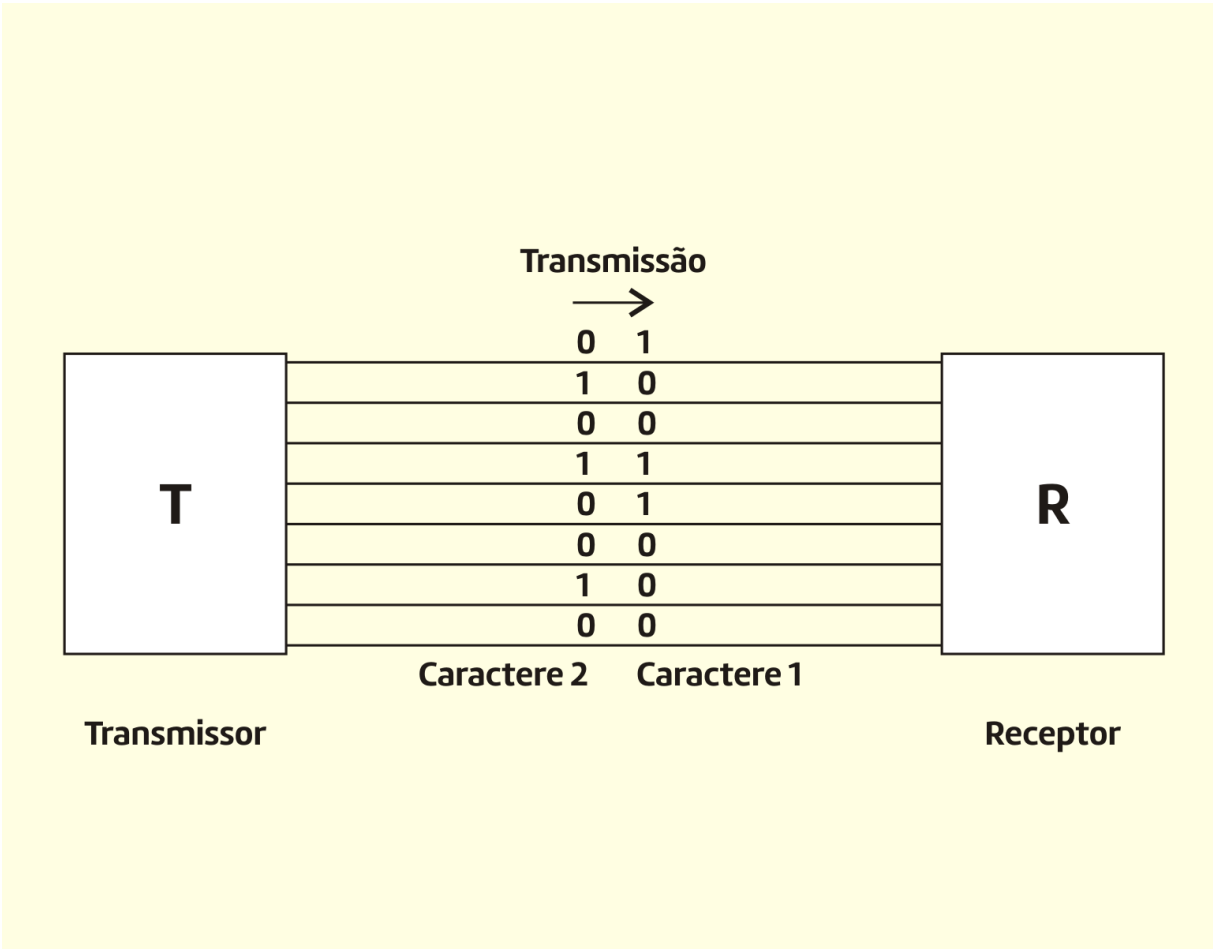
4FIGURA 16.23 - Exemplo de transmissão serial (interface-periférico) FONTE: Monteiro (2012, p. 379).

b) Transmissão paralela: os bits são transmitidos em grupos, ou seja, simultaneamente, cada um em uma linha de transmissão.

Como vários bits são enviados de uma vez, é mais rápida e mais cara, devido, justamente, à quantidade de linhas utilizadas.

Assim, é mais utilizada internamente no sistema de computação ou para periféricos ligados a curta distância. É um tipo de transmissão relativamente simples, sendo uma forma padrão utilizada nos sistemas computacionais.

A figura 4.17 mostra um exemplo de transmissão paralela.



4FIGURA 17.23 - Exemplo de transmissão paralela FONTE: Monteiro (2012, p. 384).

Métodos de realização das operações de E/S

Existem três métodos para realização das operações de E/S:

1. E/S programada ou por programa ou por *polling* (teste de estado)

Segundo Weber (2008, p. 83), ***"o processador pergunta para cada periférico se este está apto a receber ou transmitir uma unidade de informação (geralmente um byte) e, em caso afirmativo, realiza a transferência".***

O problema desse método é que a UCP precisa ficar continuamente verificando o dispositivo, o que acarreta uma grande perda de tempo, principalmente devido à diferença de velocidade entre os dispositivos e a UCP. Por isso, esse método não é mais utilizado.

¶ Reflita

"O problema com a E/S programada é que o processador tem que esperar muito tempo para que o módulo de E/S

> "O problema com a E/S programada é que o processador tem que esperar muito tempo para que o módulo de E/S de interesse esteja pronto para recepção ou transmissão de dados. O processador, enquanto espera, precisa interrogar repetidamente o estado do módulo de E/S" (STALLINGS, 2010, p. 184). Imagine a perda de tempo da UCP!

2. E/S acionada por interrupção

A alternativa para que a UCP não precise ficar continuamente ocupada interrogando o periférico é a E/S acionada por interrupção.

Stallings (2010, p. 184) explica bem esse método:

> o processador emite um comando de E/S para um módulo e depois continua realizando algum outro trabalho útil. O módulo de E/S, então, interromperá o processador para solicitar atendimento quando estiver pronto para trocar dados com o processador. O processador, então, executa a transferência de dados, como antes, e depois retorna seu processamento anterior.

