



SÃO
PAULO
TECH
SCHOOL



Arquitetura de Computadores – Nível 2

Prof. Rogério Chola

rogerio.chola@sptech.school

Computadores: Linguagem e Pensamento

Um computador não entende a linguagem humana. Um computador somente entende, em sua lógica digital, dois estados possíveis: ligado (verdadeiro ou 1) e desligado (falso ou 0). Assim dizemos que um computador possui uma linguagem de máquina, formada por 0 e 1 e assim, por utilizar somente dois dígitos dizemos que o computador utiliza o Sistema Binário Digital onde cada símbolo (0 ou 1) é conhecido por BIT, do inglês Binary digit ou Dígito Binário. Dessa forma, um BIT pode valer 0 ou 1 e assim também concluímos que esse sistema possui Base 2 por ter somente 2 dígitos. Existem outros sistemas de numeração como o Decimal (Base 10); o Hexadecimal (Base 16) e o Octal (Base 8). Assim, é possível transformar ou converter qualquer **caractere** (letra; número; símbolo) em uma representação numérica em determinada BASE

Computadores: Linguagem e Pensamento

Sistema Base Binária: 2 dígitos (0 e 1)

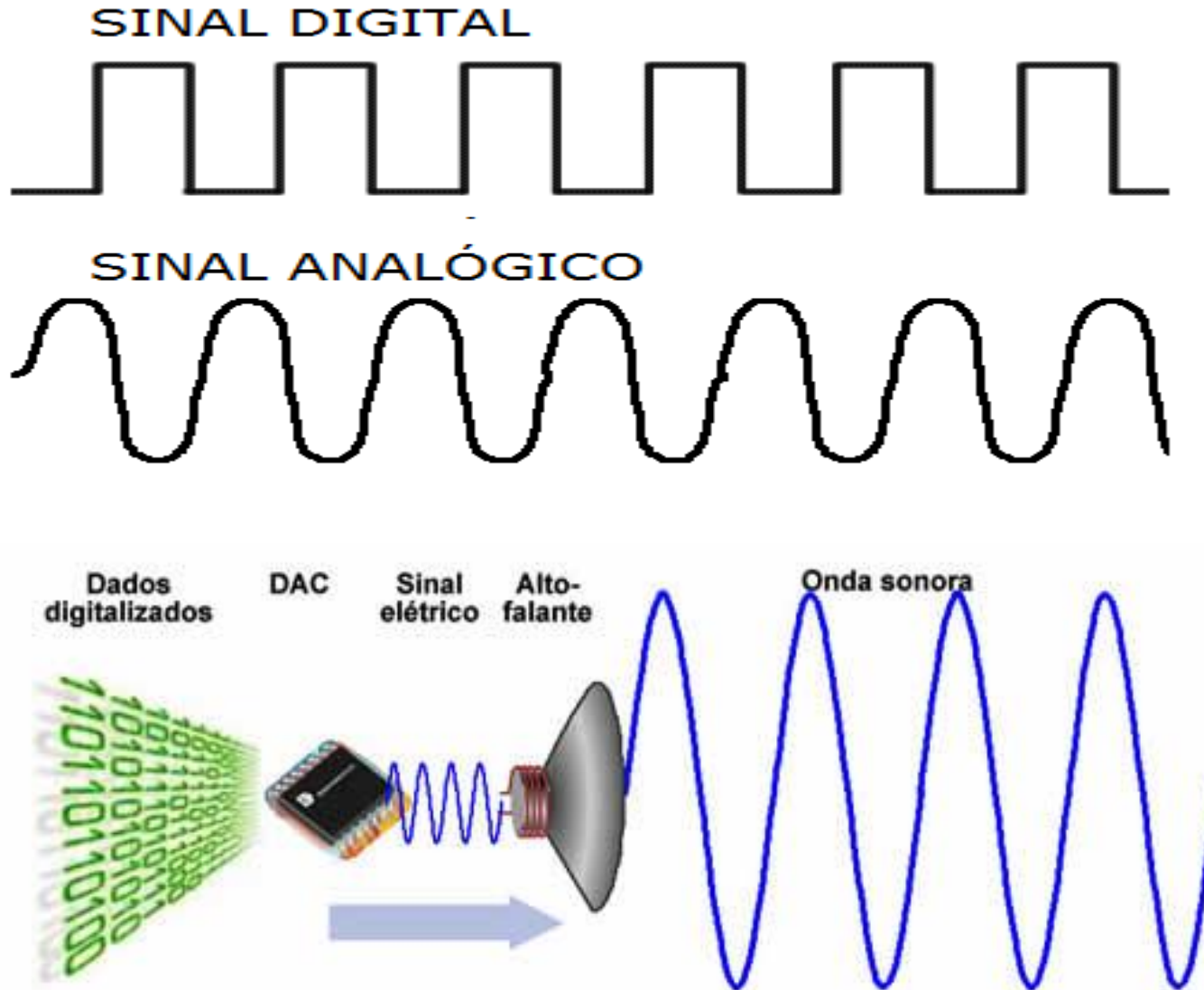
Sistema Base Decimal: 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 e 9)

Sistema Base Octal: 8 dígitos (0, 1, 2, 3, 4, 5, 6 e 7)

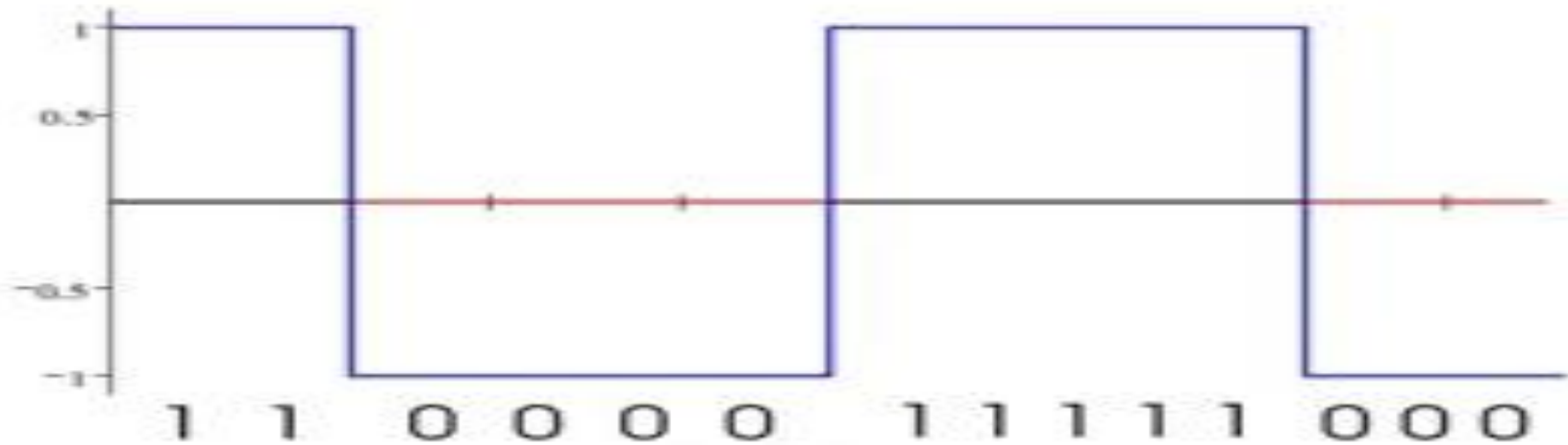
Sistema Base Hexadecimal: 16 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F)

