ELEMENTOS DE LÓGICA DIGITAL

UNITAU digital

ANTONIO ESIO MARCONDES SALGADO

Antonio Esio Marcondes Salgado

Elementos de Lógica Digital

Taubaté 2022

Unitau-Reitoria
Rua Quatro de Março,432, Centro
Taubaté – São Paulo. CEP:12.020-270
Central de Atendimento:0800557255

Polo Taubaté – Sede
Rua Conselheiro Moreira de Barros, 203 - Centro
Taubaté – São Paulo. CEP:12.010-080
Telefones: Coordenação Geral: (12) 3621-1530
Secretaria: (12) 3622-6050

edUNITAU

EXPEDIENTE EDITORA

Sistema Integrado de Bibliotecas - SIBi/ UNITAU
Grupo Especial de Tratamento da Informação – GETI

Salgado, Antonio Esio Marcondes
S164e Elementos de lógica digital [recurso eletrônico] / Antonio Esio Marcondes Salgado. – Dados eletrônicos. -- Taubaté : EdUnitau, 2022.

Formato: PDF
Requisitos do sistema: Adobe
Modo de acesso: world wide web

ISBN: 978-65-86914-51-1 (on-line)

1. Lógica. 2. Digital. 3. Sistemas. I. Título.

CDD – 511.3

Ficha catalográfica elaborada pela Bibliotecária Ana Beatriz Ramos – CRB-8/6318

Índice para Catálogo sistemático

Lógica – 511.3
Digital – 511.3
Sistemas – 511.3

Sumário

Recursos de Imersão:

Explorando ideias

Eu indico

Pensando juntos

Pípulas de conhecimento

Podcast

QRCode

Elementos de Lógica Digital

Unidade I

Conceitos de sistemas de numeração, representações e conversão de sistemas de numeração e operações aritméticas

UNIDADE I

Introdução

Fonte: Pixabay

Nesta Unidade serão abordados os conceitos de sistemas de numeração em sistemas digitais, suas representações, conversão de números em diferentes representações e operações aritméticas com números binários.

1.1 Conceitos de diferentes sistemas de numeração; representação de sistemas de numeração; conversão de números em diferentes sistemas de numeração

Os dados em computadores são registrados em diferentes formatos e têm em comum a característica de que podem ser representados a partir de sinais elétricos gerados em circuitos eletrônicos que processam combinações lógicas.

Os números que utilizamos no dia a dia são representados em formato decimal ou em formato adequado ao entendimento de quem os manipula, porém utilizar este tipo de representação em computadores e demais sistemas digitais requer, em um sistema decimal por exemplo, pelo menos dez maneiras diferentes de representar cada dígito.

Neste sentido, a representação utilizando dois estados lógicos apenas, “zeros” e “uns”, e a combinação destes valores, simplifica demais a construção de computadores e sistemas. Estes sinais são processados e gerados em circuitos digitais com diferentes capacidades de processamento, porém sempre considerando estes dois valores de saída.

Os sistemas de numeração são definidos de acordo com diferentes BASES numéricas, que são o conjunto de valores que cada dígito pode ter. Veja:

- Binário: 0 ou 1 (Base 2).
- Octal: 0, 1, 2, 3, 4, 5, 6, 7 (Base 8).
- Decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (Base 10).
- Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (Base 16, onde A = 10, B = 11, C = 12, D = 13, E = 14, F = 15).

Em qualquer representação, o valor correspondente em DECIMAL de uma BASE é a soma dos produtos dos dígitos da respectiva BASE pela potência na posição que o mesmo ocupa (ajustado nos valores das grandezas de suas posições). Para ficar mais claro, seguem alguns exemplos de representação DECIMAL de diferentes números em diferentes bases. Observe como se formam:

$0110_{2} \Rightarrow 0\times2^{3} + 1\times2^{2} + 1\times2^{1} + 0\times2^{0} = 6_{10}$

$123_{10} \Rightarrow 1\times10^{2} + 2\times10^{1} + 3\times10^{0} = 123_{10}$

$23_{8} \Rightarrow 2\times8^{1} + 3\times8^{0} = 19_{10}$

$1C5_{16} \Rightarrow 1\times16^{2} + 12\times16^{1} + 5\times16^{0} = 453_{10}$

Observe a notação:

$$\mathbf{NÚMERO_{base}}$$

Os próximos parágrafos serão dedicados a exemplos de diferentes sistemas numéricos e à forma de conversão em BINÁRIO.

Um número BINÁRIO é representado por um conjunto de bits, que configura uma PALAVRA.

Uma PALAVRA pode ter 4 bits, 8 bits (byte), 16 bits etc. O tamanho da PALAVRA determina a faixa de valores que podem ser representados pelo número BINÁRIO.

Veja:

PALAVRA de 4 bits: $0000_2 = 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 0$

$1111_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 15$

Ou seja, uma PALAVRA de 4 bits pode representar 16 números, ou 2^4 números.

Podemos então identificar quantos números podem ser representados por 8 bits?

Resposta: PALAVRA de 8 bits => 2^8 diferentes valores => 256 valores

Convertendo DECIMAL para BINÁRIO

A conversão de números DECIMAIS para números BINÁRIOS implica em saber qual a representação deste número decimal em potência de 2. Esse processo é bastante simples e consiste em fazer sucessivas divisões do número por 2, obtendo a correspondência de potências de 2 que compõem o número DECIMAL.

Com o exemplo a seguir, a compreensão ficará mais simples:

Seja o número N = 175_{10}, queremos saber este número na Base = 2 (binário).

Passo 1: vamos dividir sucessivamente o número N por 2, como na tabela a seguir:

i	***Ni***	***Ni+1*** (= *Ni* /2)	***di (resto)***
0	175	87	**1**
1	87	43	**1**
2	43	21	**1**
3	21	10	**1**
4	10	5	**0**
5	5	2	**1**
6	2	1	**0**
7	1	0	**1**

Onde:

"***i***" = número de BITs na minha conversão binária, e começa em "0", que corresponde ao 1º BIT, e vai até última tentativa de divisão por 2 possível. Veja na divisão acima que foi possível dividir por 2 sete vezes, e na oitava e última sobrou "1". Foram, portanto, 8 divisões, que resultou em um número binário de 8 BITs.

Passo 2: o resultado em "***di (resto)***" será a representação do número em binário
Ou seja:

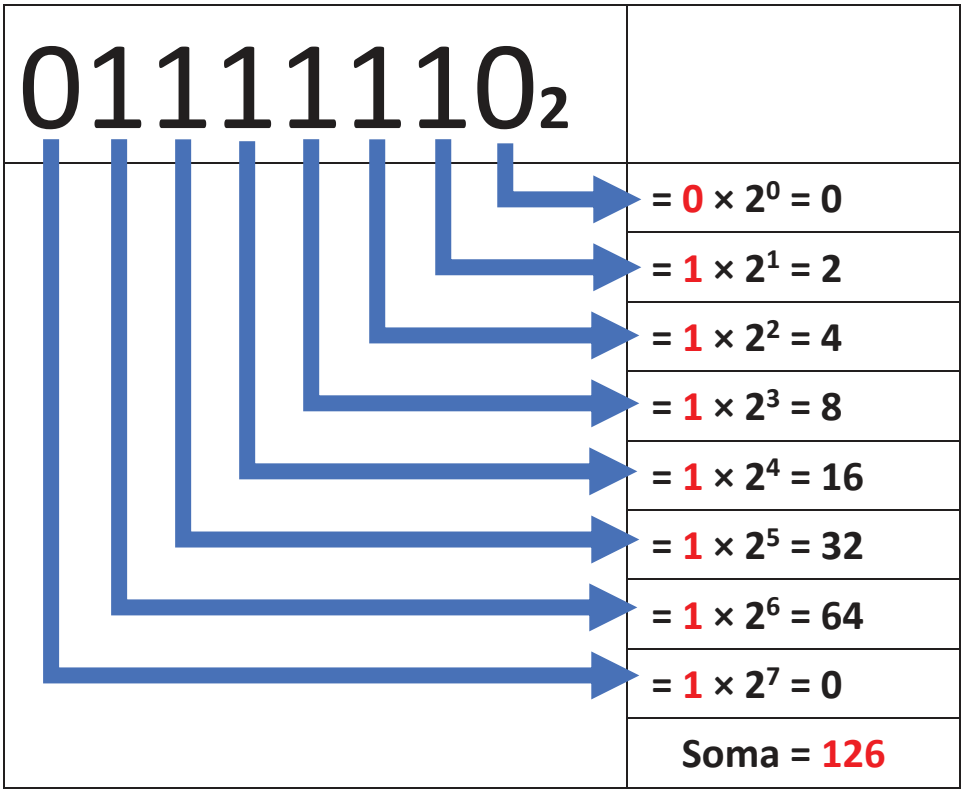
$$175_{10} = 10101111_2$$

Convertendo BINÁRIO para DECIMAL

Conforme visto, o conjunto de bits, que forma a PALAVRA, representa um número BINÁRIO que pode ser convertido em DECIMAL com simples conversão de cada bit em decimal de acordo com sua posição na PALAVRA.

A posição do bit na PALAVRA representa a potência de 2 que deve ser somada ao resultado.

Por exemplo: dado o número BINÁRIO = 0111 11102, qual o seu correspondente em DECIMAL?



Ou seja:

$01111110_2 = 126_{10}$

Convertendo BINÁRIO para HEXADECIMAL

Lembre-se que HEXADECIMAL é Base 16, e neste caso cada numeral HEXADECIMAL é representado por 4 BITs.

Veja:

$0_{16} = 0000_2$	$8_{16} = 1000_2$
$1_{16} = 0001_2$	$9_{16} = 1001_2$
$2_{16} = 0010_2$	$A_{16} = 1010_2$
$3_{16} = 0011_2$	$B_{16} = 1011_2$
$4_{16} = 0100_2$	$C_{16} = 1100_2$
$5_{16} = 0101_2$	$D_{16} = 1101_2$
$6_{16} = 0110_2$	$E_{16} = 1110_2$
$7_{16} = 0111_2$	$F_{16} = 1111_2$

Por exemplo: o número BINÁRIO 0111 1110 0001_2 corresponde a que número HEXADECIMAL?

Passo único: agrupar o BINÁRIO de 4 em 4 BITs, e obter o equivalente em HEXADECIMAL. Veja:

HEXADECIMAL	7	E	1
	⎵	⎵	⎵
BINÁRIO	0111	1110	0001_2

Convertendo HEXADECIMAL para DECIMAL

Por exemplo: converter o HEXADECIMAL 7E1 em DECIMAL.

Agora veja:

HEXADECIMAL	7	E	1
	⎵	⎵	⎵
DECIMAL	7×16^2	14×16^1	1×16^0
	= 1792	= 224	= 1
$1792 + 224 + 1 = 2017_{10}$			

Ou seja,

$\mathbf{7E1_{16} = 2017_{10}}$

1.2 Conversão de números fracionários; operações aritméticas com números binários

Convertendo NÚMEROS DECIMAIS menores que 1 para BINÁRIO

Neste caso, o tratamento dado para a parte inteira já foi explicado, que é a sucessiva divisão por 2.

Mas, para a parte menos que 1 ($1 \geq X$), ou seja, a parte fracionária, a solução é fazer sucessivas multiplicações por 2 para se obter as potências de 2 negativas que representam a respectiva parte fracionária.

Mais uma vez, é mais simples de entender com um exemplo, abordando somente a parte fracionária.

Por exemplo: converter o DECIMAL 0,7265625 em BINÁRIO.