3. E/S por acesso direto à memória (DMA – *Direct Memory Access*)

De acordo com Weber (2008, p. 83),

> na transferência de dados por acesso direto à memória, o processador não atua na transferência de cada byte individualmente. Um controlador encarrega-se da transferência de blocos de dados entre periféricos e memória, sem rotear cada byte por registradores internos ao processador e sem que o processador abandone suas tarefas correntes. O processador deve apenas inicializar o controlador e iniciar (disparar) a atividade de transferência.

Justamente por não ocupar o tempo da UCP, esse é o método de E/S utilizado atualmente para grandes volumes de dados.

Barramentos

A ligação entre a UCP e as interfaces e controladoras de periféricos é realizada pelo barramento de extensão.

Segundo Hennessy (2014, p. 394),

a maioria dos computadores permite que seus usuários conectem a eles novos tipos de dispositivos. O barramento de entrada/saída é o instrumento que permite a expansão da máquina, com o acréscimo de novos dispositivos periféricos. Para facilitar essa tarefa, a indústria de computadores desenvolveu vários padrões para barramentos. Tais padrões servem como uma espécie de especificação tanto para os fabricantes de computadores quanto para os fabricantes de dispositivos periféricos.

Os principais padrões de barramentos são:

1. ISA (*Industry Standart Adapter*)

É um dos padrões de barramento mais antigos e não é mais utilizado nos sistemas computacionais devido à sua taxa de transferência muito baixa.

2. PCI (*Peripheral Component Interconnet*)

É o padrão que veio para substituir o ISA, sendo bem superior a ele, podendo trabalhar com 32 ou 64 bits. Cada um dos controladores pode conectar até quatro dispositivos.

3. AGP (*Accelerated Graphics Port*)

Foi criado pela Intel para uso por placas de vídeo, pois possui um modo rápido de transferência, facilitando as transferências de dados em jogos e aplicativos gráficos.

4. *PCIe* ou *PCI-EX* (*PCI Express*)

Esse barramento substitui, ao mesmo tempo, os barramentos PCI e AGP, possuindo altas taxas de transferência. Permite a retirada ou conexão de um componente com o sistema ligado.

5. USB (*Universal Serial Bus*)

O USB, na realidade, é uma porta serial de alta velocidade, que permite a conexão de vários dispositivos. Sua principal vantagem é a tecnologia *plug-and-play*, que aceita que os dispositivos possam ser conectados ao computador e usados em seguida.

Dispositivos de armazenamento

Anteriormente, falamos sobre os dispositivos de E/S, sua classificação e os principais periféricos de cada categoria.

Nesse momento, vamos focar os dispositivos usados como memória secundária ou dispositivos de armazenamento. Esses dispositivos possuem um custo relativamente baixo, mas, se nos lembrarmos da nossa pirâmide de hierarquia de memória, eles estão na base, ou seja, são bem mais lentos do que a memória principal.

Esse custo mais baixo e a velocidade menor estão diretamente ligados à tecnologia de fabricação desses dispositivos.

As principais tecnologias utilizadas, atualmente, para a fabricação de dispositivos de armazenamento como memória secundária são:

a. Meios de armazenamento magnéticos: os principais exemplos dessa categoria são discos rígidos, mas podemos citar, também, os disquetes (lembram-se deles?) e a fitas magnéticas. As fitas magnéticas, apesar de não estarem presentes nos computadores pessoais, ainda são utilizadas como meio de armazenamento nos computadores de grande porte, principalmente para *backup*.
b. Meios de armazenamento óticos: seu principal exemplo são os CDs (*Compact Disc*) e os DVDs (*Digital Video Disc*). Como esses discos não são afetados pelas radiações eletromagnéticas, são seguros e confiáveis, além de possuírem um baixo custo e poderem ser usados somente para leitura ou ainda serem regraváveis.

Como os discos magnéticos ainda são os mais usados como memória secundária, por oferecer a melhor relação custo/benefício, vamos falar um pouco sobre o funcionamento deles.

Discos rígidos

Segundo Monteiro (2012, p. 406),

> no início da década de 1980, justamente com a explosão de demanda por microcomputadores, tanto para uso comercial quanto pessoal, cresceu também a necessidade de sistemas de armazenamento de maior capacidade... A IBM desenvolveu, então, uma tecnologia de fabricação de discos rígidos, de tamanho pequeno e compacto, baixo custo e desempenho elevado para o padrão dos sistemas de microcomputadores existentes. Esta tecnologia foi chamada de Winchester, embora o nome esteja relacionado (segundo alguns) ao famoso fuzil 30-30 Winchester. O fato é que a técnica ficou tão popular que todas as unidades de disco para micros vêm sendo fabricadas com essa tecnologia; como tem ocorrido com outros produtos comerciais de sucesso, o nome da tecnologia de fabricação se confundiu com o da própria unidade.

A analogia com o fuzil se dá por causa do disco rígido da época possuir dois módulos de 30 megabytes.

Se conhecermos um pouco mais sobre os discos rígidos ou, como chamados atualmente, winchesters ou HDs (*Hard Disc*), conseguimos entender por que eles funcionam em uma velocidade baixa.

O HD é formado por um ou mais discos magnéticos que podem possuir uma ou duas faces, recobertas por material magnetizável.