É necessário um conversor Analógico-Digital para interagir os computadores com os humanos e vice-versa

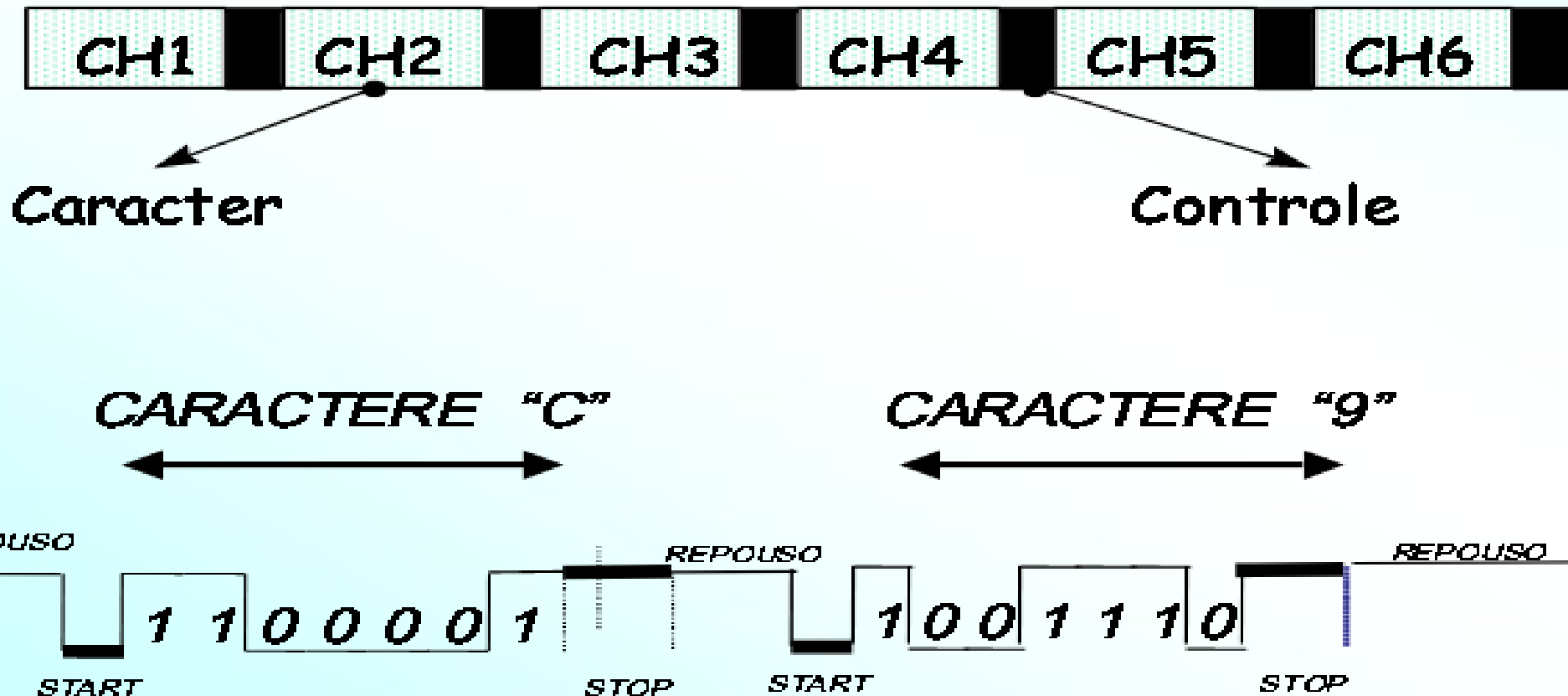
Computadores: Linguagem e Pensamento



Computadores: Linguagem e Pensamento



Computadores: Linguagem e Pensamento



Computadores: Linguagem e Pensamento

Obviamente que “falar” com um computador, bit a bit, leva tempo e gera complexidade. E também representar todos os caracteres possíveis somente utilizando dois valores não é possível. Dessa forma, o computador e seus circuitos foram “ensinados” a compor conjuntos de BIT's para assim formar padrões que podem representar, com mais facilidade todos os **caracteres** possíveis. Assim temos:

Nibble: formado por 4 bits ($2^4 = 16$)

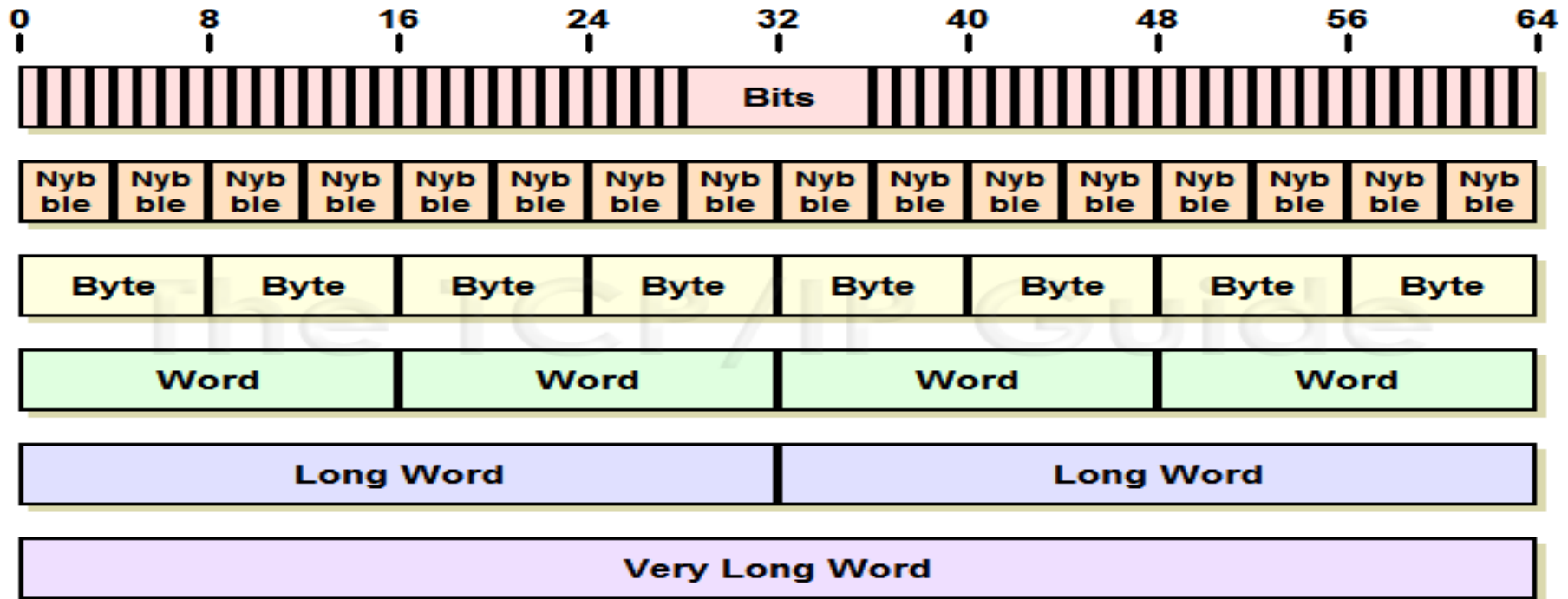
Byte: formado por 8 bits (octeto) ($2^8 = 256$)