Passo 1: vamos multiplicar sucessivamente o número N por 2, como na tabela a seguir:

-i	Valor decimal	Valor decimal × 2	*Inteiro*
1	0,7265625	1,453125	**1**
2	0,453125	0,90625	**0**
3	0,90625	1,8125	**1**
4	0,8125	1,625	**1**
5	0,625	1,25	**1**
6	0,25	0,5	**0**
7	0,5	1	**1**

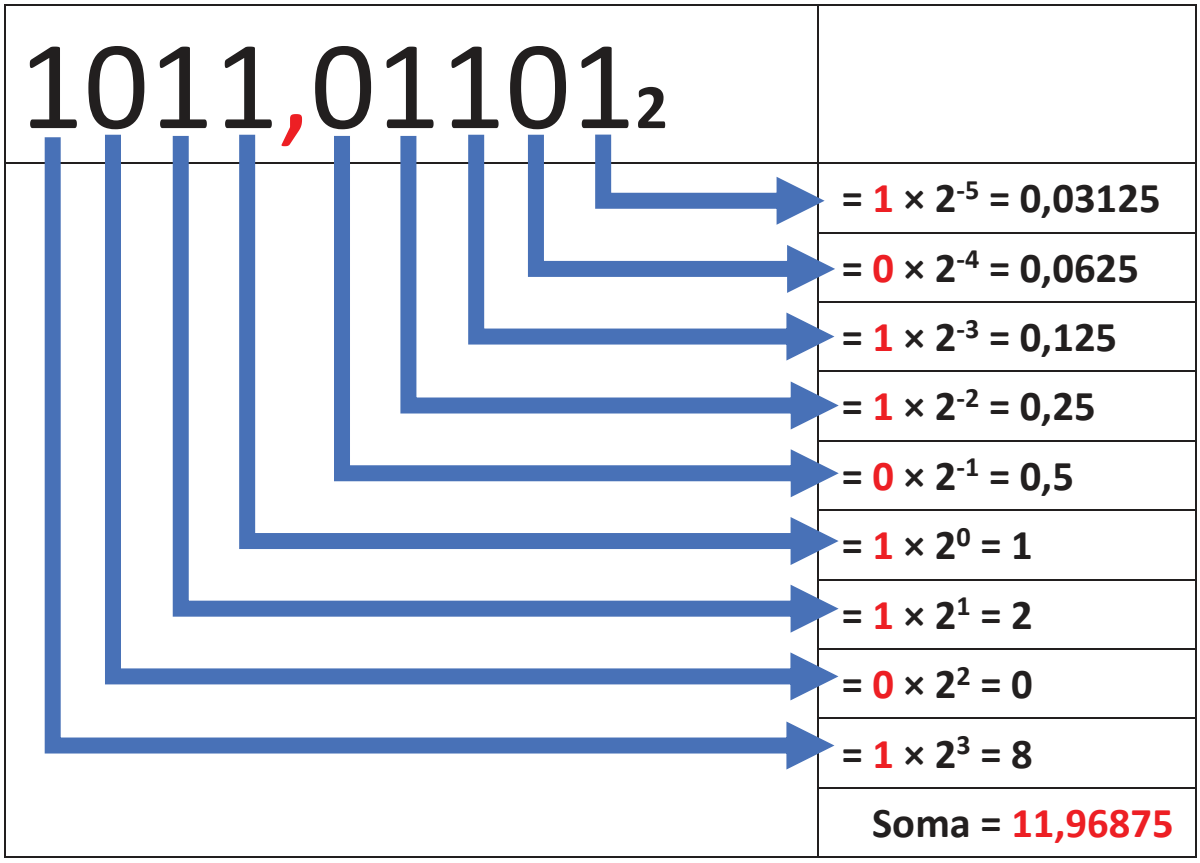
Temos então:

$$0{,}7265625_{10} = 0{,}1011101_2$$

Convertendo BINÁRIO fracionário para DECIMAL

Esta conversão é feita com a soma das potências de 2 negativas que compõem a parte fracionária, e a soma das potências de 2 (positivas) que compõem a parte inteira do número BINÁRIO.

Por exemplo: converter o BINÁRIO 1011,011012 em DECIMAL.



Ou seja:

$1101{,}01101_2 = 11{,}96875_{10}$

Operações aritméticas com números BINÁRIOS

Adição

Observe na tabela a seguir os resultados de operações bit a bit.

Circuitos eletrônicos, de lógica digital, fazem este mesmo tipo de operação, chamada de lógica combinacional.

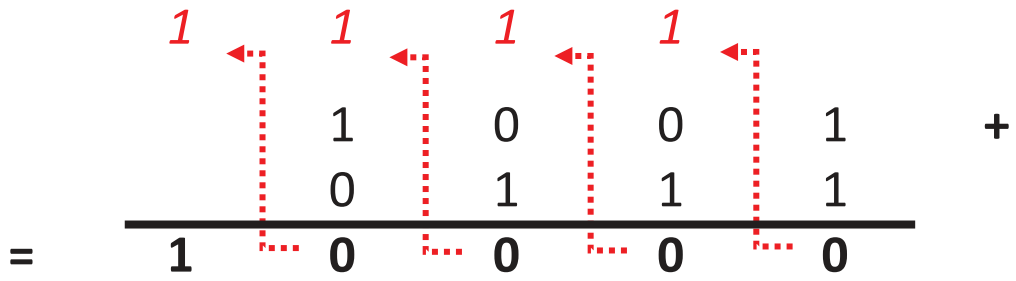
0 + 0 = 0	**0 + 1 = 1**	**1 + 0 = 1**	**1 + 1 = 10**	**1 + 1 + 1 = 11**

Transporta “1” para o dígito de ordem superior.

Transporta “1” para o dígito de ordem superior.

Exemplo de uma ADIÇÃO:

$1001_2 + 0111_2 = 10000_2$



Subtração

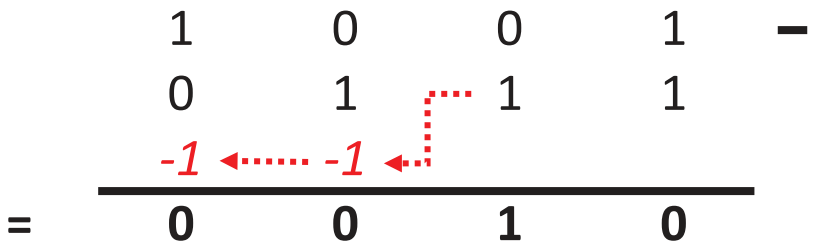
Observe na tabela a seguir os resultados de operações bit a bit em uma SUBTRAÇÃO:

0 - 0 = 0	**0 - 1 = 1**	**1 - 0 = 1**	**1 - 1 = 0**

Empresta “1” do dígito de ordem superior.

Exemplo:

$1001_2 - 0111_2 = 0010_2$



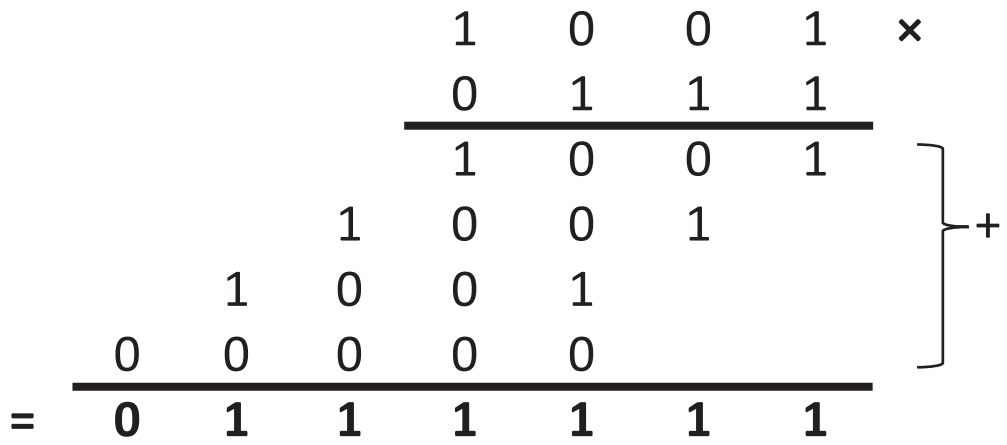
Multiplicação

A tabela a seguir mostra o resultado de multiplicação bit a bit, com todas as combinações possíveis.

0 × 0 = 0	**0 × 1 = 0**	**1 × 0 = 0**	**1 × 1 = 1**

Exemplo:

$1001_2 \times 0111_2 = \mathbf{111111_2}$

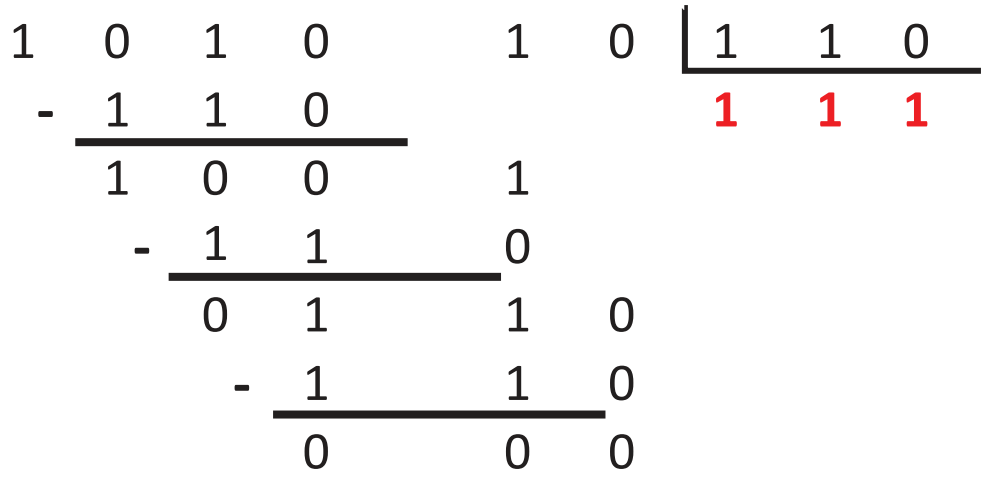


Divisão

A DIVISÃO em números BINÁRIOS usa o mesmo método que o decimal: deslocamentos e subtrações.

Exemplo:

$101010_2 \div 110_2 = \mathbf{111_2}$



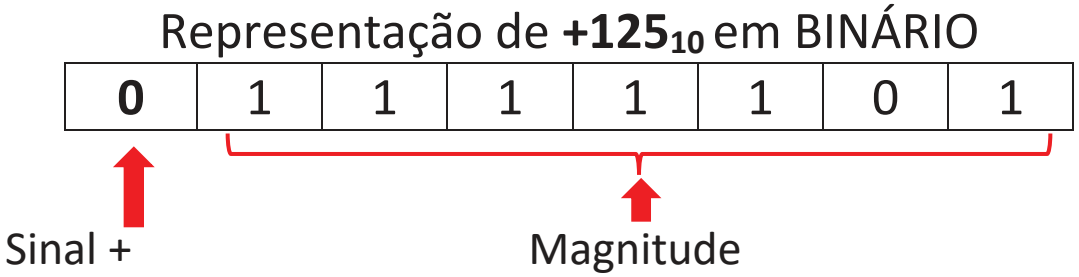
1.3 Representação de números com sinal; sistema sinal-módulo; complemento de 1 (C1); complemento de 2 (C2); excesso de 2 elevado a (N-1); overflow em operações aritméticas com C1 e C2

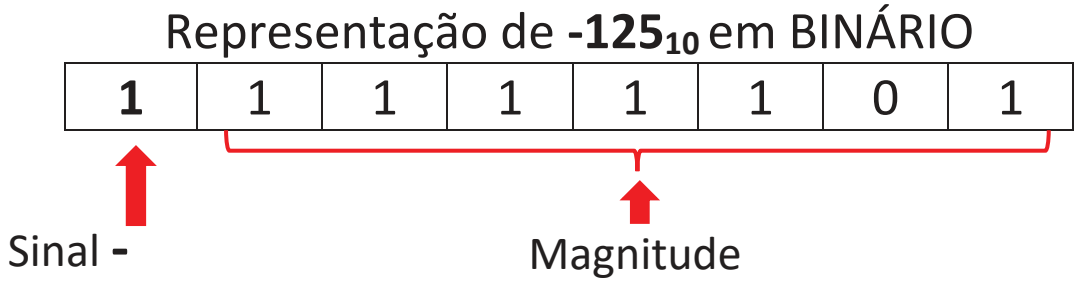
As operações aritméticas em computadores podem envolver números com diferentes graus de precisão e também números positivos e negativos. A seguir, serão apresentados os principais métodos para representar números inteiros N.

Sistema Sinal-Módulo ou Sinal-Magnitude (SM)

Suponha um número BINÁRIO de "N" bits.

- O bit que está situado mais à esquerda representa o sinal, e seu valor será "0" para o sinal "+" e "1" para o sinal "-".
- Os bits restantes (N-1 bits) representam a magnitude (ou módulo) do número.
- Esses números não possuem parte fracionária, e a vírgula está implícita à direita.