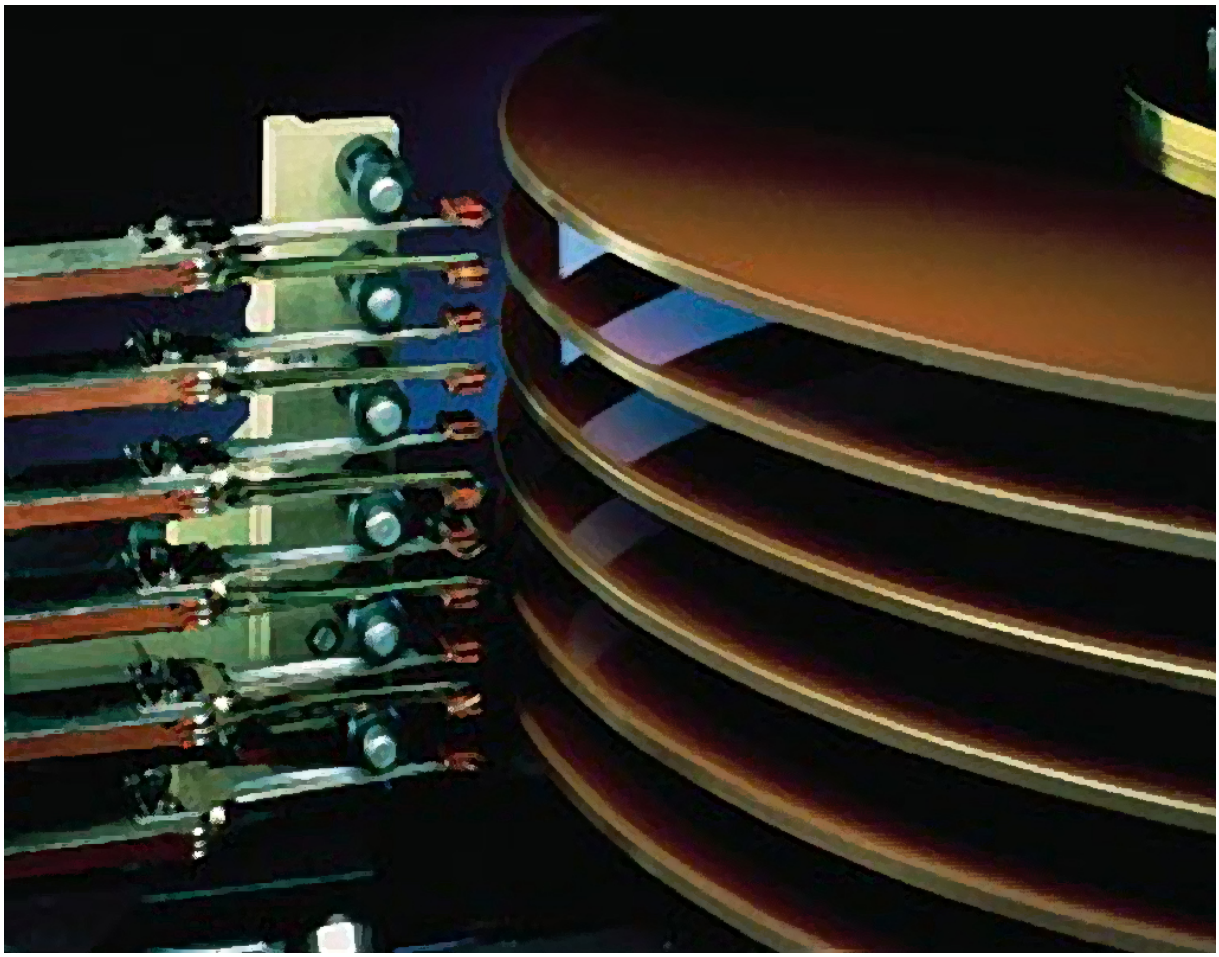
Tanenbaum (2007, p. 47) aponta que

um cabeçote de disco que contém uma bobina de indução flutua logo acima da superfície, apoiado sobre um colchão de ar (exceto para discos flexíveis, onde tocam a superfície). Quando uma corrente positiva ou negativa passa pelo cabeçote, ela magnetiza a superfície logo abaixo dele, alinhando as partículas magnéticas para a esquerda ou para a direita, dependendo da polaridade da corrente. Quando o cabeçote passa sobre uma área magnetizada, uma corrente positiva ou negativa é induzida nele, o que possibilita a leitura dos bits armazenados antes. Assim, à medida que o prato gira sob o cabeçote, uma corrente de bits pode ser escrita e mais tarde lida.

A figura 4.18 mostra o disco rígido de um computador pessoal.

4FIGURA 18.23 - Disco rígido FONTE: . Acesso em: 08 jan. 2017.

Se pensarmos em um disco com mais de uma superfície e duas faces, como os utilizados atualmente, temos a estrutura interna mostrada na figura 4.19.



4FIGURA 19.23 - Cabeças de leitura/gravação e braços de acesso FONTE: Capron (2004, p. 167).

Analisando a figura, vemos que cada uma das faces de cada superfície possui uma cabeça de leitura e gravação, ligada a um braço de acesso.

Essas faces são organizadas em trilhas e setores. As trilhas são anéis concêntricos divididos em setores de tamanho fixo. Assim, para que a cabeça de leitura/gravação acesse um determinado dado, deve ser fornecido seu endereço, compostos pelo número da superfície, da trilha e do setor.

O tempo de acesso a um HD é formado por três tempos menores: tempo de busca, de latência e transferência.

Monteiro (2012, p. 403) define bem esses tempos:

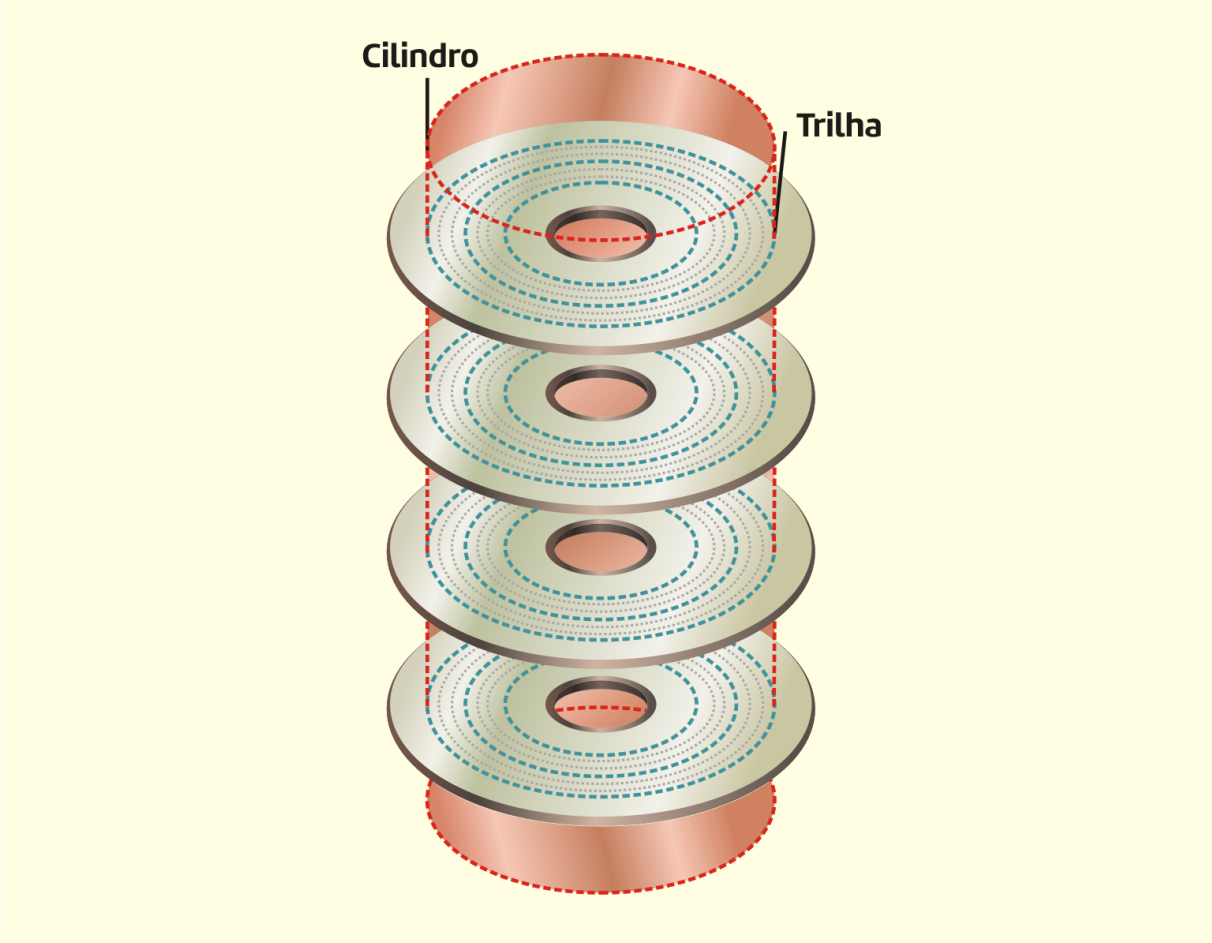
a. Tempo de SEEK (busca) – gasto para interpretação do endereço pela unidade de controle e movimento mecânico do braço para cima da trilha desejada. É o maior componente do tempo de acesso.
b. Tempo de latência – período decorrido entre a chegada da cabeça de leitura e gravação sobre a trilha e a passagem do bloco (setor) sobre a referida cabeça (depende da velocidade de rotação do disco).
c. Tempo de transferência – tempo gasto para a efetiva transmissão dos sinais elétricos (bits) para o destinatário.

Vamos voltar, agora, com a comparação com a nossa "vitrola". O tempo de busca é justamente o tempo que o usuário usava para posicionar a agulha de leitura no início da música.

Exatamente por depender de movimentos mecânicos, esse tempo é muito grande. Imagine se nossa informação estiver "espalhada" pelo disco e o braço mecânico tiver que ficar se movimentando de lá para cá.

Por isso é que surgiu o chamado acesso por cilindro. O cilindro é formado por todas as trilhas de mesma posição nas diferentes superfícies do disco. Assim, se as informações forem gravadas na sequência desse cilindro, pode ser acessada mais de uma trilha com somente um movimento do braço mecânico.

A figura 4.20 mostra um exemplo de organização de um disco magnético com múltiplos pratos.

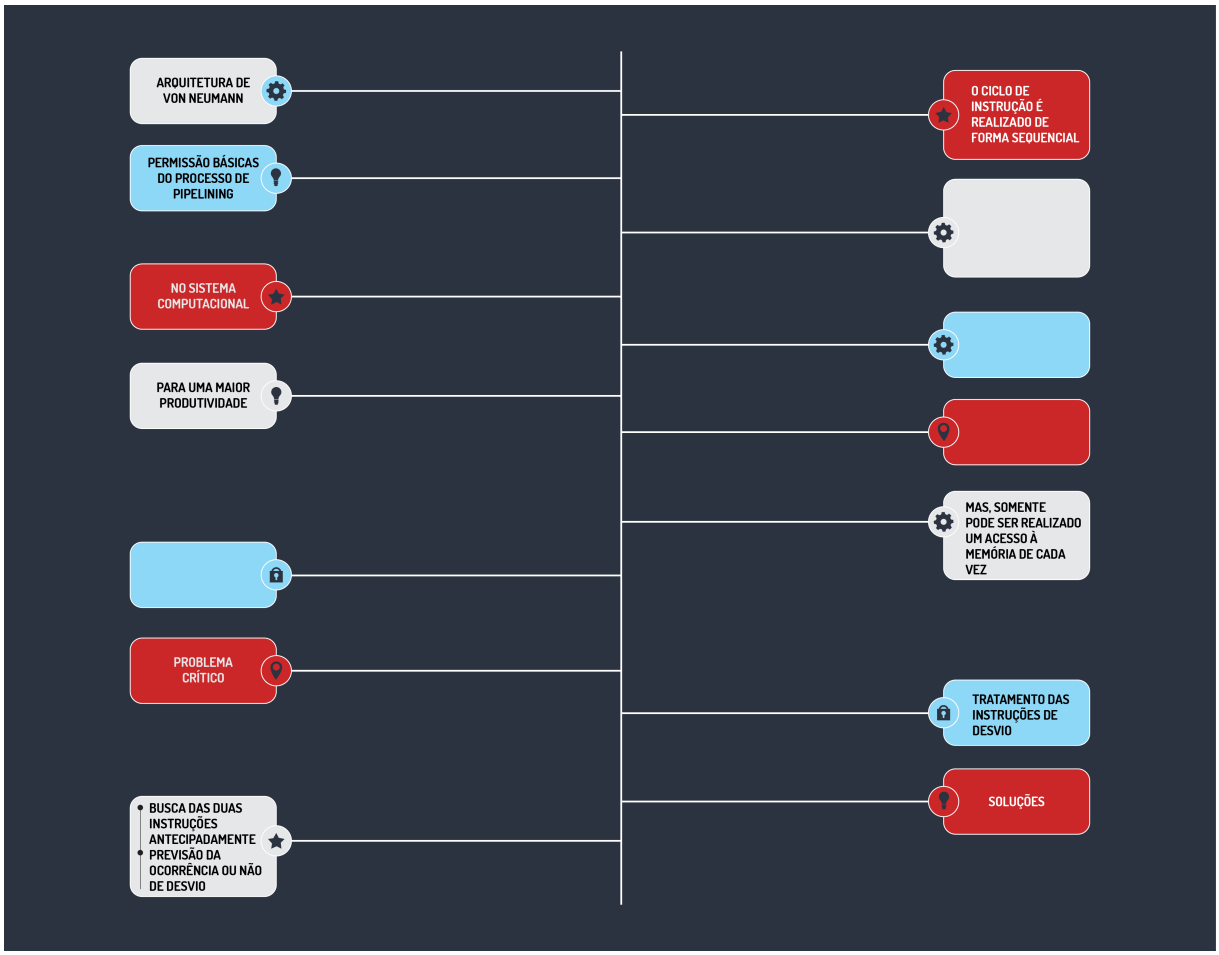


4FIGURA 20.23 - Organização de dados em cilindro FONTE: Capron (2004, p. 171).