Word: formado por 16 bits (double byte) ($2^{16} = 65.536$)

Double Word: formado por 32 bits (long word) ($2^{32} = 4.294.967.296$)

Quad Word: formado por 64 bits (very long word) ($2^{64} = 18.446.744.073.709.551.616$)

Computadores: Linguagem e Pensamento



Ano	Chip	Largura do barramento	Velocidade do clock	Transistores
1971	4004	4 bits	740KHz	2300
1974	8080	8 bits	2 MHz	6.000
1979	8088	16 bits	Até 8 MHz	29.000
1982	80286	16 bits	Até 12 MHz	134.000
1985	80386	32 bits	Até 33 MHz	275.000
1989	Intel 486	32 bits	Até 100 MHz	1.600.000
1993	Pentium (original)	64 bits	Até 200 MHz	3,3 milhões
1998	Pentium II	64 bits	233 MHz	7,5 milhões
	Pentium III	64 bits	Até 1 GHz	9,5 milhões
	Pentium IV	64 bits	Até 3,4 GHz	55,0 milhões

O Padrão ASCII

O nome ASCII vem do inglês American Standard Code for Information Interchange ou "Código Padrão Americano para o Intercâmbio de Informação". Ele é baseado no alfabeto romano e sua função é padronizar a forma como os computadores representam letras, números, acentos, sinais diversos e alguns códigos de controle

No ASCII existem apenas 95 caracteres que podem ser impressos, eles são numerados de 32 a 126 sendo os caracteres de 0 a 31 reservados para funções de controle. Ou seja, funções de computador. Alguns caracteres acabaram caindo em desuso pois eram funções específicas para computadores da época como o Teletype (máquinas de escrever eletro-mecânicas), fitas de papel perfurado e impressoras de cilindro. para computadores da época como o Teletype (máquinas de escrever eletro-mecânicas), fitas de papel perfurado e impressoras de cilindro

O Padrão ASCII

O ASCII é um código que foi proposto por Robert W. Bemer como uma solução para unificar a representação de caracteres alfanuméricos em computadores. Antes de 1960 cada computador utilizava uma regra diferente para representar estes caracteres e o código ASCII nasceu para se tornar comum entre todas as máquinas

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

O Padrão ASCII

Tabela ASCII – Binário, Decimal e Hexadecimal

Binário	D	H	G	Binário	D	H	G	Binário	D	H	G
0010 0000	32	20	vazio	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	\$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x
0011 1001	57	39	9	0101 1001	89	59	Y	0111 1001	121	79	y
0011 1010	58	3A	:	0101 1010	90	5A	Z	0111 1010	122	7A	z
0011 1011	59	3B	;	0101 1011	91	5B	[0111 1011	123	7B	{
0011 1100	60	3C	<	0101 1100	92	5C	\	0111 1100	124	7C	
0011 1101	61	3D	=	0101 1101	93	5D]	0111 1101	125	7D	}
0011 1110	62	3E	>	0101 1110	94	5E	^	0111 1110	126	7E	~
0011 1111	63	3F	?	0101 1111	95	5F	_				

O Padrão ASCII

Tabela ASCII –Decimal. Octal, HTML e Hexadecimal

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com

O Padrão ASCII

Tabela ASCII Extendida

128	Ç	144	É	160	á	176	░	192	Ł	208	⌌	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	±	209	⌍	225	β	241	±
130	é	146	Æ	162	ó	178	▓	194	⌐	210	⌎	226	Γ	242	≥
131	â	147	ø	163	ú	179		195	⌑	211	⌏	227	π	243	≤
132	ä	148	ö	164	ñ	180	⌈	196	—	212	⌐	228	Σ	244	∫
133	à	149	ò	165	Ñ	181	⌋	197	+	213	⌒	229	σ	245	∫
134	å	150	û	166	ª	182	⌌	198	⌔	214	⌓	230	μ	246	÷
135	ç	151	ù	167	º	183	⌍	199	⌕	215	⌔	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	⌎	200	⌕	216	⌕	232	Φ	248	°
137	ë	153	Ö	169	⌈	185	⌌	201	⌒	217	⌔	233	⊕	249	.
138	è	154	Ü	170	⌋	186	⌌	202	⌌	218	⌈	234	Ω	250	.
139	ï	155	•	171	½	187	⌍	203	⌐	219	■	235	δ	251	√
140	î	156	£	172	¾	188	⌌	204	⌑	220	■	236	∞	252	∞
141	ì	157	¥	173	¡	189	⌌	205	=	221	■	237	φ	253	²
142	Ä	158	£	174	«	190	⌋	206	⌑	222	■	238	ε	254	■
143	Å	159	ƒ	175	»	191	⌈	207	±	223	■	239	∩	255	