O número "X" pode variar de:

$$-2^{N-1} + 1 \leq X \leq 2^{N-1} - 1$$

No caso do exemplo N = 8, podem ser representados os números "X":

$$-127 \leq X \leq 127$$

Inconveniente: duas representações para o "0":

0000 0000 (+0) e 1000 0000 (-0)

Complemento de 1 (C1)

- O bit à esquerda representa o sinal "+" e "-" (como no sistema sinal-módulo SM).
- Números positivos: N-1 bits da direita definem o módulo (análogo ao SM).
- Números negativos: é necessário obter o simétrico do correspondente.
- Número positivo através do complemento de todos os seus dígitos, ou seja, trocando "0" por "1", inclusive o bit de sinal.

Exemplo: representação do número +12 e -12 com N = 8 bits.

- +12 = 0 000 1100
- -12 = 1 111 0011

Veja a tabela a seguir:

Decimal (positivo)	BINÁRIO	Decimal (negativo)	BINÁRIO em C1
0	0 000 0000	0	1 111 1111
1	0 000 0001	- 1	1 111 1110
2	0 000 0010	- 2	1 111 1101
3	0 000 0011	- 3	1 111 1100
4	0 000 0100	- 4	1 111 1011
⋮	⋮	⋮	⋮
126	0 111 1110	- 126	1 000 0001
127	0 111 1111	- 127	1 000 0000

Faixa de representação de N dígitos é de:

$$-2N-1 + 1 \leq X \leq 2N-1 - 1$$

Inconveniente: duas representações para o “0”:

0 000 0000 (+0) e 1 111 1111 (-0)

Soma em Complemento de 1 (C1)

- Dois números são somados da mesma forma que na soma BINÁRIA;
- Quando ocorrer "transporte" de "1" para o dígito mais significativo à esquerda este "1" deve ser somado ao resultado.

Exemplo: soma de +12 com -5 em C1:

		0	0	0	0	1	1	0	0	(+12)
	+	1	1	1	1	1	0	1	0	+(-5)
Transporte	(1)	0	0	0	0	0	1	1	0	
Soma o transporte									1	
		0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	+7

Complemento de 2 (C2)

- O bit à esquerda representa o sinal "+" e "-" (como no sistema sinal-módulo SM e no C1).
- Números positivos: N-1 bits da direita definem o módulo (mais uma vez análogo ao SM, e também análogo ao C1).
- Números negativos: é necessário obter o simétrico do correspondente número positivo da seguinte maneira:
    - 1º passo: obtém-se o complemento de todos os seus dígitos, ou seja, trocando "0" por "1", inclusive o bit de sinal.
    - 2º passo: ao resultado obtido soma-se "1" (soma binária), desprezando-se o transporte.

Exemplo: representação em C2 do número +12 e -12 com N = 8 bits.

- +12 = 0 000 1100
- -12 => 1º passo: obter o C1 => 1 111 0011

  2º passo: somar "1" ao C1 e desprezar o transporte

  1 111 0011 + 1 = 1 111 0100

  - 12 = 1 111 0100

Veja a tabela a seguir:

Decimal (positivo)	BINÁRIO	Decimal (negativo)	BINÁRIO em C1
0	0 000 0000	-1	1 111 1111
1	0 000 0001	- 2	1 111 1110
2	0 000 0010	- 3	1 111 1101
⋮	⋮	⋮	⋮
126	0 111 1110	- 127	1 000 0001
127	0 111 1111	- 128	1 000 0000

Faixa de representação de N dígitos é de:

$$-2^{N-1} \leq X \leq 2^{N-1} - 1$$

$$-128 \leq X \leq 127$$

Inconveniente: assimetria entre números positivos e negativos.

Vantagem: única representação para “0”.

Excesso de 2 elevado a (N-1)

- Nenhum bit é utilizado para sinal, ou seja, todos os bits representam um módulo ou valor.
- O valor corresponde ao número representado mais um “excesso”, que para “N” bits é igual a 2(N-1).
- Exemplo: para N = 8 bits o excesso é 2(8-1) = 128.
- Neste caso, a representação do número +12 e -12 seria dada por:

  +12 seria representado por 1000 1100, veja:

  12 + 128 = 140

  0000 1100 + 1000 0000 = 1000 1100

-12 seria representado por 1000 1100, veja:

-12 + 128 = 116

-0000 1100 + 1000 0000 = 0111 0100

- A faixa de representação para "N" bits é dada por:

$$-2^{N-1} \leq X \leq 2^{N-1}-1$$

- No caso de N = 8 bits, a faixa é $-128 \leq X \leq 127$.
- Observe que a faixa é assimétrica, e há uma única representação para "0", que no caso de 8 bits fica:

(0) = 1000 0000

- Todo número representado em excesso de 2 é igual à sua correspondente representação em complemento de 2 (C2), com o 1º dígito (sinal) trocado, ou seja, com sinal invertido.

Soma em Complemento de 2 (C2)

- Dois números são somados da mesma forma que na soma BINÁRIA.
- Quando ocorrer "transporte" de "1" para o dígito mais significativo à esquerda, este "1" deve ser desprezado.

Exemplo: soma de +12 com -5 em C2:

		0	0	0	0	1	1	0	0	(+12)
	+	1	1	1	1	1	0	1	1	+(-5)
Transporte desprezado	(1)	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	+7

Erro de "overflow" no uso de operações aritméticas com C1 e C2

Operações aritméticas utilizando Complemento de 1 (C1) ou Complemento de 2 (C2) podem ter resultado com sinal ("+" ou "-") diferente do que deveria ocorrer, o que indica "erro de overflow", ou seja, o resultado ultrapassou o "range" (a faixa) de números possíveis com o número de bits utilizado.

Exemplo: soma de +126 com +7 em C1

	0	1	1	1	1	1	1	0	(+126)
+	0	0	0	0	0	1	1	1	+ (+7)
bit de sinal ficou em "1" ➔	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	+133

Ou seja: houve "erro de overflow" (estourou o range possível).

Você se lembra? O range admitido com N = 8 é:

$$-2^{N-1} + 1 \leq X \leq 2^{N-1} - 1 \Rightarrow -127 \leq X \leq +127$$

Agora, a mesma soma em C2:

	0	1	1	1	1	1	1	0	(+126)
+	0	0	0	0	0	1	1	1	+ (+7)
bit de sinal ficou em "1" ➔	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	+133

Ocorre o mesmo "erro de overflow".

O range admitido em C2 é:

$$-2^{N-1} \leq X \leq 2^{N-1} - 1 \Rightarrow -128 \leq X \leq +127$$

1.4 Síntese da Unidade

Nesta Unidade, foram abordados diferentes modos de representar números em sistemas digitais e as operações aritméticas mais simples envolvendo estes números.

Os conceitos são bastante simples, mas fundamentais para compreender a lógica de funcionamento de diferentes sistemas computacionais e seu funcionamento.

1.5 Para Saber Mais

Vídeos:

CÓDIGO binário: aprenda fácil, 03 fev. 2018. 1 vídeo (10min21s). Publicado por Atech-Info. Disponível em: https://youtu.be/Y0dTHv65yxU. Acesso em: 27 set. 2021.

COMO transformar um número decimal em binário, 10 out. 2016. 1 vídeo (2min). Publicado por GCF Aprende Livre. Disponível em: https://youtu.be/mttrG_kbHN4. Acesso em: 27 set. 2021.

ERROS - Representação de Números e Aritmética de Ponto Flutuante - Representação de Números, 19 ago. 2018. 1 vídeo (9min16s). Publicado por Responde Aí. Disponível em: https://youtu.be/YHquEw23qRo. Acesso em: 27 set. 2021.

REPRESENTAÇÃO Numérica - Números Inteiros e Ponto Fixo (Eletrônica Digital - 06), 30 mar. 2020. 1 vídeo (11min04s). Publicado por Prof. Edil. Disponível em: https://youtu.be/7i3shg5Ss90. Acesso em: 27 set. 2021.

Sites:

PROENÇA, Alberto José. Representação binária de números - notas de estudo. Universidade do Minho, Departamento de Informática, 2004. Disponível em: http://gec.di.uminho.pt/LEi/sc/RepresNum_Mar04.pdf. Acesso em: 27 set. 2021.

TRISTÃO, Isadora. Números binários: o que são, para que servem e como calculá-los. **Conhecimento Científico**, s.d. Disponível em: https://conhecimentocientifico.com/numeros-binarios/. Acesso em: 27 set. 2021.

UNIVERSIDADE DE SÃO PAULO. Escola politécnica. Departamento de Engenharia de Sistemas Eletrônicos PSI - EPUSP. Laboratório de Instrumentação Elétrica, s.d. Disponível em: https://edisciplinas.usp.br/pluginfile.php/3643948/mod_resource/content/0/Representa%C3%A7%C3%A3o%20de%20N%C3%BAmeros%20em%20Bin%C3%A1rio%20e%20Hexadecimal.pdf. Acesso em: 27 set. 2021.

Unidade II

Representação em "ponto fixo" e em "ponto flutuante", e formato IEEE 754 para ponto flutuante

UNIDADE II

Introdução

Fonte: Freepik

Nesta Unidade, serão abordados os conceitos de representação de números utilizando o conceito de ponto fixo e ponto flutuante em sistemas computacionais, e recomendação IEEE 754 para representação e uniformização de representação em ponto flutuante.

2.1 Representação de dados em "ponto fixo"; forma de representação, intervalo de representação e conversões; representação de números utilizando o conceito de "ponto flutuante" em sistemas digitais; mantissa, base exponencial e expoente

Representação de dados em "ponto fixo" e "ponto flutuante"

Em computadores e sistemas digitais, os números reais são representados por números BINÁRIOS ("0" e "1").

Ocorre que o número de bits em um número BINÁRIO é limitado, e isso tem como consequência uma limitação na representação do número real, havendo truncamento em valores muito grandes ou em números com parte fracionária muito grande.

O conjunto de bits que representa um número é chamado de "PALAVRA". O tamanho da PALAVRA pode variar em cada computador ou sistema, daí a informação de um computador ou sistema operar em 32 bits, 64 bits, 128 bits e assim por diante.

Representação de dados em "ponto fixo"

Nesta representação, a posição de um ponto decimal é fixa, daí o nome "ponto fixo". Para uma PALAVRA de N bits, deve ser informado o número de bits para a parte fracionária.

Podemos usar a notação:

- PALAVRA = N bits.
- Parte inteira = t bits ($t \geq 0$).
- Parte fracionária = f bits ($f \geq 0$).

Exemplo: uma PALAVRA de N = 8 bits em Complemento de 2 (C2) com t = 3 e f = 4 tem:

- 1 bit para sinal (s).
- 3 bits para representar a parte inteira (t).
- 4 bits para representar a parte fracionária (f).

Vamos usar como exemplo o número BINÁRIO em C2 que corresponde ao número 47_{10}:

0 0 1 0 1 1 1 1

Veja que, dependendo da definição de t e de f, esta PALAVRA pode representar diferentes números:

0 0 1 0 1 1 1 1 $= (0\times2^6+1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0) = 47$

0 0 1 0 1 1 1, 1 $= (0\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0+1\times2^{-1}) = 23{,}5$

0 0 1 0 1 1, 1 1 $= (0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}) = 11{,}75$

0 0 1 0 1, 1 1 1 $= (0\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}+1\times2^{-3}) = 5{,}875$

0 0 1 0, 1 1 1 1 $= (0\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2}+1\times2^{-3}+1\times2^{-4}) = 2{,}9375$

0 0 1, 0 1 1 1 1 $= (0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}+1\times2^{-3}+1\times2^{-4}+1\times2^{-5}) = 1{,}96875$

0 0, 1 0 1 1 1 1 $= (0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}+1\times2^{-5}+1\times2^{-6}) = 0{,}984375$

0, 0 1 0 1 1 1 1 $= (0\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}+1\times2^{-5}+1\times2^{-6}+1\times2^{-7}) = 0{,}9921875$

Para uma determinada notação em "ponto fixo", deve ser indicado quantos bits para:

Parte inteira (*t*).

Parte fracionária (*f*).

Todos os números manipulados devem seguir a mesma notação.

A quantidade de valores representáveis é dada por 2^N, onde "N" é o número total de bits da PALAVRA, e a faixa de valores representáveis depende da posição da vírgula.

Como observado no exemplo, os números fracionários estão separados entre si por uma diferença igual a 2^{-f}.