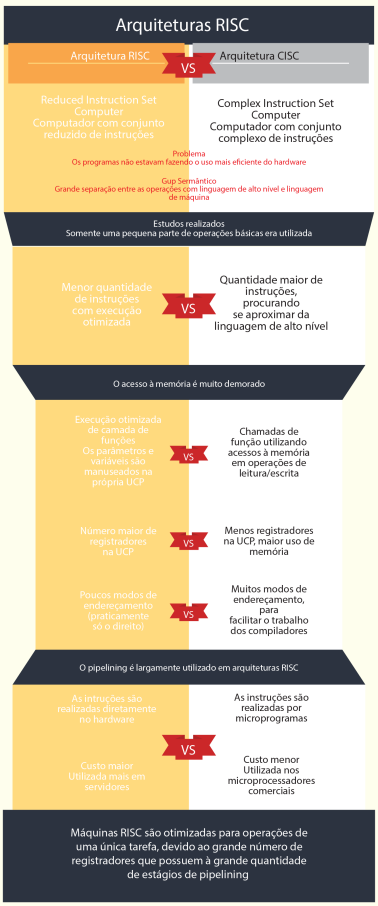
Inovações para melhorar o desempenho do sistema computacional

Muitas pesquisas têm sido realizadas visando melhorar o desempenho do processador, esperando conseguir cada vez mais velocidade. Mas, todas elas acabam "esbarrando" no funcionamento mais básico do processador: o ciclo de instruções. Por mais que se aumente a velocidade dos elementos de hardware, ainda precisamos seguir determinadas etapas para que a instrução seja executada. Para minimizar esses problemas, podemos falar da metodologia *pipelining*, das arquiteturas RISC e dos sistemas com multiprocessadores.

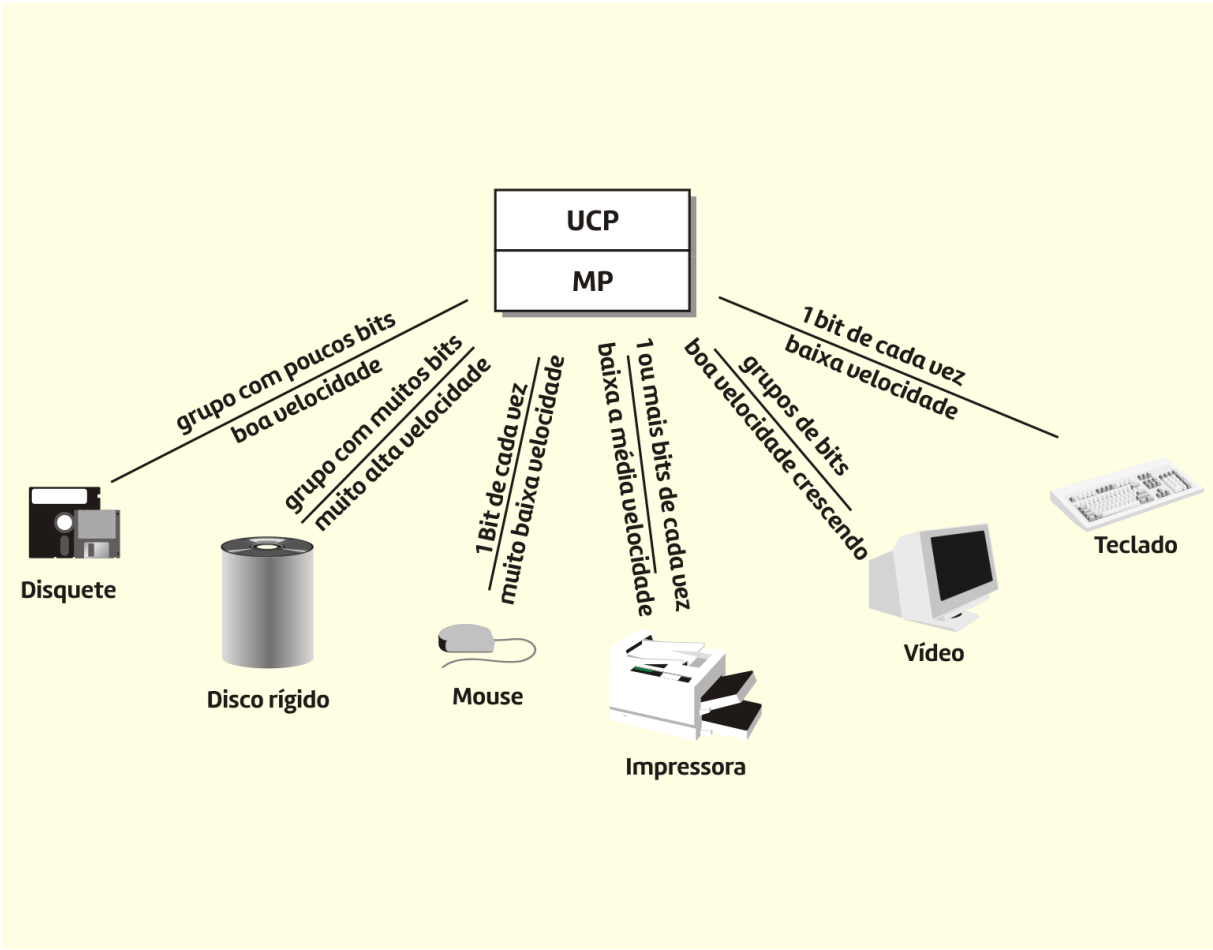
As figuras 4.21, 4.22 e 4.23 ilustram, respectivamente, a metodologia *pipelining*, a arquitetura RISC e os sistemas com multiprocessadores.