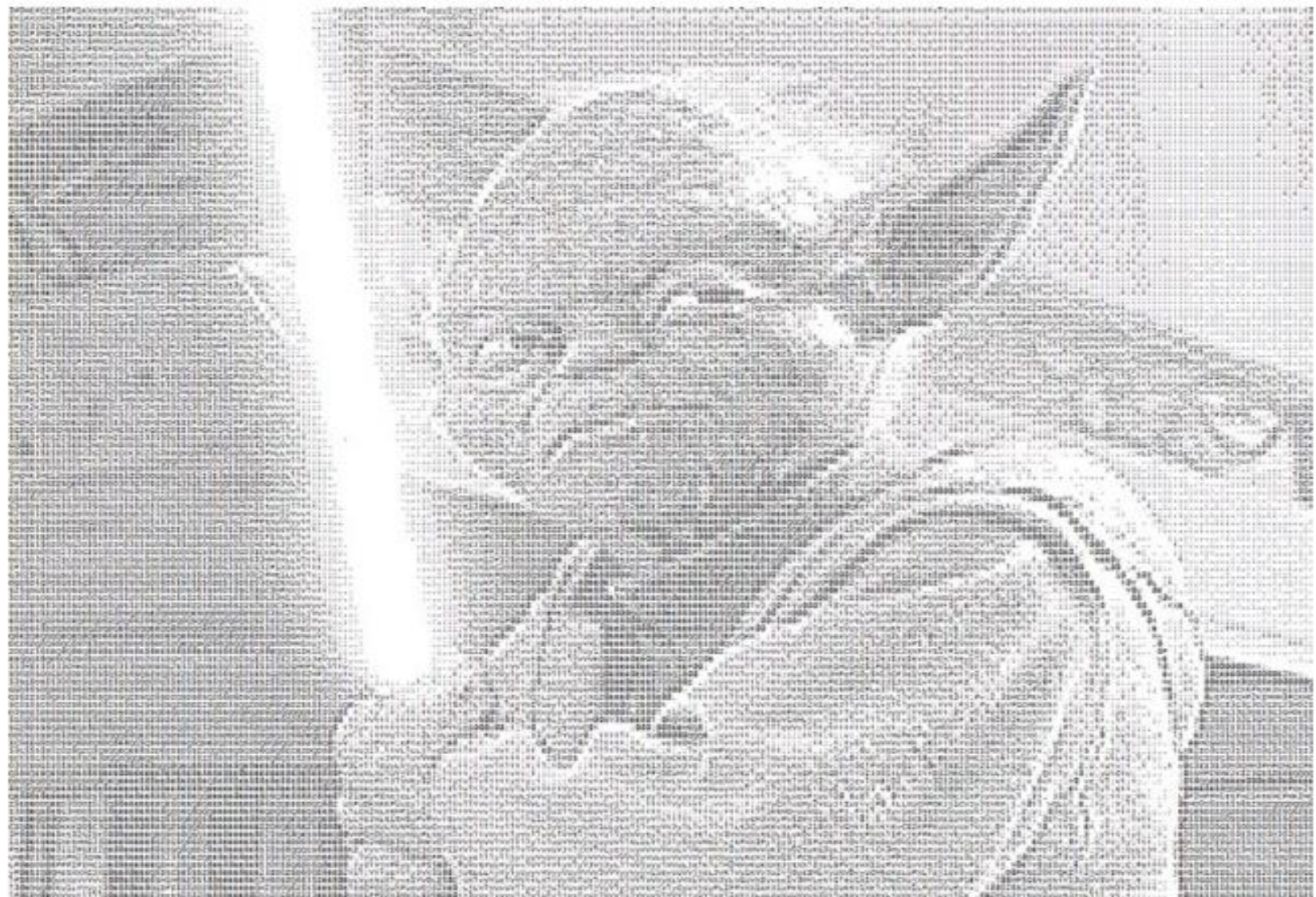
Source: www.LookupTables.com

Desenhando com ASCII

Outro uso bem interessante dos códigos ASCII é para a criação de desenhos. Os códigos podem ser utilizados para representar qualquer tipo de imagem, coloridas ou não. Quem utilizava os antigos canais de IRC (Internet Relay Chat) ou as BBS (Bulletin Board System) talvez se lembre disso. Caso você queira experimentar algumas possibilidades em código ASCII hoje existem soluções online como o conversor de textos para ASCII como o **Text to ASCII Art Generator** (<http://www.techtudo.com.br/tudo-sobre/text-ascii-art-generator.html>) ou o conversor de imagens Picascii (<http://www.techtudo.com.br/tudo-sobre/picascii.html>)

No site Asciiart (asciiarte.com) é possível conferir uma galeria de imagens criadas utilizando o código ASCII, apesar de ser um código antigo ele ainda é muito útil e divertido

Desenhando com ASCII



Exemplo de ASCII Art (Foto: Reprodução/André Sugai) — Foto: TechTudo

O Padrão ASCII

Tarefa:

Faça seu primeiro nome em ASCII nas representações em binário, hexadecimal, octal e decimal

Site para Referência: <https://www.asciitable.com/>

O Sistema Binário

Um sistema digital é um conjunto de funções usados para lidar com informações lógicas ou com quantidades físicas de forma digital. Para isso, é comum usarmos o sistema numérico binário, também conhecido como sistema de base 2, que consiste em dois possíveis valores de tensão, simbolizados pelos números 0 e 1, ou nível lógico baixo e nível lógico alto, respectivamente (Desligado/ligado; falso/verdadeiro)

Isso ocorre devido ao fato de que não seria viável projetar dispositivos eletrônicos capazes de operar com muitos níveis de tensão. Apesar disso, em algumas situações é necessário converter as saídas digitais binárias em valores com base decimal, como na utilização de calculadoras e computadores

A Álgebra Booleana

Nos circuitos lógicos, há uma predeterminação para a tensão, com valores de entrada e saída definidos. Portanto, para analisar e projetar determinados circuitos lógicos usamos a Álgebra Booleana, uma técnica que caracteriza as relações entre as entradas e saídas a partir de equações, também conhecidas como expressões booleanas

Com o uso da álgebra booleana, também é possível simplificar expressões de circuitos com a intenção de construir sistemas mais simples, com menos conexões e portas lógicas. Na álgebra booleana as variáveis assumem apenas valores de 0 ou 1, para representar os níveis de tensão, também conhecido como níveis lógicos

A Lógica Digital

As operações de um computador resumem-se então na combinação de operações aritméticas básicas: somar, complementar, comparar e mover bits

Essas complicadíssimas operações são realizadas por circuitos eletrônicos conhecidos como Circuitos Lógicos ou Logical Gates. Esses sistemas lógicos estão baseados na álgebra dos chaveamentos ou Álgebra de Boole, instituída pelo matemático inglês George Boole (1815 – 1864) e que admite apenas duas grandezas: falso ou verdadeiro, representados por 0 e 1 respectivamente

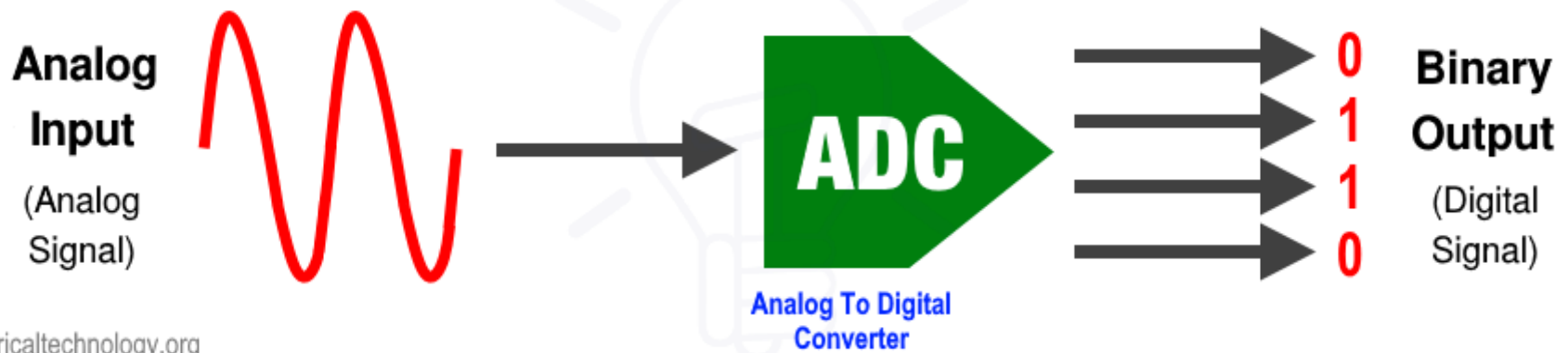
Esses sinais binários são representados por níveis de tensão nos circuitos do computador: 0 = Desligado ou 0 Volts e 1 = Ligado ou +5 Volts. A ação Analógica gera uma representação Digital