Operações de soma e subtração em "ponto fixo" são realizadas da mesma forma que aprendemos para números inteiros, mas observe:

- Somente podem ser somados ou subtraídos números que apresentem a mesma posição para a vírgula.
- Caso os números não tenham a mesma posição de vírgula, é necessário "normalizar" estes números, colocando ambos com a mesma posição de vírgula.

2.2 Representação de números utilizando o conceito de "ponto flutuante": normalização e faixa de representação em "ponto flutuante" em sistemas digitais

Representação de dados em "ponto flutuante"

A notação em "ponto flutuante" surgiu por conta da necessidade de representar números reais com amplitude maior que a faixa proporcionada pela notação em "ponto fixo".

A imprecisão no caso do "ponto flutuante" depende do tamanho da PALAVRA utilizada no computador ou sistema digital:

- Na representação em "ponto flutuante", é utilizada a seguinte notação:
- Número = Mantissa × $\text{Base}^{(\text{Expoente})}$

$$N = m \times b^e$$

A precisão do número em "ponto flutuante" é definida pelo número de bits da mantissa, e a faixa de representação é definida pelo número de bits do expoente.

Considerando que a "Base" é constante em um sistema digital (pode ser "2", pode ser "10" etc.), o número em "ponto flutuante" é representado por (m,e), onde "m" pode ser representado em Sinal-Módulo (SM), Complemento de 1 (C1) ou Complemento de 2 (C2), e "e" é o expoente, sempre inteiro, positivo ou negativo.

Representação de dados em "ponto flutuante": normalização

Um número utilizando a notação em "ponto flutuante" pode ter infinitas representações. Exemplo: 1.000.000 pode ser representado por:

$$\mathbf{1{,}0 \times 10^6 \text{ ou } 0{,}1 \times 10^7 \text{ ou } 10{,}0 \times 10^5}$$

Desta forma, é conveniente utilizar uma forma "normalizada", na qual a mantissa não tem parte inteira, ou seja, a mantissa só tem a parte fracionária, e o primeiro dígito à direita da vírgula é diferente de "0", exceto na representação do "0".

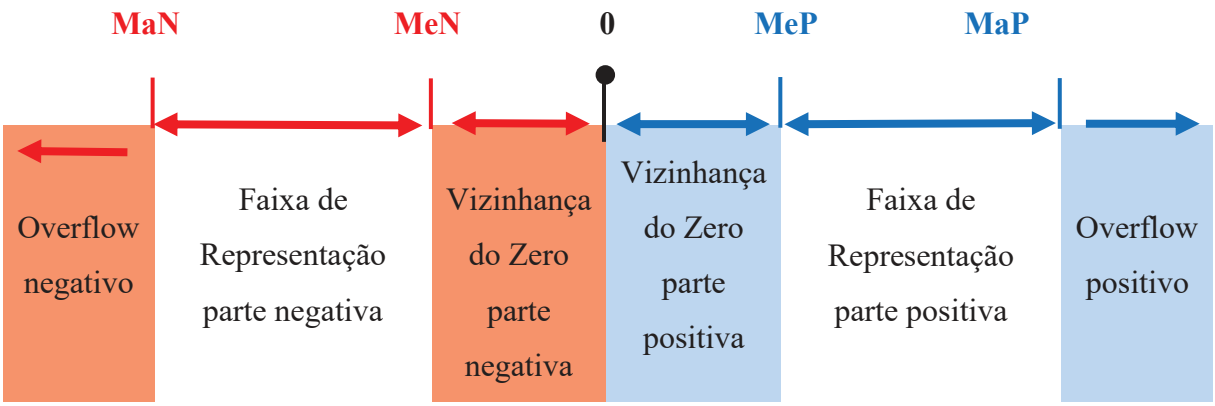
Neste sentido, a normalização da representação de 1.000.000 é dada por $0{,}1 \times 10^7$.

Com base binária, a normalização da mantissa exige que seus dois bits mais significativos sejam diferentes:

- Para números em Complemento de 2 (C2), é necessário que o dígito mais significativo da mantissa e o seu bit de sinal sejam diferentes.
- Assim, para números positivos, a mantissa deve iniciar por (0,1)2 e para números negativos, iniciar por (1,0)2.

Conforme visto anteriormente, a normalização pode ser feita por deslocamentos da mantissa para a direita ou esquerda, com respectivos incrementos ou decrementos do expoente.

Faixa de representação em Ponto Flutuante



Um número X pode ser:

$$\mathbf{MaN \leq X \leq MeN} \text{ ou}$$
$$\mathbf{X = 0} \text{ ou}$$
$$\mathbf{MaN \leq X \leq MeN}$$

Onde:

- **MaN** = **Ma**ior amplitude **N**egativa que se pode representar, que corresponde a:
  MaN =Máximo Negativo de mantissa × Base$^{(\text{máximo Expoente } \textbf{positivo})}$
  $\mathbf{MaN = Manm \times B^{Ep}}$

- **MeN** = **Me**nor amplitude **N**egativa que se pode representar, que corresponde a:
  MeN = Mínimo Negativo de mantissa × Base$^{(\text{máximo Expoente } \textbf{negativo})}$
  $\mathbf{MeN = Minm \times B^{En}}$

- **MeP** = **Me**nor amplitude **P**ositiva que se pode representar, que corresponde a:
  MeP = Mínimo Positivo de mantissa × Base$^{(\text{máximo Expoente } \textbf{negativo})}$
  $\mathbf{MeP = Mipm \times B^{En}}$

- **MaP** = **Ma**ior amplitude **P**ositiva que se pode representar, que corresponde a:
  MaP = Máximo Positivo de mantissa × Base$^{(\text{máximo Expoente } \textbf{positivo})}$
  $\mathbf{MaP = Mapm \times B^{Ep}}$

Existem quatro "zonas" nas quais não conseguimos representar os números (coloridas no desenho apresentado anteriormente), que são:

- **Overflow negativo** (valores abaixo do limite mínimo negativo de representação).
- **Overflow positivo** (valores acima do limite máximo positivo de representação).
- **Vizinhança do Zero**, parte negativa (valores entre o "0" e o primeiro valor negativo representado).
- **Vizinhança do Zero**, parte positiva (valores entro o "0" e o primeiro valor positivo representado).

A quantidade de números que podem ser representados dentro da faixa de representação (parte negativa e positiva) é finita, havendo casos de números que não podem ser representados.

2.3 Formato IEEE 754 para ponto flutuante

Formato de Ponto Flutuante IEEE 754

O IEEE (Institute of Electrical and Electronics Engineers) publicou a recomendação 754 com proposta de formato de representação de "ponto flutuante", com o objetivo de criar um padrão compatível entre diferentes famílias de computadores, de diferentes fabricantes.

Nesta proposta:

- O bit de sinal é representado no bit mais significativo (à esquerda).
- Os bits seguintes representam o expoente, que usa representação em excesso.
- Os bits menos significativos representam a mantissa, que deve ser ajustada para ter somente a parte fracionária.

Sinal (**S**) *"0" ou "1"*	Expoente (**E**) em excesso *8, 11 ou 15 bits*	Mantissa (**M**) *23, 52 ou112 bits*

O IEEE 754 propõe 3 formatos:

	Sinal (S) 1 bit	Expoente (E) 8 bits	Mantissa (M) 23 bits
Precisão **Simples** (32 bits)	31	30 23	22 0

	Sinal (S) 1 bit	Expoente (E) 11 bits	Mantissa (M) 52 bits
Precisão **Dupla** (64 bits)	63	62 52	51 0

	Sinal (S) 1 bit	Expoente (E) 15 bits	Mantissa (M) 112 bits
Precisão **Quádrupla** (128 bits)	127	126 112	111 0

Os grupos de números podem ser representados como números normalizados, zero, números não normalizados e não números (NaN).

O valor do número é dado por:

$$N = (-1)^{S} \times (M) \times 2^{E \text{ em excesso}}$$

Em números normalizados utilizam expoentes “E” em excesso, com 8 bits (1 a 254), 11 bits (1 a 2046) e 15 bits (1 a 32766); o primeiro bit da mantissa é sempre zero, e por isso não é representado.

O excesso para o expoente “E” é definido em:

- 127 para palavra de 32 bits.
- 1023 para palavra de 64 bits.
- 16383 para palavra de 128 bits.

O “zero” é representado com E = 0 e M = 0.

Para uma palavra de 32 bits:

Sinal (S)	Expoente (E)	Mantissa (M)	Valor
0	0000 0000	000 0000 0000 0000 0000 0000	+0
1	0000 0000	000 0000 0000 0000 0000 0000	-0

O infinito é representado pelo maior valor do expoente (E = 255 ou 2047 ou 32767) e por uma fração em zero (M = 0).

Para uma palavra de 32 bits:

Sinal (S)	Expoente (E)	Mantissa (M)	Valor
0	1111 1111	000 0000 0000 0000 0000 0000	+Inf
1	1111 1111	000 0000 0000 0000 0000 0000	-Inf

NaN: não números, eles são representados pelo maior expoente e por uma fração diferente de zero.

NaN são usados para representar códigos de erro e situações imprevistas em sistemas digitais, entre outros.

Sinal (S)	Expoente (E)	Mantissa (M)	Valor
0	1111 1111	010 0000 0000 0000 0000 0000	NaN
1	1111 1111	010 0000 0000 0000 0000 0000	NaN

Soma e Subtração com Ponto Flutuante

Para SOMA e SUBTRAÇÃO com números em ponto flutuante, os números devem ter o mesmo expoente (E).

A SOMA ou SUBTRAÇÃO é realizada sobre as mantissas (M).

O resultado é formado pela mantissa resultante e o expoente dos operandos.

Para se ter expoentes iguais, o menor dos expoentes deve ser igualado ao maior, e a mantissa correspondente deve ser deslocada para a direita de forma que o número representado pelo par M,E não seja alterado.

Exemplo de soma com padrão IEEE 754

Somar os números 12,5 e 22,25 utilizando representação em ponto flutuante IEEE 754, com precisão SIMPLES (32 bits):

1º passo: transformar 12,5 em binário, para obter a mantissa (normalizada) e o expoente.

12,5 / 2 = 6,25 6,25 / 2 = 3,125 3,125 / 2 = 1,5625 => 2^3 0,5625 × 2 = **1**,125 0,125 × 2 = **0**,25 0,25 × 2 = **0**,50 0,50 × 2 = **1**
$12,5 = (1,1001)_2 \times 2^3$ Normalizando: $12,5 = (0,11001)_2 \times 2^4$ Mantissa = 1100 1000 0000 0000 0000 000 Preenche com "0" para completar 23 bits

2º passo: transformar 22,25 em binário para obter a mantissa (normalizada) e o expoente.

22,25 / 2 = 11,125

11,225 / 2 = 5,5625

5,6125 / 2 = 2,78125

2,80625 / 2 = 1,390625 $\Rightarrow 2^4$

0,390625 × 2 = **0**,78125

0,78125 × 2 = **1**,5625

0,5625 × 2 = **1**,125

0,125 × 2 = **0**,25

0,25 × 2 = **0**,5

0,50 × 2 = **1**

$22,25 = (1,011001)_2 \times 2^4$

Normalizando:

$22,25 = (0,1011001)_2 \times 2^5$

Mantissa =

1011 0010 0000 0000 0000 000

Preenche com “0” para completar 23 bits

3º passo: igualar o expoente menor ao maior.

$12{,}5 = (0{,}11001)_2 \times 2^4$

Expoente = 4

Igualando o expoente:

$12{,}5_{10} = (0{,}011001)_2 \times 2^5$

Como houve deslocamento do número, a mantissa também tem de ser deslocada.

Mantissa =

0110 0100 0000 0000 0000 000

$22{,}25 = (0{,}1011001)_2 \times 2^5$

Mantissa =

1011 0010 0000 0000 0000 000

4º passo: obter o expoente em excesso (para 32 bits o valor de excesso é 127).

$12{,}5_{10} = (0{,}011001)_2 \times 2^5$

expoente = **5** => 0000 0101

excesso (127) => + 0111 1111

expoente em excesso = 1000 0100 (132)

$22{,}25 = (0{,}1011001)_2 \times 2^5$

expoente = **5** => 0000 0101

excesso (127) => + 0111 1111

expoente em excesso = 1000 0100 (132)

5º passo: obter a representação final dos números.