4FIGURA 21.23 - Metodologia *Pipelining* FONTE: A autora.



4FIGURA 22.23 - Arquitetura RISC FONTE: A autora.



4FIGURA 23.23 - Sistemas com multiprocessadores FONTE: A autora.

Indicação de leitura

Nome do livro: Arquitetura de Computadores Pessoais

Editora: Bookman

Autor: Raul Fernando Weber

ISBN: 85-241-0624-7

O livro, escrito por um professor da Universidade Federal do Rio Grande do Sul, surgiu da ideia de mostrar a arquitetura de um computador pessoal e seus componentes. São abordados temas como o computador na informática, microprocessadores, memórias, sistema de entrada e saída e periféricos. Possui uma linguagem bem clara e didática, facilitando bastante o aprofundamento dos conhecimentos já adquiridos e a obtenção de novas informações sobre os temas estudados.

Indicação de leitura

Nome do livro: Organização estruturada de computadores

Editora: Pearson Prentice Hall

Autor: Andrew S. Tanenbaum

ISBN: 85-7605-067-6

Segundo o próprio autor, o livro está baseado na ideia de que um computador pode ser considerado uma hierarquia de níveis, cada qual executando alguma função bem definida. Apresenta conceitos de portas e circuitos lógicos, organização de computadores, além de exemplos de programação em linguagem de montagem. É uma excelente opção para quem quer se aprofundar mais nos temas tratados nesta unidade e nas demais.

Conclusão

Quando falamos em arquitetura de computadores, estamos nos referindo à estrutura operacional de um sistema computacional, contemplando as condições necessárias para que um computador funcione de maneira adequada e como os seus componentes devem ser organizados de modo a se obter o melhor desempenho. Já a organização de computadores consiste justamente no estudo dos componentes físicos que compõem esse sistema.

Durante nossos estudos de Arquitetura e Organização de Computadores, trabalhamos com diversos conceitos e aplicações dessa área.

Iniciamos, na Unidade 1, com o estudo dos sistemas de numeração, principalmente o sistema binário, que é utilizado pelo computador, pois esse é formado de circuitos lógicos que entendem somente dois valores: 0 e 1. Vimos os principais métodos utilizados para a conversão entre as bases decimal, binária, octal e hexadecimal e definimos quais são os mais utilizados para cada um dos casos. Justamente por causa da importância da representação em binário, também aprendemos a realizar as operações aritméticas mais importantes nesse sistema e representar os números inteiros e fracionários em padrões que possam ser utilizados pelos computadores.

O conhecimento dos números binários nos ajudou a entender melhor os conceitos visualizados na Unidade 2, que tratou dos elementos mais básicos que formam o sistema computacional: as portas lógicas. Conhecendo a função de cada uma dessas portas, pudemos passar a trabalhar com os circuitos lógicos, aprendendo como eles funcionam e como projetar e implementar circuitos lógicos simples, com base na definição do problema que queremos resolver. Vimos, também, alguns circuitos já existentes e que executam funções específicas não só no computador como também em outros equipamentos. Nesse ponto, o aprendizado da Álgebra de Boole e dos Mapas ou Diagramas de Vetch-Karnaugh foi extremamente importante para que pudéssemos chegar à expressão simplificada dos circuitos lógicos.

Vistos esses conceitos básicos, na Unidade 3, começamos a trabalhar os fundamentos da organização de computadores. Vimos como é o funcionamento simplificado de um computador, os conceitos básicos de um sistema computacional e suas funções e operações, assim como a maneira como essas podem ser executadas. Aqui, o conhecimento do desenvolvimento dos computadores desde seus primórdios nos auxiliou muito a compreender os principais componentes do sistema computacional e como funciona a conexão entre eles. Quando trabalhamos de forma mais aprofundada com a Unidade Central de Processamento, pudemos entender como ela funciona e, principalmente, como o ciclo de instrução é executado. Também foram apresentados alguns conceitos de linguagem de montagem, para que pudéssemos entender como as informações são repassadas, pelo programador, para o computador.

Na Unidade 4, complementamos e finalizamos nossos estudos com os subsistemas de memória e de entrada e saída. Vimos como funcionam as diversas memórias existentes no computador individualmente e em grupo, focando nas memórias principal, cache e secundária. Também foi estudada a maneira como é feita a troca de informações entre os dispositivos de entrada e saída e a UCP e quais as formas existentes para realizar essa comunicação. Terminamos com algumas tecnologias que podem ser usadas para melhorar o desempenho do computador, como a *metodologia* pipelining, a arquitetura RISC e os computadores multiprocessadores.

Obrigada, caro(a) aluno(a), pela atenção e esperamos que tenham aproveitado, como nós, esses ensinamentos.

Grande abraço e até a próxima.

Referências

ALMEIDA, V. A. F. Análise e modelagem de desempenho de sistemas de computação. Atualização em 2009. Acesso em: 02 dez. 2016. <http://homepages.dcc.ufmg.br/~virgilio/download/perf2009-1/week1.pdf>

ANDRADE, G. E. Medida e Análise de Desempenho. Acesso em: 09 dez. 2016. <http://www.gileduardo.com.br/ifpr/oac/downloads/oac_aula14.pdf>

CANALTECH. O que é benchmark? [2016]. Acesso em: 10 out. 2016. <https://canaltech.com.br/o-que-e/o-que-e/O-que-e-benchmark--26350/>

CAPRON, H. L; JOHNSON, J. A. **Introdução à informática**. 8. ed. São Paulo: Pearson Prentice Hall, 2004.

CARVALHO, C. S. R. **Microprocessador 8085**. Campinas: Editora da UNICAMP, 1990.

GÜNTZEL, J. L.; NASCIMENTO, F. A. Introdução aos sistemas digitais. Atualização em 2008. Acesso em: 29 nov. 2016. <http://www.inf.ufsc.br/~j.guntzel/isd/isd.html>

HENNESSY, J. L; PATTERSON, D. A. **Arquitetura de computadores**: uma abordagem quantitativa. Rio de Janeiro: Campus, 2003.