A Lógica Digital

ADC

ANALOG TO DIGITAL CONVERTER

Types, Working, Block Diagram & Applications



As Portas Lógicas Digitais

As portas lógicas são blocos fundamentais que combinados dão origem aos circuitos lógicos, a partir do qual é possível realizar operações. As portas lógicas podem possuir diversas variáveis de entrada, porém apenas uma variável de saída por vez

São 8 portas lógicas básicas: Buffer; NOT; AND; NAND; OR; NOR; XOR e XNOR

Cada porta tem uma função específica e uma simbologia e um relacionamento denominado de Tabela Verdade e é assim que os circuitos lógicos e os circuitos integrados são construídos. As portas lógicas funcionam como chaves que liga/desligam e roteiam os caminhos para a corrente eletrônica

A Tabela Verdade

A tabela-verdade define a relação de dependência da saída de um circuito lógico em relação aos níveis lógicos na sua entrada, representando todas as possíveis combinações

O número de combinações será igual a 2 elevado a N , para uma tabela-verdade de N variáveis de entrada

Por exemplo, para um circuito com duas variáveis de entrada, o número de saídas será igual a relação apresentada na imagem a seguir:

Número de saídas = 2 elevado a N

Número de saídas = 2^2

Número de saídas = 4


Uma tabela verdade de 2 entradas terá 4 saídas!

** N = Número de entradas*

A Tabela Verdade

Exemplo de Tabela Verdade básica

Entradas		Saída
A	B	X
0	0	1
0	1	0
1	0	1
1	1	0



A logic diagram to the right of the table shows two inputs, A and B, entering a rectangular box containing a question mark. An output line exits the box to the right, ending in a solid black circle labeled X. This diagram represents the logical function defined by the truth table.

Portas Lógicas Básicas



OR



NOR



AND



NAND



XOR



XNOR

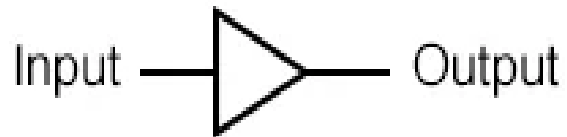


Buffer



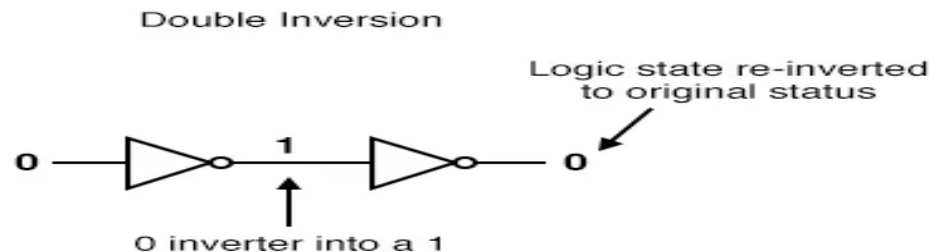
NOT

Porta Lógica BUFFER



Input	Output
0	0
1	1

Se conectarmos duas portas inversoras de modo que a saída de uma seja alimentada na entrada da outra, as duas funções de inversão se “cancelarão” uma à outra para que não haja inversão da entrada para a saída final:

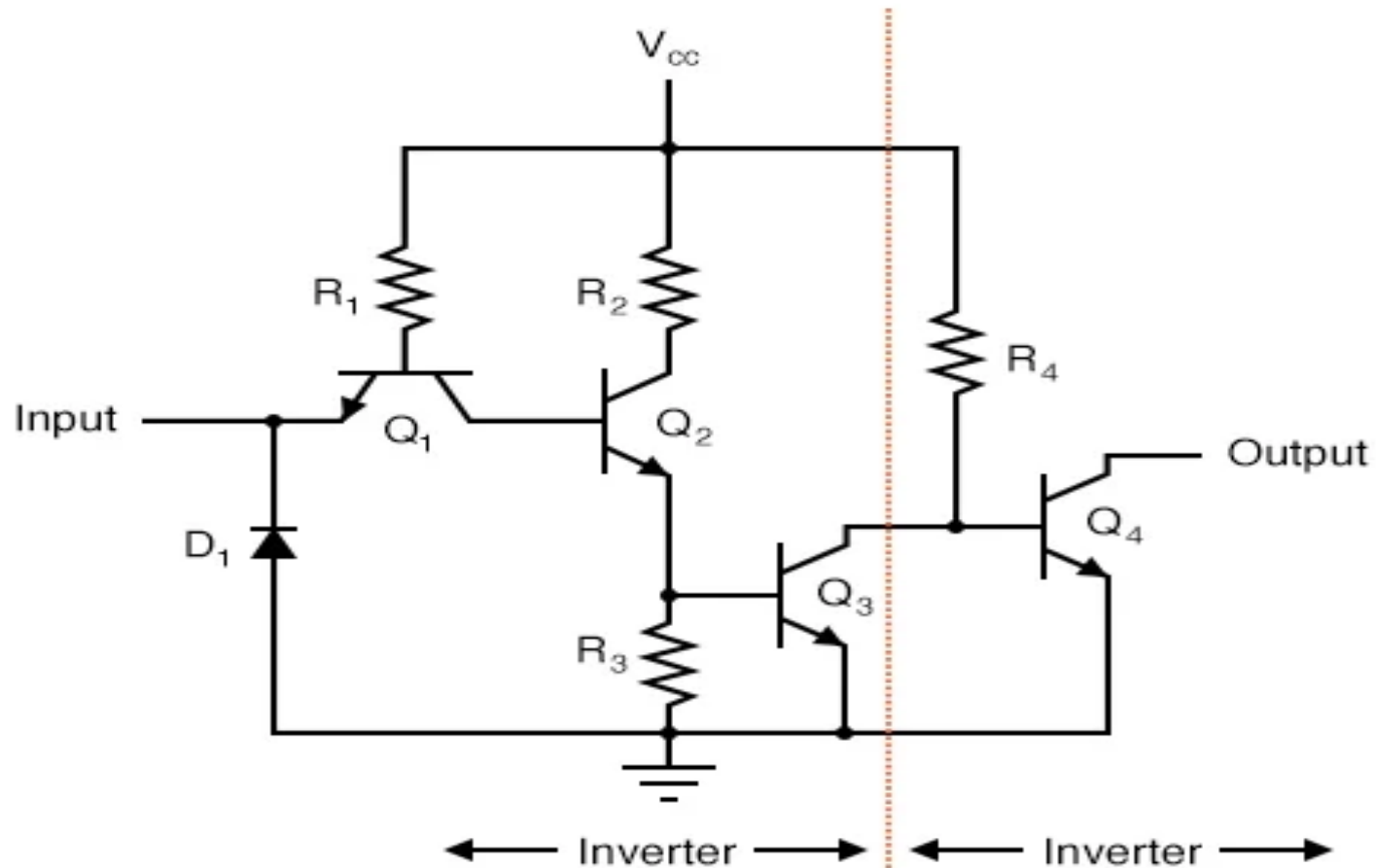


Embora isso possa parecer uma coisa inútil de se fazer, tem aplicação prática. Lembre-se de que os circuitos de porta são amplificadores de sinal, independentemente da função lógica que possam desempenhar.

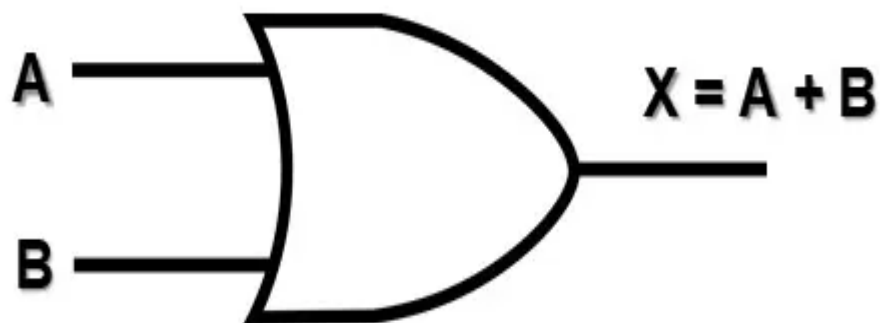
Uma fonte de sinal fraco (uma que não é capaz de fornecer ou transferir muita corrente para uma carga) pode ser reforçada por meio de dois inversores como o par mostrado na ilustração anterior. O nível lógico permanece inalterado, mas os recursos completos de fornecimento ou redução de corrente do inversor final estão disponíveis para acionar uma resistência de carga, se necessário. Para isso, uma porta lógica especial chamada buffer é fabricada para realizar a mesma função que dois inversores. Seu símbolo é simplesmente um triângulo, sem “bolha” invertida no terminal de saída

Porta Lógica BUFFER

Buffer Circuit with Open-Collector Output



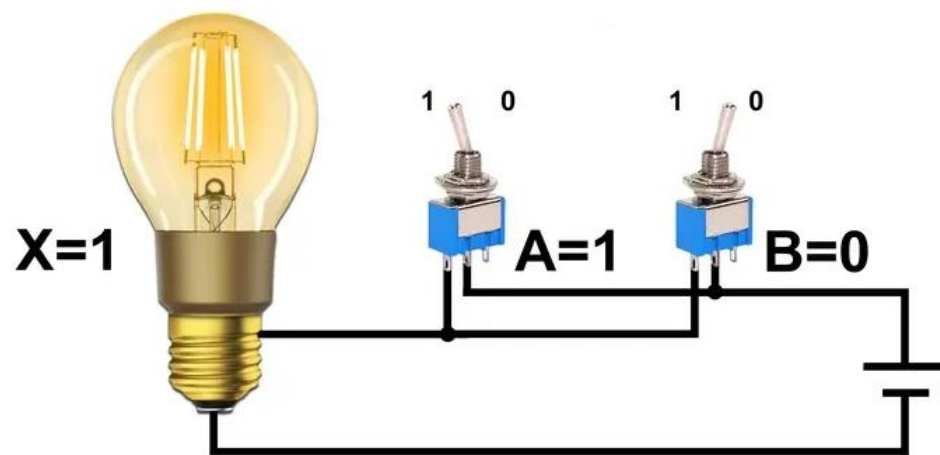
Porta Lógica OR



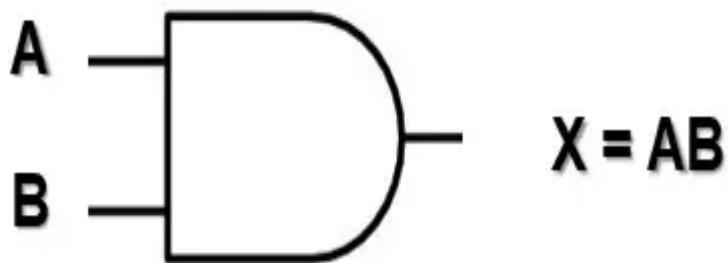
A	B	$X=A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Uma porta lógica **OR** corresponde a um circuito lógico capaz de realizar a operação booleana OR, operação esta que consiste em valores de saída com nível lógico alto sempre que qualquer uma das variáveis de entrada apresentar também nível lógico alto. O CI mais conhecido que é capaz de realizar essa operação é o 7432, que possui quatro portas OR de duas entradas cada

Nas figuras acima temos a simbologia e a expressão lógica booleana usadas para representar a porta OR. A saída da porta OR é igual à soma lógica das entradas! Por isso, ao representarmos a operação OR usamos o sinal de soma (+), e lemos a expressão como: “x é igual a A ou B” ou “x é igual a A or B”