$12{,}5_{10} = (0{,}011001)_2 \times 2^5$

Sinal = **0**

Expoente em excesso = **1000 0100**

Mantissa = 0110 0100 0000 0000 0000 000

$12{,}5_{10} =$ **0 1000 0011 0110 01**00 0000 0000 0000 000

$22{,}25 = (0{,}1011001)_2 \times 2^5$

Sinal = **0**

Expoente em excesso = **1000 0100**

Mantissa = 1011 0010 0000 0000 0000 000

$22{,}25_{10}$ = **0 1000 0011 1011 0010** 0000 0000 0000 000

6º Passo: somar as mantissas, ajustar o expoente se necessário e obter o resultado.

$12{,}5_{10}$ => Mantissa = 0110 0100 0000 0000 0000 000

$+22{,}25_{10}$ => Mantissa = +1011 0010 0000 0000 0000 000

= 1,0001 0110 0000 0000 0000 00~~00~~

= 1000 1011 0000 0000 0000 000

Como a mantissa foi deslocada para a direita, para normalizar o resultado, o expoente deve ser incrementado.

E = 5 + 1 = 6 => Expoente em excesso = 6 + 127 = 133 => E = 1000 0101

Resultado:

12,5 + 22,25 = **0 1000 0101 1000 1011 0000 0000 0000 000**

S E M

Vamos conferir!

Lembrando que $12{,}5 + 22{,}25 = (-1)^S \times (M) \times 2^E$

Resultado:

12,5 + 22,25 =**0 1000 0100 1000 1011 0000 0000 0000 000**

S E M

Vamos converter a mantissa obtida em decimal:

= **1000 1011 0000 0000 0000 000**

$= (1 \times 2^{-1}) + (1 \times 2^{-5}) + (1 \times 2^{-7}) + (1 \times 2^{-8})$

$= 0{,}5 + 0{,}03125 + 0{,}0078125 + 0{,}00390625$

= 0,54296875

Resultado = $(-1)^S \times (M) \times 2^E$

S = 0

E = 6

M = 0, 54296875

Resultado = $(-1)^0 \times (0,54296875) \times 2^6 = 0,54296875 \times 64 =$ **34,75**

2.4 Síntese da Unidade

A utilização de ponto fixo permite aos sistemas computacionais representar números positivos, negativos e de maior magnitude. A necessidade de cálculos mais precisos, envolvendo números de grande magnitude e com partes fracionárias, é atendida com o uso de números representados em ponto flutuante. O entendimento destes conceitos é importante para a definição mais adequada de sistemas computacionais em atendimento a necessidades de diferentes tipos de operação matemática.

2.5 Para Saber Mais

Vídeo:

ERROS - Representação de Números e Aritmética de Ponto Flutuante - Representação de Números, 19 ago. 2018. 1 vídeo (9min16s). Publicado por Responde Aí. Disponível em: https://youtu.be/YHquEw23qRo. Acesso em: 28 set. 2021.

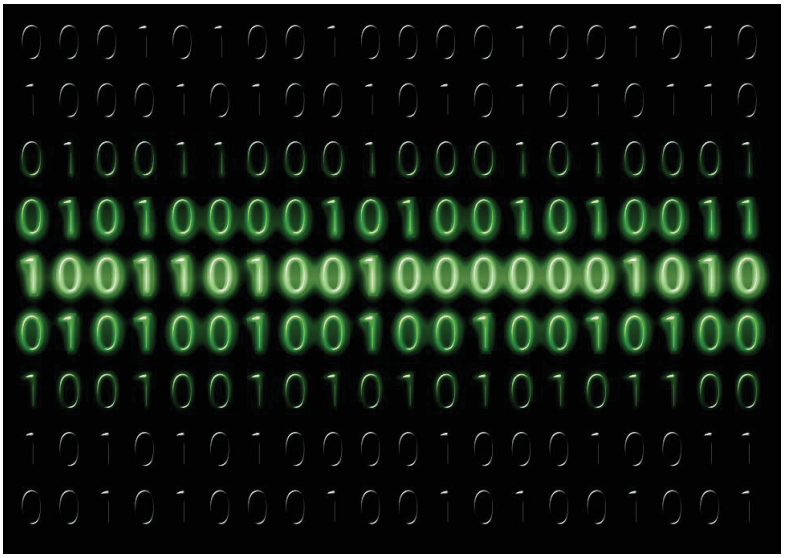
REPRESENTAÇÃO Numérica - Números Inteiros e Ponto Fixo (Eletrônica Digital - 06), 30 mar. 2020. 1 vídeo (11min04s). Publicado por Prof. Edil. Disponível em: https://youtu.be/7i3shg5Ss90. Acesso em: 27 set. 2021.

Unidade III

Representações de dados alfanuméricos utilizadas para diferentes modelos de escrita e comandos de sistemas computacionais; Álgebra de Boole

UNIDADE III

Introdução



Fonte: Freepik

Nesta Unidade, são abordados diferentes modelos de representações alfanuméricas, que possibilitam representar diferentes modelos de escrita, símbolos e também comandos utilizados em sistemas de computação. Também será apresentada a Álgebra de Boole, com seus postulados, identidades e propriedades, que são a base de funcionamento de circuitos lógicos digitais que compõem os sistemas computacionais.

3.1 Representação de dados alfanuméricos

Computadores e sistemas digitais podem ter a necessidade de representar dados numéricos ou alfanuméricos.

Conforme visto nas unidades anteriores, para representar dados numéricos os sistemas digitais utilizam a notação em “0”s e “1”s, formatada em números de representação binária (Base 2), números em Hexadecimal (representação em Base 16), e outros.

De forma a dar maior cobertura de representação numérica, modelos em ponto fixo e ponto flutuante são utilizados em sistemas computacionais, porém para a representação de dados alfanuméricos, que dizem respeito a sistemas de escrita, a padronização proposta por entidades normativas e mesmo por diferentes fabricantes utiliza outros padrões.

Neste sentido, fabricantes, entidades normativas e outros desenvolveram padrões de representação de dados alfanuméricos com o objetivo de permitir a compatibilidade de escrita em diferentes sistemas computacionais, de diferentes fabricantes.

Os primeiros códigos de representação se mostraram bastante eficientes e atingiram seu objetivo (de proporcionar compatibilidade entre diferentes sistemas computacionais) rapidamente, porém também rapidamente as limitações apareceram, e a principal foi a não representação de toda a gama de sistemas de escrita que existem no mundo, notadamente sistemas de escrita asiáticos, que exigem uma simbologia totalmente diferente da usada no sistema latino de escrita.

Neste sentido, dados alfanuméricos são utilizados para representar letras, números, caracteres especiais, comandos etc.

Os primeiros códigos alfanuméricos utilizavam 6 bits, e com isso podiam representar 2^6 (64) diferentes caracteres ou comandos especiais.

No início dos anos 1960 (1963), foi publicado o Código Padrão Americano para Intercâmbio de Informação, em inglês American Standard Code for Information Interchange – ASCII.

A ideia básica foi representar com 7 bits o conjunto de caracteres do alfabeto e os comandos utilizados em sistemas de computação.

Código ASCII (7 bits): Caracteres de Controle (sinais de controle, não imprimíveis):

Representação			Abreviação	Nome	Descrição
Binário (7 bits)	Decimal	Hexadecimal			
0000 0000	00	00	NULL	Null	Caractere Nulo

Representação			Abreviação	Nome	Descrição
Binário (7 bits)	Decimal	Hexadecimal			
0000 0001	01	01	SOH	Start of Header	Início de cabeçalho
0000 0010	02	02	STX	Start of Text	Início de texto
0000 0011	03	03	ETX	End of Text	Fim de texto
0000 0100	04	04	EOT	End of Transmission	Fim de transmissão
0000 0101	05	05	ENQ	Enquiry	Consulta; Solicitação de transmissão
0000 0110	06	06	ACK	Acknowledge	Reconhecimento de transmissão; Confirmação
0000 0111	07	07	BEL	Bell	Campainha; Sinal sonoro
0000 1000	08	08	BS	Back-space	Espaço atrás; Retorno de 1 caractere
0000 1001	09	09	HT	Horizontal Tabulation	Tabulação horizontal
0000 1010	10	0A	LF	Line Feed	Alimentação de linha; Mudança de linha; Nova linha
0000 1011	11	0B	VT	Vertical Tabulation	Tabulação vertical
0000 1100	12	0C	FF	Form Feed	Alimentação de formulário
0000 1101	13	0D	CR	Carriage Return	Retorno do carro; Retorno ao início da linha
0000 1110	14	0E	SO	Shift Out	Mover para fora; Deslocamento para fora; Desligar deslocador de bits
0000 1111	15	0F	SI	Shift In	Mover para dentro; Deslocamento para dentro; Acionar deslocador de bits

Representação			Abreviação	Nome	Descrição
Binário (7 bits)	Decimal	Hexadecimal			
0001 0000	16	10	DLE	Data-Link Escape	Escape do link de dados; Escape de conexão
0001 0001	17	11	DC1	Device Control 1	Controle de dispositivo 1
0001 0010	18	12	DC2	Device Control 2	Controle de dispositivo 2
0001 0011	19	13	DC3	Device Control 3	Controle de dispositivo 3
0001 0100	20	14	DC4	Device Control 4	Controle de dispositivo 4
0001 0101	21	15	NAK	Negative-Acknowledge	Confirmação negativa
0001 0110	22	16	SYN	Synchronous Idle	Estado ocioso síncrono; Espera síncrona
0001 0111	23	17	ETB	End of Transmission Block	Bloco de fim de transmissão
0001 1000	24	18	CAN	Cancel	Cancelar
0001 1001	25	19	EM	End of Medium	Fim de mídia; Fim do meio
0001 1010	26	1A	SUB	Substitute	Substituir
0001 1011	27	1B	ESC	Escape	Escapar
0001 1100	28	1C	FS	File Separator	Separador de arquivos
0001 1101	29	1D	GS	Group Separator	Separador de grupos
0001 1110	30	1E	RS	Record Separator	Separador de registros
0001 1111	31	1F	US	Unit Separator	Separador de unidades
0111 1111	127	7F	DEL	Delete	Deletar

Código ASCII (7 bits): Caracteres Gráficos (sinais gráficos, imprimíveis):

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0010 0000	32	20	(espaço)
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q

Representação			Sinal
Binário (7 bits)	Decimal	Hexadecimal	
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

Em muitos casos, o 8º bit no código ASCII é utilizado de maneira “customizada”, ou seja, de maneira particular em cada fabricante/desenvolvedor.

Pode ser usado como bit de “paridade”, para verificação de erro, ou para definir alguma extensão do próprio ASCII.

Outros códigos de representação de caracteres foram apresentados por diferentes fabricantes de computadores; o que mais foi disseminado foi o código EBCDIC: Extended Binary Code Decimal Interchange Code, proposto pela IBM (International Business Machine Corp), por volta de 1963~1964.

Tanto o ASCII como o EBCDIC apresentam limitação para suportar múltiplos sistemas de escrita, notadamente os sistemas asiáticos.

A ISO (International Organization fo Standardization) definiu a ISO 8859, com representação dos caracteres ASCII de forma estendida, incluindo caracteres acentuados. Mesmo a ISO 8859 não apresentou condições de representar os diversos sistemas de escrita existentes.

Com isso, novos códigos foram desenvolvidos, notadamente o alfabeto UNICODE, cuja gestão é de responsabilidade do UNICODE Consortium. O código foi publicado no livro “The Unicode Standard”, e propõe um padrão que permite aos computadores representar e manipular texto de qualquer sistema de escrita existente.

Os 128 primeiros códigos do UNICODE, por exemplo, correspondem aos caracteres ASCII, porém são representados em hexadecimal.

Em tempo: fazem parte do Unicode Consortium as grandes empresas de software e hardware que se beneficiam da padronização de processamento de texto, dentre elas: IBM, Microsoft, Xerox, Adobe, Apple, Google, HP e muitas outras.

O alfabeto Unicode tem mais de 1 milhão de caracteres, e seriam necessários pelo menos 3 bytes (24 bits) para representar cada caractere em número binário.

A forma de simplificar e deixar mais eficiente a representação foi utilizar um código multibyte, onde alguns caracteres usam 1 byte, outros usam 2, outros 3 bytes, e assim por diante.

O código multibyte mais conhecido é o UTF-8, que usa de 1 a 4 bytes.

Os primeiros 256 códigos Unicode são idênticos aos do padrão ISO 8859-1, facilitando a conversão de textos em linguagens ocidentais.