HENNESSY, J. L.; PATTERSON, D. A. **Organização e projeto de computadores**: a interface Hardware/Software. 4. ed. Rio de Janeiro: Campus, 2014.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de Eletrônica Digital**. 41. ed. São Paulo: Érica, 2012.

MONTEIRO, M. A. **Introdução à organização de computadores**. 5. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 2012.

RAINER, K.R.; DEGIESLK, C.G. **Introdução a sistemas de informação**. 3. ed. Rio de Janeiro: Elsevier, 2012.

RAMOS, L. F. **Conversas sobre números, ações e operações**: uma proposta criativa para o ensino da matemática nos primeiros anos. São Paulo: Ática, 2011.

RICARTE, I. L. M. Organização de Computadores. Universidade Estadual de Campinas. Acesso em: 05 nov. 2016. (ftp://ftp.dca.fee.unicamp.br/pub/docs/ea960/ea960.pdf)

Significados. Acesso em 10/11/2016. <https://www.significados.com.br/>

SOUZA FILHO, G.; ALEXANDRE, E. S. M. **Introdução à computação**. 2. ed. João Pessoa: UFPB, 2014.

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

TANENBAUM, A. S. **Organização estruturada de computadores**. 5. ed. São Paulo: Pearson Prentice Hall, 2007.

TECHTUDO. O que é setup? Tire suas dúvidas sobre como funciona o sistema. Atualizado em 18/02/2015. Acesso em: 30 nov. 2016. <http://www.techtudo.com.br>

TECHTUDO. SSD e HD: entenda a diferença entre as tecnologias. Acesso em: 06 jan. 2017. <http://www.techtudo.com.br/artigos/noticia/2011/06/qual-diferenca-entre-hd-e-ssd.html>

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais**: princípios e aplicações. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

WEBER, R. F. **Arquitetura de Computadores Pessoais**. 2. ed. Porto Alegre: Bookman, 2008.

WEBER, R. F. **Fundamentos de Arquitetura de Computadores**. 4. ed. Porto Alegre: Bookman, 2012.

Atividades

Atividades - Unidade I

Notação posicional, Sistemas numéricos e Conversões

Temos quatro métodos que podem ser utilizados para realizar a conversão entre as bases numéricas. São eles: método polinomial, método das subtrações, método das divisões e método da substituição direta. Cada um desses métodos é adequado para a conversão de determinadas bases. Levando isso em consideração, qual é o método que deve ser utilizado quando se deseja converter um número em decimal para as bases binária, octal ou hexadecimal?

A) Método da substituição direta.

B) Método das divisões.

C) Método polinomial.

D) Método das subtrações.

E) Todos os métodos são adequados.

Aritmética no Sistema Binário

O estouro de bits ou *overflow* é um problema muito sério no sistema computacional, pois compromete o resultado encontrado. Ele ocorre quando o número de bits utilizado em uma operação aritmética é excedido. Levando em consideração os valores, em decimal, 137 e 89, converta-os para a representação em 8 bits, em binário, e realize a soma. Após isso, assinale a alternativa correta:

A) o resultado da adição é 11100010 e ocorreu *overflow*.

B) O resultado da adição é 11100010 e não ocorreu *overflow*.

C) O resultado da adição é 11000010 e ocorreu *overflow*.

D) o resultado da adição é 11000010 e não ocorreu *overflow*.

E) o resultado da adição é 11010010 e não ocorreu *overflow*.

Representação Numérica em Ponto Fixo

Utilizando a notação em Complemento de 2, realize a subtração 28 – 10, em binário com 6 bits, e analise o resultado, verificando se houve overflow ou não. Em seguida, assinale a alternativa correta:

A) O resultado da operação é 010010 e não houve *overflow*, pois não ocorreu nenhum dos vai um.

B) O resultado da operação é 010010 e não houve *overflow*, pois ocorreu vai um para o bit de sinal e para fora.

C) O resultado da operação é 010110 e houve *overflow*, pois não ocorreu nenhum dos vai um.

D) O resultado da operação é 010110 e houve *overflow*, pois ocorreu vai um somente para o bit de sinal.

E) O resultado da operação é 010010 e houve *overflow*, pois ocorreu vai um somente para fora.

Representação em Ponto Flutuante

Assim como na conversão de valores com ponto fixo, temos também métodos específicos para converter valores fracionários da base decimal para a base binária. Utilizando os conteúdos estudados, transforme o valor decimal 175, 90625 para seu correspondente em binário e depois assinale a alternativa que apresenta o valor encontrado:

A) O valor resultante é 1010111,11101.

B) O valor resultante é 10101111,11101.

C) O valor resultante é 10101111,1101.

D) O valor resultante é 1001001,10101.

E) O valor resultante é 1100110,11011.

Atividades - Unidade II

Portas e Operações Lógicas

As portas lógicas, também chamadas *gates*, realizam operações sobre suas variáveis de entrada, fornecendo resultados de acordo com a função que desempenham. Cada uma destas portas lógicas possui uma representação algébrica diferente. Assinale a alternativa que apresenta essa representação para as portas *AND*, *OR* e XOR, respectivamente:

A) S = A. B;$S = \bar{A}$; $S = A\,B$

B) S = A. B; S = A + B; $S = A \quad B$

C) S = A. B; $S = A\,B$; S = A + B

D) S = A + B; S = A. B; $S = A\,B$

E) $S = \bar{A}$; S = A + B; $S = A\,B$

Noções de Álgebra Booleana

A Álgebra de Boole é uma ferramenta muito importante para a realização de simplificações de expressões booleanas. É por meio de seus postulados, identidades, propriedades e teoremas que conseguimos obter circuitos bem menores. Assinale a alternativa que apresenta propriedades da álgebra booleana:

A) Distributiva, aditiva e associativa.

B) Comutativa, distributiva e associativa.

C) Comutativa, distributiva, associativa e multiplicativa.

D) Complementar, aditiva e multiplicativa.

E) Ordenativa, distributiva e associativa.

Diagramas ou Mapas de Vetch-Karnaugh

Os mapas de Karnaugh são uma das formas que pode ser utilizada para simplificarmos expressões booleanas, onde agrupamos os valores lógicos 1 para reduzi-las. Assinale a alternativa que contém todos os agrupamentos possíveis para 3 variáveis:

A) Hexa, oitavas, quadras, pares e termos isolados.