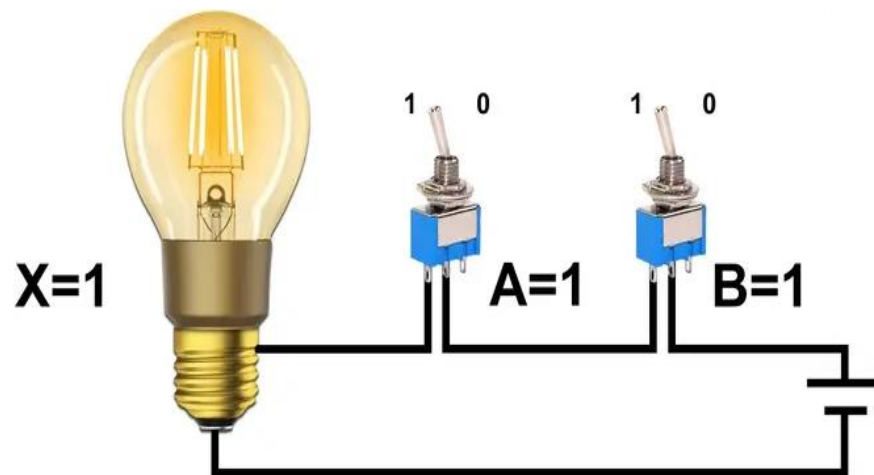
Porta Lógica AND



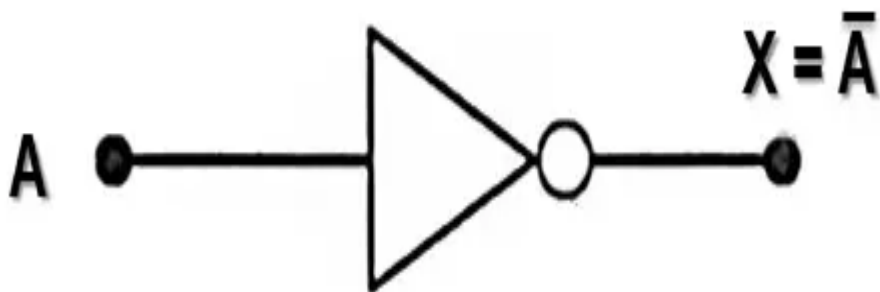
A	B	X=AB
0	0	0
0	1	0
1	0	0
1	1	1

A porta lógica **AND** corresponde a um circuito lógico capaz de realizar a operação booleana AND, que consiste em valores de saída com nível lógico baixo sempre que qualquer uma das variáveis de entrada apresentar também nível lógico baixo. O circuito integrado mais conhecido que consegue realizar essa operação lógica é o CI 7408, que possui quatro portas AND de duas entradas cada

Nas imagens vemos a simbologia e a expressão lógica booleana usadas para representar a porta AND. A saída da porta AND é igual ao produto lógico das entradas! Por isso, ao representarmos a operação AND usamos o sinal de multiplicação, e lemos a expressão como: “x é igual a A e B” ou “x é igual a A and B”



Porta Lógica NOT

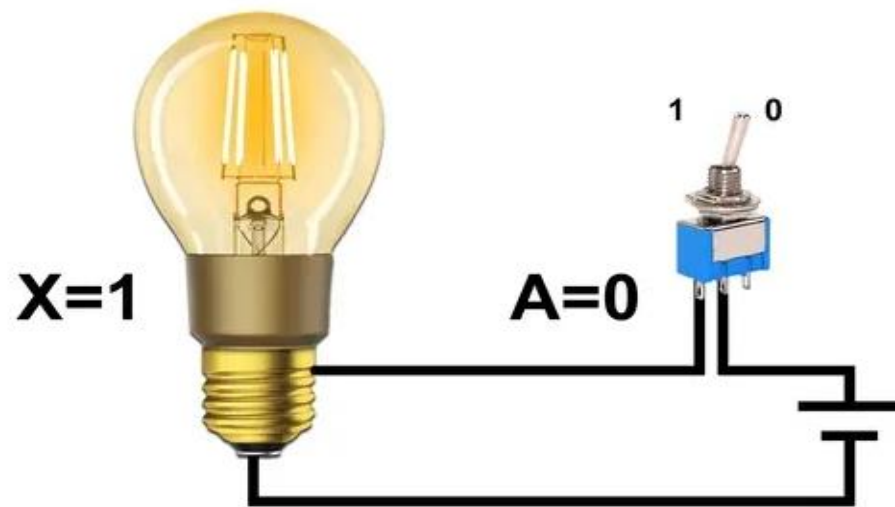


A	$X = \bar{A}$
0	1
1	0

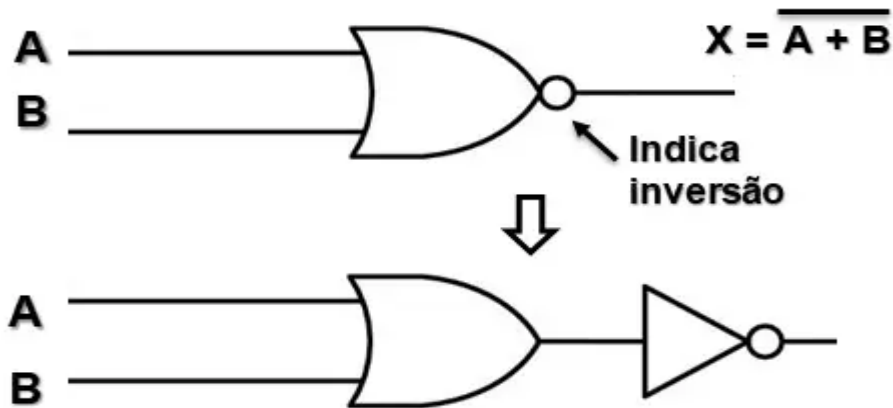
A porta lógica **NOT**, também conhecida como porta inversora, equivale a um circuito lógico capaz de realizar a operação booleana NOT, uma operação muito simples, em que a saída corresponde basicamente ao oposto do valor de entrada

Diferentemente das portas lógicas OR e AND, ela realiza apenas um valor de entrada. O CI mais conhecido é o 7404, que possui seis portas lógicas inversoras

Nas imagens vemos a simbologia e a expressão lógica booleana usadas para representar a porta NOT. Representamos a operação NOT com uma barra, e lemos a expressão como: “x é igual a NOT A” ou “x é igual ao inverso de A”. Vale a pena destacar que a presença do pequeno círculo sempre indica inversão



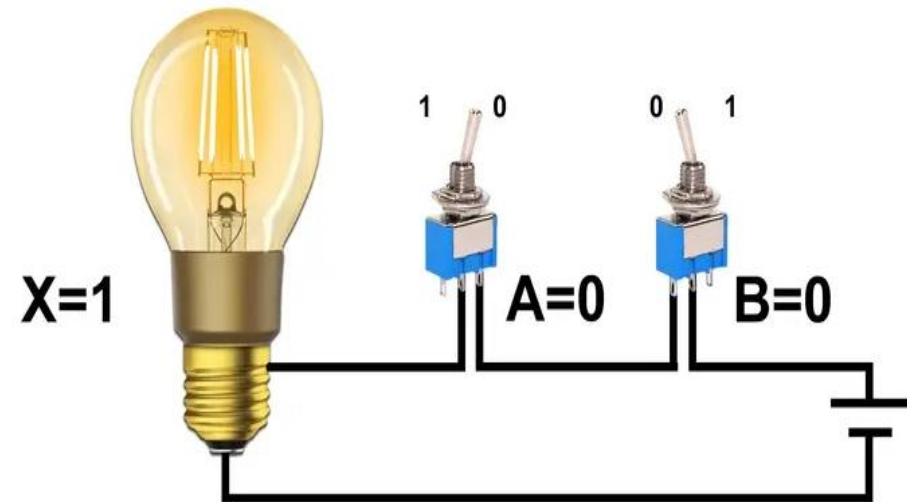
Porta Lógica NOR



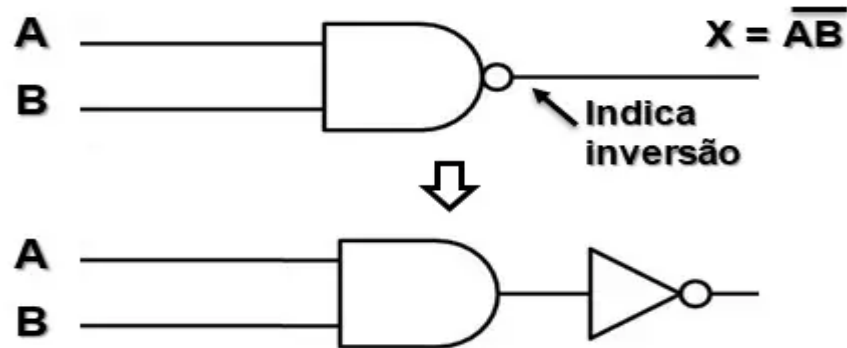
		OR	NOR
A	B	$A+B$	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