A representação em Unicode UTF-8 (Unicode Transformation Format – 8) é feita acrescentando o prefixo “U+” a cada número, por exemplo:

Número Unicode	Caractere
U+0041	A
U+0042	B
U+0061	a
U+0062	b
U+007E	~
U+00C0	À
U+00E3	ã
U+00E7	ç
U+00E9	é
U+03B1	α
U+2014	—
U+201C	“

No site a seguir é possível encontrar todos os códigos Unicode.

https://unicode-table.com/pt/

3.2 Álgebra de Boole

Fonte: Freepik

George Boole (1815~1864), em meados do século 19, desenvolveu uma teoria baseada em postulados e operações lógicas simples para possibilitar a solução de vários problemas.

Essa teoria é chamada de "Álgebra de Boole", e estabelece que só existem duas condições possíveis ou estados aplicáveis a situações que se apresentem, observando que estes estados são opostos. Desta forma, de acordo com a Álgebra de Boole, variáveis lógicas podem assumir dois estados somente:

- 0 ou 1
- Verdadeiro ou Falso
- Aberto ou Fechado
- Ligado ou Desligado
- Alto ou Baixo

Na álgebra tradicional, as variáveis representam números reais e quando operadores são aplicados a estas variáveis o resultado é um número real.

Na Álgebra Booleana, as operações envolvendo as variáveis “0” e “1” têm como resultado “0”s e “1”s. Boole definiu então POSTULADOS, IDENTIDADES BÁSICAS e PROPRIEDADES com relação a variáveis lógicas, que regem as operações envolvendo as variáveis “0” e “1".

POSTULADOS:

Postulado da Complementação:

se A = 0 então $\overline{A} = 1$

se A = 1 então $\overline{A} = 0$

Postulado da Adição:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 1

Postulado da Multiplicação:

0 • 0 = 0

0 • 1 = 0

1 • 0 = 0

1 • 1 = 1

IDENTIDADES BÁSICAS:

Complementação:

$\overline{\overline{A}} = A$

se A = 0 então $\overline{A} = 1$

se A = 1 então $\overline{A} = 0$

se A = 0 então $\overline{\overline{A}} = 0$

se A = 1 então $\overline{\overline{A}} = 1$

Adição:

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + \overline{A} = 1$

Multiplicação:

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A \cdot A = A$

$A \cdot \overline{A} = 0$

PROPRIEDADES:

Comutativa na adição e multiplicação:

$A + B = B + A$

$A \cdot B = B \cdot A$

Associativa na adição e multiplicação:

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

$A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$

Distributiva:

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

IDENTIDADES AUXILIARES:

$A + A \cdot B = A$

$A + \overline{A} \cdot B = A + B$

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

TEOREMAS DE MORGAN (1806~1871)

Os teoremas do matemático De Morgan definem regras usadas para converter operações lógicas “OU” (OR) em “E” (AND) e vice-versa, procurando simplificar expressões em álgebra booleana.

Os Teoremas de De Morgan declaram:

1º Teorema:

A soma de “n” variáveis todas negadas (ou invertidas) é igual ao produto das “n” variáveis negadas (ou invertidas) individualmente.

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

2º Teorema:

O produto de “n” variáveis todas negadas (ou invertidas) é igual à soma da “n” variáveis negadas (ou invertidas) individualmente.

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

3.3 Síntese da Unidade

Nesta Unidade, foram apresentados os códigos de representação de dados alfanuméricos, importantes para representar o alfabeto e permitir interoperabilidade entre diferentes sistemas computacionais ao trocar informações envolvendo textos, planilhas e outros documentos. O uso de códigos que sejam comuns permite, além de troca de informações, o uso de formatação compatível também de comandos e principais ações no uso de sistemas digitais, inclusive considerando diferentes línguas. Foi também apresentada a Álgebra Booleana, com os conceitos fundamentais para a compreensão de circuitos digitais, circuitos lógicos e lógica combinacional, que são o núcleo das operações em sistemas computacionais.

3.4 Para Saber Mais

Vídeo:

ME SALVA! SCC01 - Álgebra Booleana - Circuitos Digitais, 15 mar. 2017. 1 vídeo (10min47s). Publicado por Me Salva! ENEM 2021. Disponível em: https://youtu.be/AvkzMOpmxgY. Acesso em: 20 out. 2021.

Livros:

IDOETA, V.; CAPUANO, F. G. **Elementos de Eletrônica Digital.** 42.ed. São Paulo: Érica, 2019.

Sites:

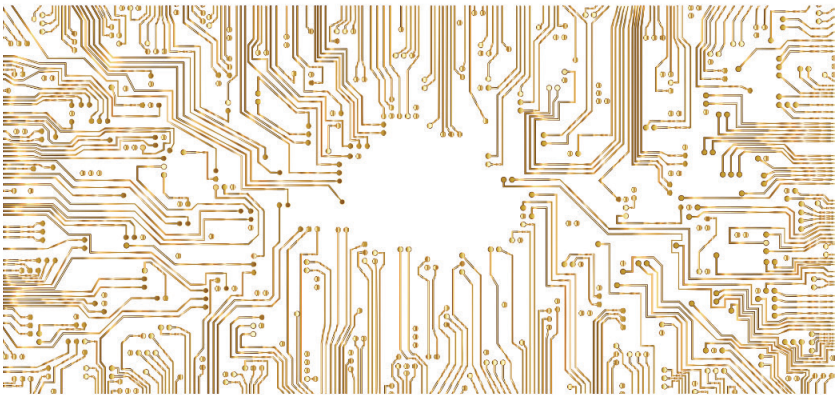
Acesse o link a seguir sobre álgebra de Boole e simplificação de circuitos lógicos.

Unidade IV

Blocos Lógicos e suas representações, Lógica Digital, Equivalência de Circuitos, Função Maioria de Três, Circuitos Lógicos Básicos, Circuitos Integrados, Circuitos Combinacionais

UNIDADE IV

Introdução



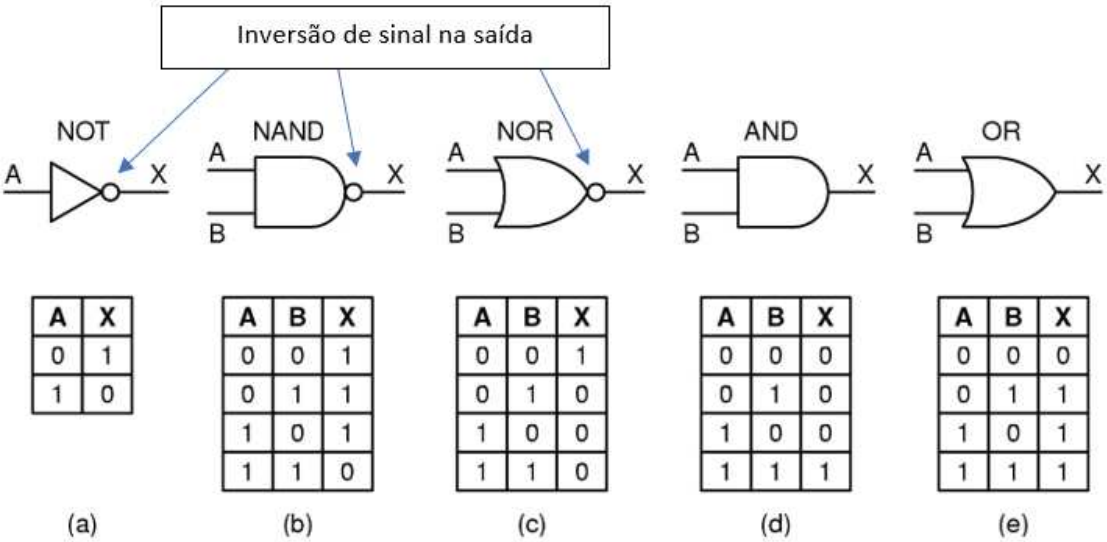
Fonte: Pixabay

Nesta Unidade, serão abordadas formas de representação de blocos lógicos, Portas OR, AND, NAND e XOR. Será, também, introduzido o conceito de lógica digital com combinação de diferentes portas lógicas para representar funções, dando como exemplo a “função maioria de três”. É apresentada a composição de circuitos lógicos básicos, com exemplo de circuito integrado e exemplos de circuitos combinacionais.

4.1 Representação de blocos lógicos

Blocos lógicos, conhecidos como portas lógicas, representam os postulados definidos na Álgebra de Boole.

O conjunto de blocos lógicos, suas representações e respetivas tabelas da verdade são dados por:



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(a) Porta NOT

Expressão Boolena: saída $X = \overline{A}$

Características:

A saída X corresponde à entrada invertida.

(b) Porta NAND

Expressão Boolena: saída $X = \overline{(A \bullet B)}$

Características:

Se uma das entradas estiver em estado lógico “0” (zero), a saída estará em estado lógico “1” (um), independentemente do estado lógico da outra entrada.

A saída somente estará em estado lógico “0” (zero) se ambas as entradas estiverem em estado lógico “1” (um).

(c) Porta NOR

Expressão Boolena: saída $X = \overline{(A + B)}$

Características:

Se uma das entradas estiver em estado lógico “1” (um), a saída estará em estado lógico “0” (zero), independentemente do estado lógico da outra entrada.

A saída somente estará em estado lógico “1” (um) se ambas as entradas estiverem em estado lógico “0” (zero).

(d) Porta AND

Expressão Boolena: saída X = (A • B)

Características:

Se uma das entradas estiver em estado lógico “0” (zero), a saída estará em estado lógico “0” (zero), independentemente do estado lógico da outra entrada.

A saída somente estará em estado lógico “1” (um) se as duas entradas estiverem em estado lógico “1” (um).

(e) Porta OR

Expressão Boolena: saída X = (A + B)

Características:

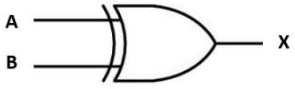
Se uma das entradas estiver em estado lógico “1” (um), a saída estará em estado lógico “1” (um), independentemente do estado lógico da outra entrada.

A saída somente estará em estado lógico “0” (zero) se ambas as entradas estiverem em estado lógico “0” (zero).

Além destas portas lógicas apresentadas há as seguintes portas:

(f) Porta XOR (Exclusive OR)

Porta EXCLUSIVE OR (XOR, OU Exclusivo)



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Expressão Boolena: saída X = (A $\oplus$ B) , ou A or-exclusive B

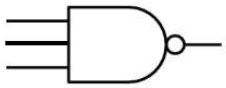
Características:

Se uma das entradas estiver em estado lógico “1” (um), a saída estará em estado lógico “1” (um) independentemente do estado lógico da outra entrada.

A saída somente estará em estado lógico “0” (zero) se ambas as entradas estiverem em estado lógico “0” (zero), ou se ambas estiverem em estado lógico “1”.

(g) Porta AND com 3 entradas

Porta NAND com 3 entradas (A, B, C)



A	B	C	Saída
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

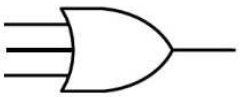
Expressão Boolena: saída X = (A • B • C)

Características:

Se todas as entradas estiverem em estado lógico “1” (um), a saída estará em estado lógico “0” (zero), caso contrário a saída será sempre “1”.

(h) Porta OR com 3 entradas

Porta OR com 3 entradas (A, B, C)



A	B	C	Saída
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

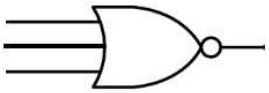
Expressão Boolena: saída X = (A + B + C)

Características:

Se uma ou mais entradas estiverem em estado lógico “1” (um), a saída estará em estado lógico “1” (um), caso todas as entradas estejam em estado lógico “0” (zero) a saída será “0”.

(i) Porta NOR com 3 entradas

Porta NOR com 3 entradas (A, B, C)



A	B	C	Saída
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Expressão Boolena: saída $X = \overline{(A + B + C)}$

Características:

Se uma ou mais entradas estiverem em estado lógico “1” (um), a saída estará em estado lógico “0” (zero), caso todas as entradas estejam em estado lógico “0” (zero) a saída ver “1”.

4.2 Lógica Digital, Equivalência de Circuitos, Função Maioria de Três

As portas lógicas são utilizadas para compor funções lógicas, resultado de combinação de diferentes estados de entrada de dados, seja para cálculos matemáticos, seja para controle em ambientes de automação.

Um exemplo de uso de portas lógicas é a implementação da função “maioria de três” como segue.