B) Oitava, quadras, pares e termos isolados.

C) Pares e termos isolados.

D) Quadra, pares e termos isolados.

E) Oitava, pares e termos isolados.

Introdução aos Circuitos Combinatórios

Um conjunto de portas lógicas que possui sua saída dependente somente das variáveis de entrada é chamado de circuito combinatório, que pode ser projetado para resolver situações diversas. Mas temos alguns destinados a aplicações específicas, por exemplo, os multiplexadores, decodificadores e comparadores. Assinale a alternativa que apresenta conceitos corretos sobre esses circuitos:

A) Os circuitos comparadores verificam as entradas do sistema e fornecem o sinal lógico 0 se as duas forem iguais.

B) Os circuitos multiplexadores podem ser utilizados para a seleção de uma determinada entrada.

C) Os circuitos comparadores podem ser utilizados para comparar, também, as saídas do sistema.

D) Os circuitos decodificadores, assim como os multiplexadores, podem ser utilizados para a seleção de uma entrada.

E) Os circuitos multiplexadores podem ser utilizados para efetuar a conversão serial-paralelo.

Atividades - Unidade III

Conceitos básicos

O computador não trabalha com programas codificados em linguagem de alto nível, como C, C++, C#, Java, entre outras, e sim com a chamada linguagem de máquina, composta por instruções simples que executam pequenas atividades no computador. Assinale a alternativa que não contém uma operação que pode ser realizada por uma instrução de máquina:

A) Cálculos aritméticos.

B) Comunicação com a internet.

C) Desvios na sequência do programa.

D) Movimentação de bits.

E) Operações lógicas.

Evolução histórica dos computadores

Os computadores modernos foram classificados em gerações, de acordo com sua tecnologia e desempenho. Cada uma dessas gerações foi marcada por inovações diferentes. Assinale a alternativa que apresenta as inovações da segunda e quarta gerações:

A) Válvulas e circuitos integrados.

B) Transistores e microprocessadores.

C) Microprocessador e dispositivos multimídia.

D) Transistores e circuitos integrados.

E) Circuitos integrados e internet.

Componentes de um sistema de computação

Para cada função que o computador executa existe uma série de placas e equipamentos que a torna possível. Cada função básica também pode ser chamada de Unidade, assim temos: Unidade de Entrada, Unidade de Saída, Unidade de Processamento e Unidade de Armazenamento, cada unidade com seus respectivos equipamentos e placas (OLIVEIRA, 2007). Levando esses dados em consideração, principalmente a questão dos equipamentos, assinale a alternativa que contém elementos correspondentes, respectivamente, à Unidade de Entrada, Unidade de Saída, Unidade de Processamento e Unidade de Armazenamento:

A) Caixa de som; Processador; HDs; Tela sensível ao toque.

B) Tela sensível ao toque; Caixa de som; Processador; HDs.

C) HDs; Processador; Caixa de som; Tela sensível ao toque.

D) Tela sensível ao toque; HDs; Processador; Caixas de som.

E) Processador; Tela sensível ao toque; HDs; Caixa de som.

Unidade Central de Processamento (UCP) e Linguagem de Montagem

A quantidade e o uso dos registradores variam bastante de modelo para modelo de UCP. Além dos registradores de dados, a UCP possui sempre outros registradores que não participam diretamente da função processamento, com funções específicas ou que funcionam para a área de controle. Entre estes registradores, pode-se citar o Registrador de Instrução (RI) e o Contador de Instrução (CI), além do Registrador de Endereços de Memória (REM) e o Registrador de Dados de Memória (RDM). Levando isso em consideração, é correto afirmar que:

A) O registrador de dados armazena a próxima instrução a ser executada pela UCP.

B) O registrador de instrução armazena a instrução que está sendo executada no momento.

C) O registrador de endereços da memória é utilizado pela UCP e dispositivos de entrada e saída para comunicação e transferência de informações.

D) O contador de instrução armazena a quantidade de operações que foram realizadas.

E) O registrador de endereços, juntamente com o registrador de dados, é responsável pelo envio de sinais de controle para a UCP.

Atividades - Unidade IV

Subsistemas de memória

(FCC, 2010) Na hierarquia de velocidade de armazenamento de dados, na sequência da mais alta para a mais baixa, se encontram:

A) Memória cache, memória principal, memória secundária e registradores.

B) Registradores, memória cache, memória principal e memória secundária.

C) Memória principal, memória secundária, memória cache e registradores.

D) Memória principal, memória cache, memória secundária e registradores.

E) Memória cache, memória secundária, memória principal e registradores

Memória principal

(ESAF, 2004) "Quando se abre um documento do Word, esse documento será copiado do disco rígido para a memória, porque a memória permite um acesso muito mais rápido para que se façam modificações nesse documento. Quando se edita esse documento, as modificações surgem instantaneamente na tela, mas, enquanto não são salvas no disco rígido, elas não se tornam efetivas."

Analisando esse texto, é correto afirmar que o termo "a memória":

A) Indica a memória ROM.

B) Indica a memória RAM.

C) Indica BIOS.

D) Está aplicado de forma incorreta. O correto seria substituí-lo por "o processador".

E) Está aplicado de forma incorreta. O correto seria substituí-lo por "o chipset da placa mãe".

Memória cache

(FCC, 2008) No microcomputador, uma memória Cache Ll encontra-se, fisicamente:

A) Dentro do processador e funcionalmente entre os registradores do processador.

B) Dentro do processador e funcionalmente entre o processador e a memória RAM.

C) Fora do processador e funcionalmente entre o processador e a memória RAM.

D) Fora do processador e funcionalmente entre o processador e o buffer do HD.

E) Fora do processador e funcionalmente entre a memória RAM e o buffer do HD.

Subsistema de entrada e saída

(CESGRANRIO, 2008) Em um computador, a memória de massa ou memória secundária é:

A) Definida como uma sequência de células numeradas, cada uma contendo uma pequena quantidade de informação.

B) Usada para gravar grande quantidade de dados que não serão perdidos com o desligamento do computador.

C) Representada por um barramento ou canal de dados com velocidade de acesso superior à memória RAM.

D) Armazenada no processador secundário do computador para processamento dos dados nela contidos.

E) Usada para interligar os dispositivos periféricos conectados ao barramento de dados do computador.