Função maioria de três:

- Dados 3 sinais de entrada (A, B e C)
- Se dois ou mais deles forem 1: a saída é 1
- Caso contrário: a saída é 0

Possibilidades corretas:

$$ABC,\ AB\bar{C},\ A\bar{B}C,\ \bar{A}BC$$

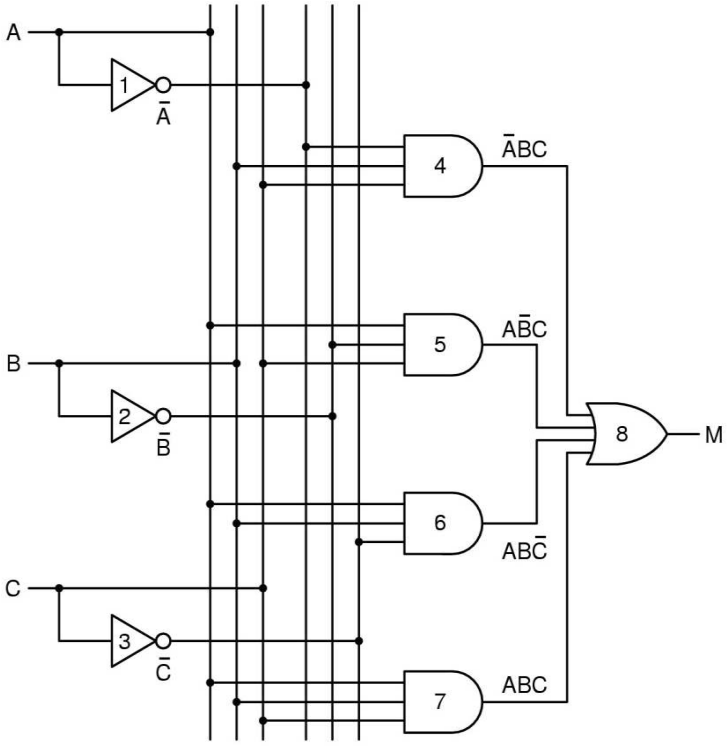
Representação matemática:

$$ABC + AB\bar{C} + A\bar{B}C + \bar{A}BC$$

Com isso, tem-se a seguinte tabela da verdade:

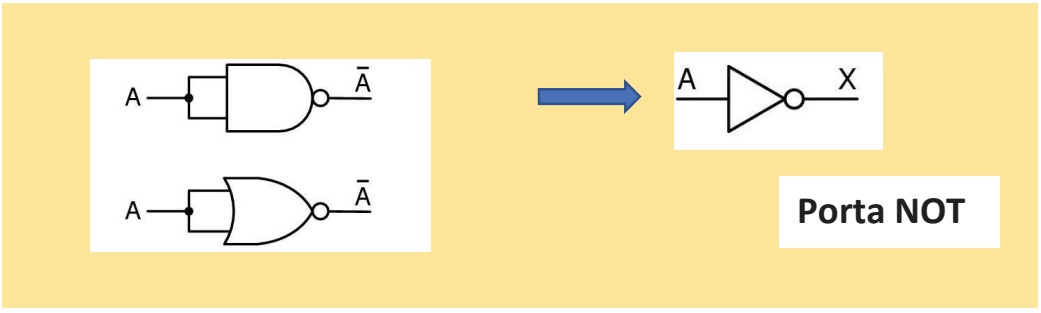
A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

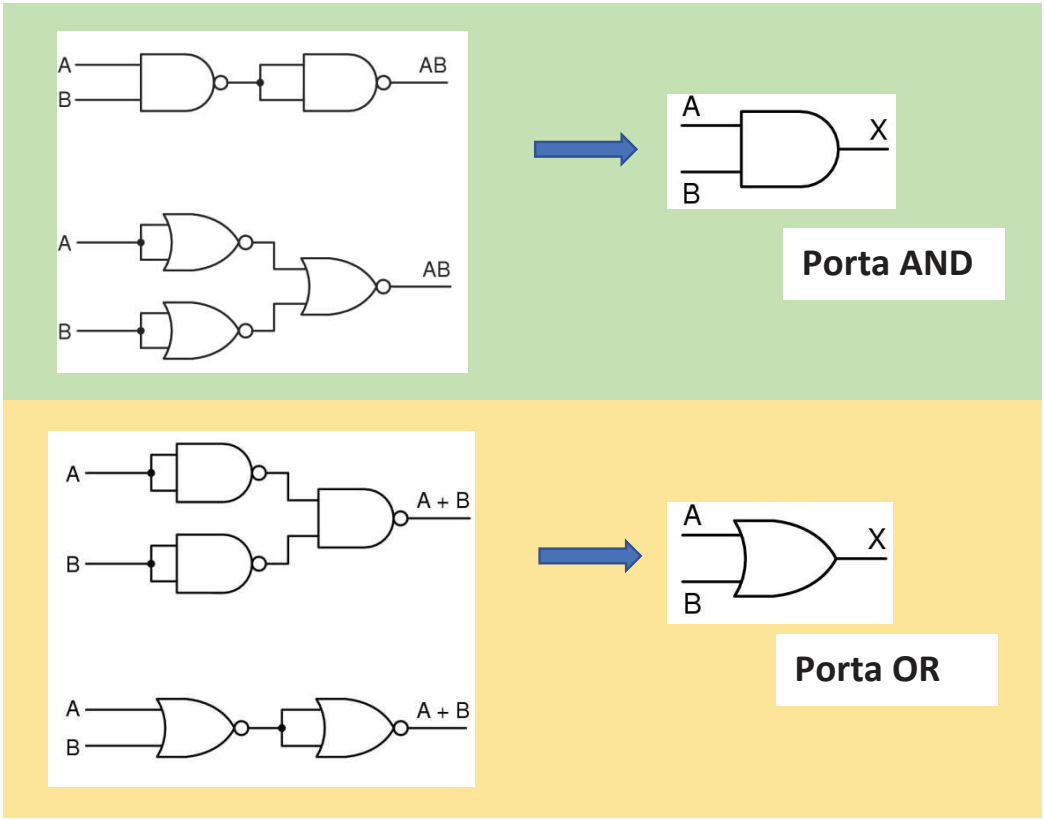
Que pode ser implementada com a seguinte combinação de portas lógicas:



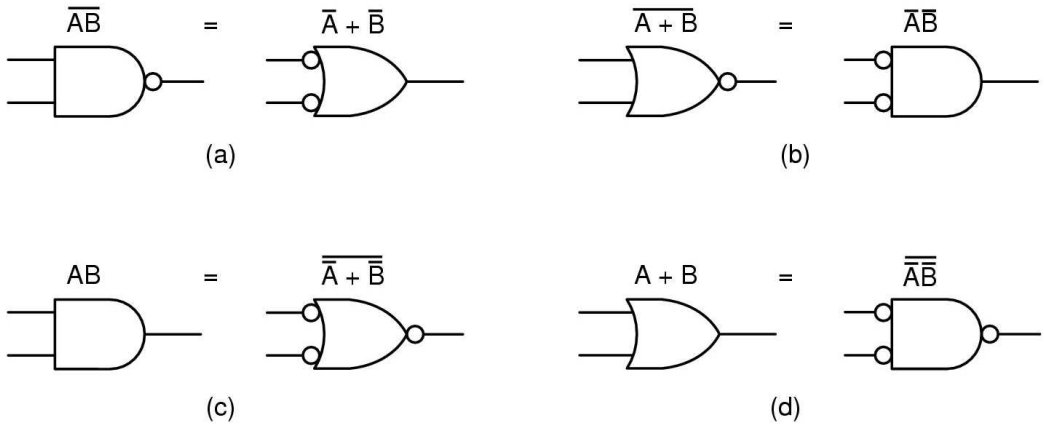
Da mesma forma, pode-se ter equivalência de portas lógicas a partir de portas NAND e NOR, por exemplo.

Construindo equivalências com portas NAND e NOR:



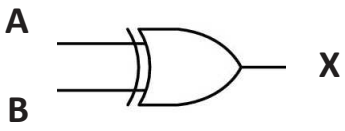


As portas lógicas podem também ter símbolos alternativos, como:

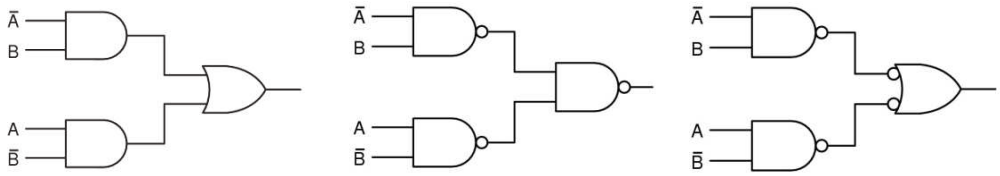


Símbolos alternativos: (a) NAND; (b) NOR; (c) AND; (d) OR

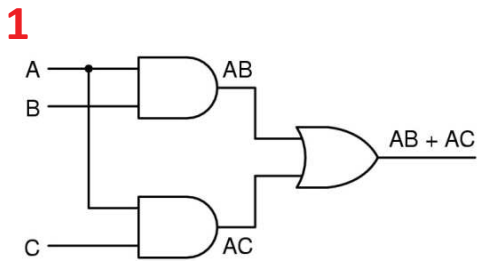
Uma porta lógica Exclusive OR (XOR) pode ser representada por uma combinação de portas lógicas, como a seguir, onde são utilizados três possíveis circuitos para representá-la.



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

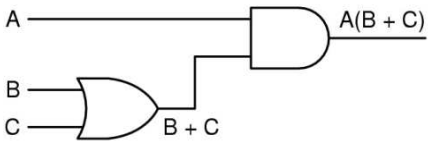


Outro exemplo seria a equivalência de circuitos com diferentes combinações de portas lógicas, como a seguir:



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

2



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

4.3 Circuitos Lógicos Básicos, Circuitos Integrados, Circuitos Combinacionais

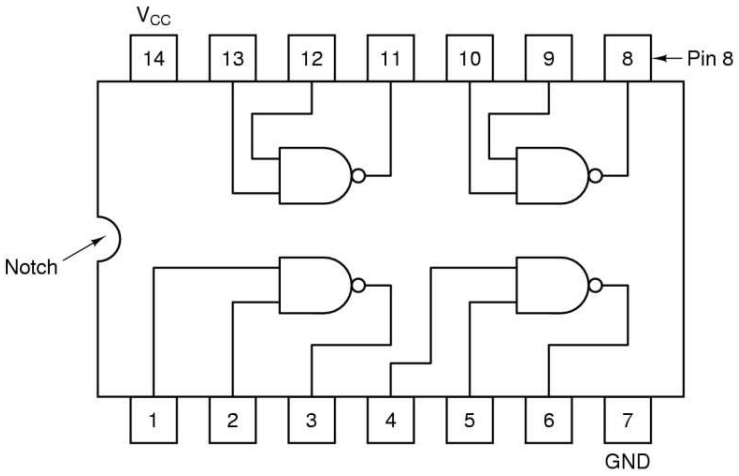
Conforme explicado, as portas lógicas, combinadas, são utilizadas para implementar diferentes funções em sistema digitais, e são agrupadas de forma a compor circuitos lógicos básicos e, em geral, da seguinte maneira:

- **Circuitos integrados:** um pedaço quadrado de silício no qual é colocado um conjunto de portas lógicas.
- **Circuitos combinacionais:** circuito com a característica de as saídas serem determinadas exclusivamente pelo valor presente nas respectivas entradas. Usuários devem configurar o chip.
- **Circuitos aritméticos:** circuito criado já com uma função específica implementada. Usuários não configuram a função do chip.
- **Clock:** circuito que emite uma série de pulsos do mesmo tamanho, a intervalos precisos entre pulsos consecutivos.

Os circuitos integrados, por exemplo, podem ser classificados de acordo com o número de portas lógicas que contêm, da seguinte maneira:

- Circuito SSI: de 1 a 10 portas lógicas
- Circuito MSI: de 10 a 100 portas lógicas
- Circuito LSI: de 100 a 100.000 portas lógicas
- Circuito VLSI: maior que 100.000 portas lógicas

Veja o exemplo a seguir:



Chip SSI com quatro portas lógicas NAND (7408)

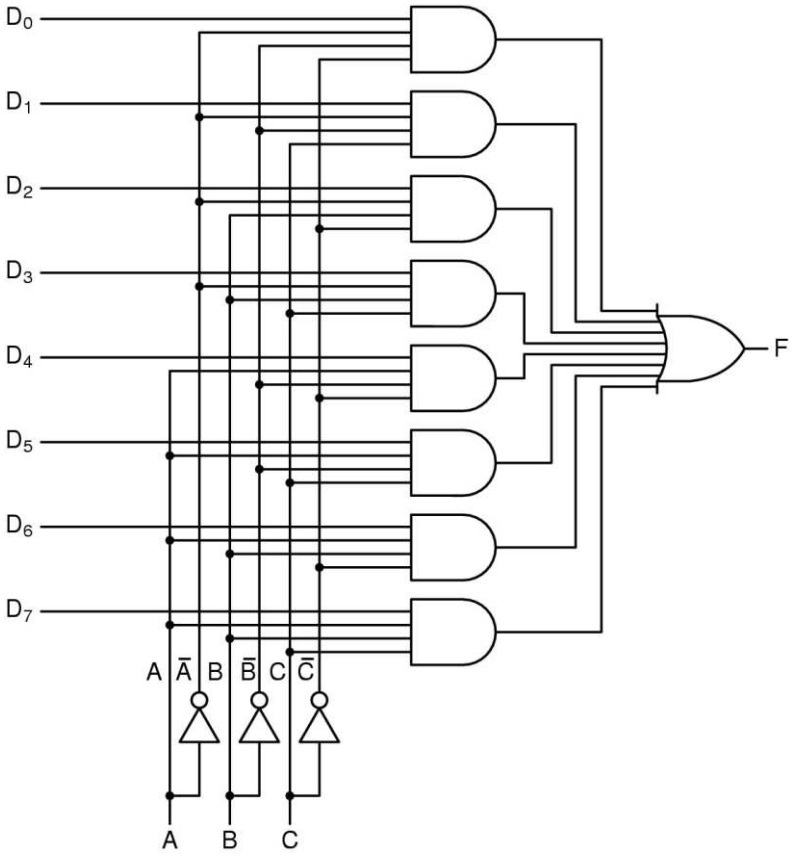
Os circuitos combinacionais podem ser divididos em:

- Multiplexadores
- Decodificadores
- Comparadores
- Matrizes Lógicas Programáveis

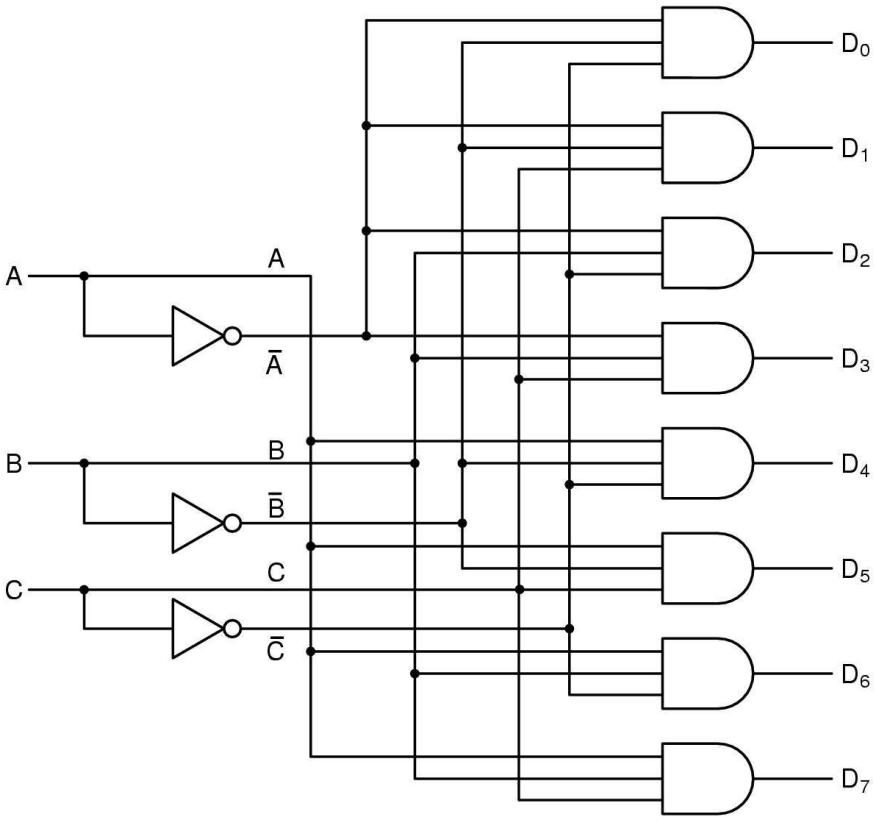
Onde:

Multiplexadores: circuito com n entradas de saída, 2n entradas de controle e uma saída de dados para efetuar a seleção de uma das entradas de dados.

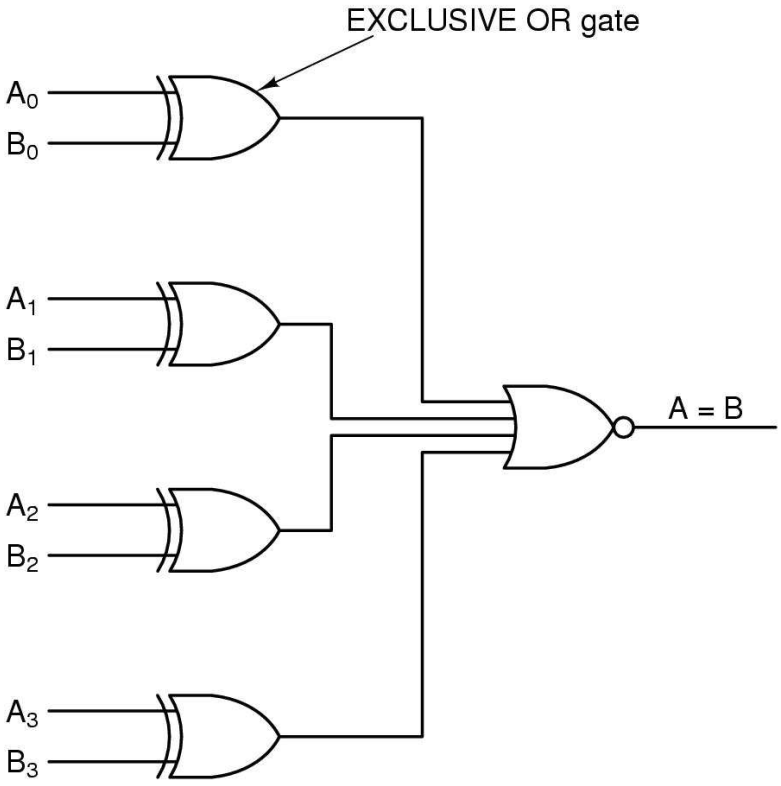
Segue o exemplo.



Decodificadores: recebe n bits de entrada que seleciona uma das 2^n saídas. Utilizado, por exemplo, para localizar chips de memória.
Segue exemplo com $n = 3$, ou seja, um decodificador 3 x 8



Comparadores: compara duas palavras que são entregues na entrada e produz 1 se as palavras são iguais e 0 caso contrário.
Segue exemplo de comparador de 2 palavras de 4 bits.

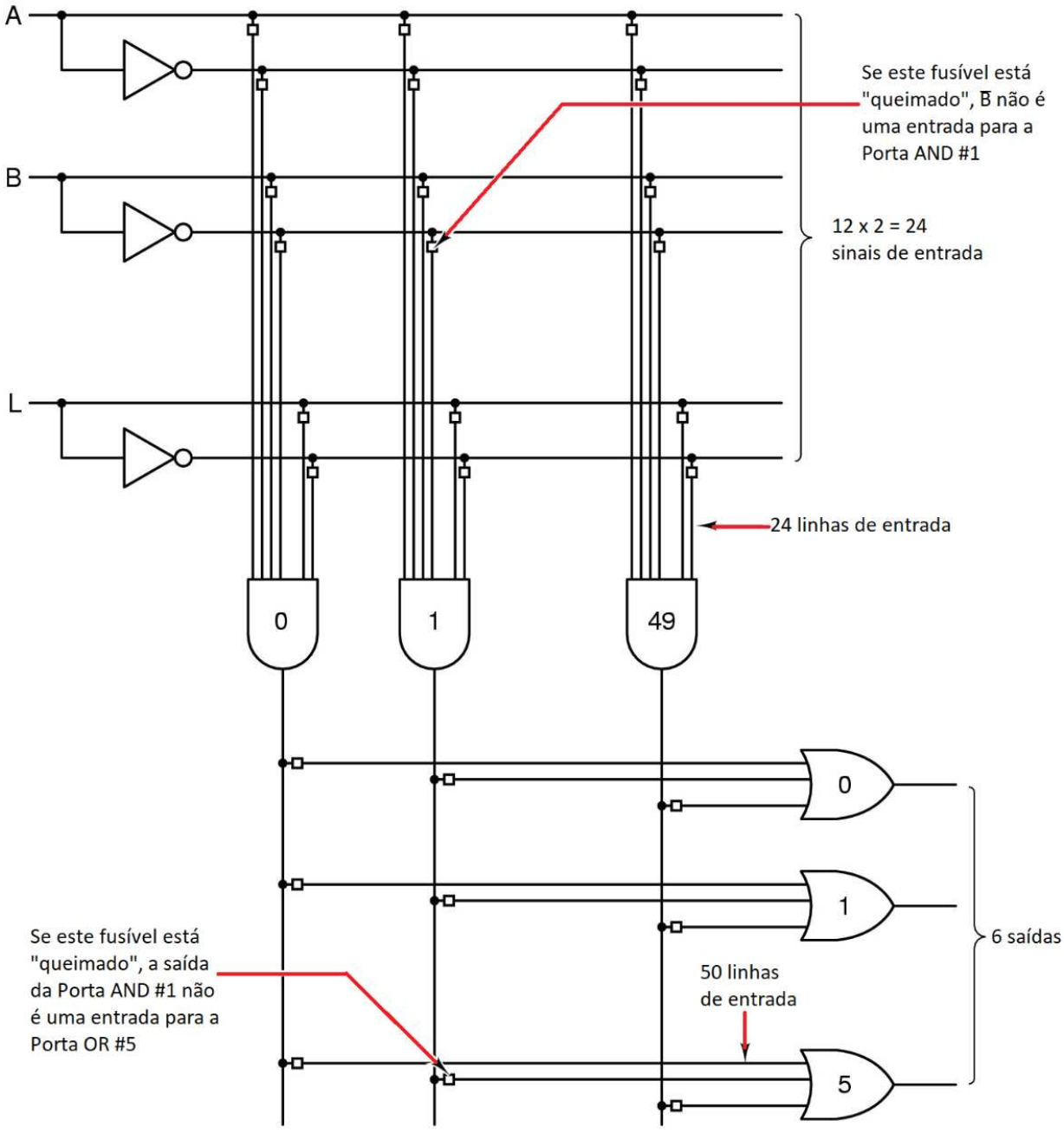


Matrizes Lógicas Programáveis: circuito combinacional genérico que permite a formação de somas de produtos.
Também chamado de **PLA** (*Programmable Logic Array*).

Exemplo: PLA de 12 entradas e 6 saídas, onde

- Complemento de cada entrada é gerado inteiramente, ou seja, para cada entrada tem-se o sinal X e o sinal $\overline{X}$;
- Com isso existem 24 sinais de entrada;
- Circuito possui 50 portas AND;
- Existem 1500 fusíveis intactos de fábrica;
- Programação feita com a queima dos fusíveis.

PLA de 12 entradas e 6 saídas



4.4 Síntese da Unidade

Nesta Unidade, foram apresentados os blocos lógicos, que representam diferentes portas lógicas, e também como a combinação destas portas lógicas pode ser utilizada para representar funções em sistemas digitais e sistemas computacionais, incluindo a abordagem sobre circuitos lógicos básicos (circuitos integrados e circuitos combinacionais).

4.4 Para Saber Mais

Vídeo

PORTAS LÓGICAS aprenda e não esqueça nunca mais (dica incrível), 06 fev. 2019. 1 vídeo (14min50s). Publicado por WR Kits. Disponível em: https://youtu.be/4tOpRh5o3QE. Acesso em: 20 out. 2021.

Livros

IDOETA, V.; CAPUANO, F. G. **Elementos de Eletrônica Digital.** 42.ed. São Paulo: Érica, 2019.

Sites

Acesse o link a seguir sobre funções lógicas e portas lógicas.

UNITAU digital

ISBN: 978-65-86914-51-1