A porta lógica **NOR** é a combinação entre as portas lógicas OR e NOT e funciona de modo semelhante à porta OR, porém seguida de uma porta inversora, de forma que a saída da porta NOR será igual ao inverso da saída da porta OR

Neste caso, a variável de saída irá para o nível lógico baixo sempre que alguma das variáveis de entrada estiver em nível lógico alto. O CI mais conhecido que é capaz de fazer essa operação lógica é o 7402, que possui quatro portas NOR de duas entradas cada



Porta Lógica **NAND**

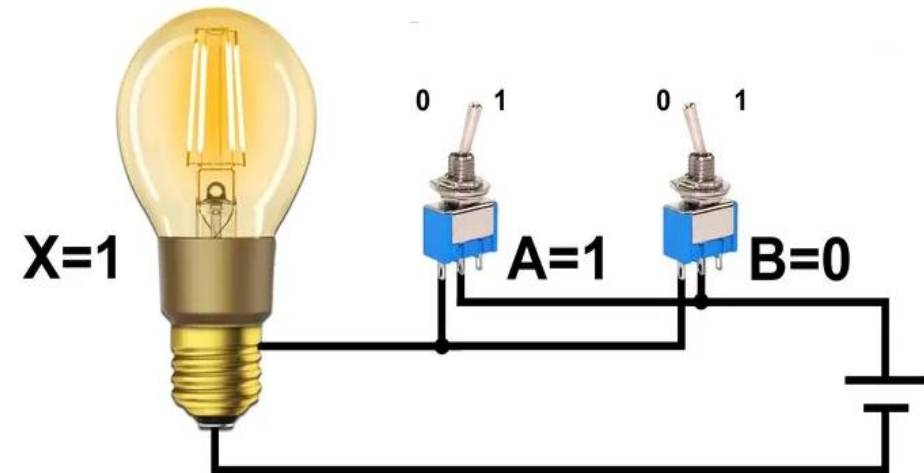


		AND	NAND
A	B	AB	\overline{AB}
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

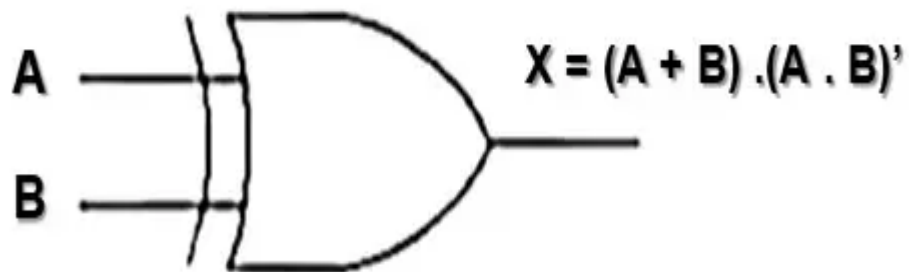
A porta **NAND** é a junção entre as portas lógicas AND e NOT e funciona de modo semelhante à porta AND, porém seguida de uma porta inversora, de forma que a saída da porta NAND é igual ao inverso da porta AND

Sendo assim, a variável de saída irá para o nível lógico alto sempre que alguma das variáveis de entrada estiver em nível lógico baixo. O circuito integrado mais conhecido que é capaz de fazer essa operação lógica é o 7400, que possui quatro portas NAND de duas entradas cada

Nas imagens vemos a simbologia e a expressão lógica booleana usadas para representar a porta NAND. Representamos a operação NAND como um produto lógico barrado, ou como um produto lógico inverso



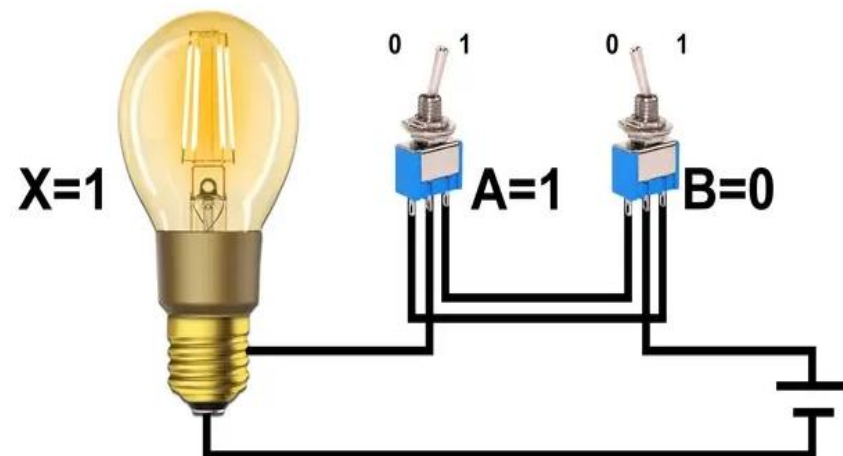
Porta Lógica XOR



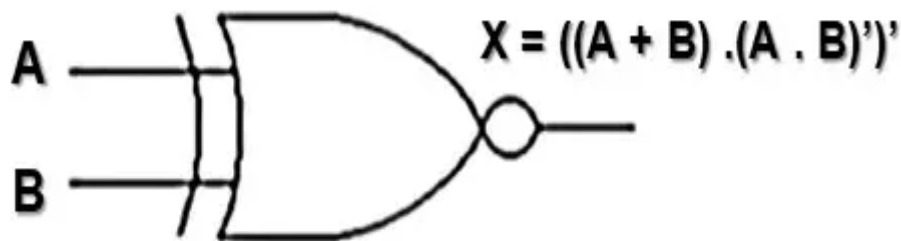
A	B	$X = (A \oplus B)$
0	0	0
0	1	1
1	0	1
1	1	0

A porta lógica **XOR**, conhecida também como circuito anti coincidência, é representada por uma porta OR exclusiva, que consiste em valores de saída com nível lógico alto sempre que as variáveis de entrada, A e B, forem diferentes. O CI mais conhecido é o 7486, que possui quatro portas XOR de duas entradas cada

Nas imagens vemos a simbologia e a expressão lógica booleana usadas para representar a porta XOR



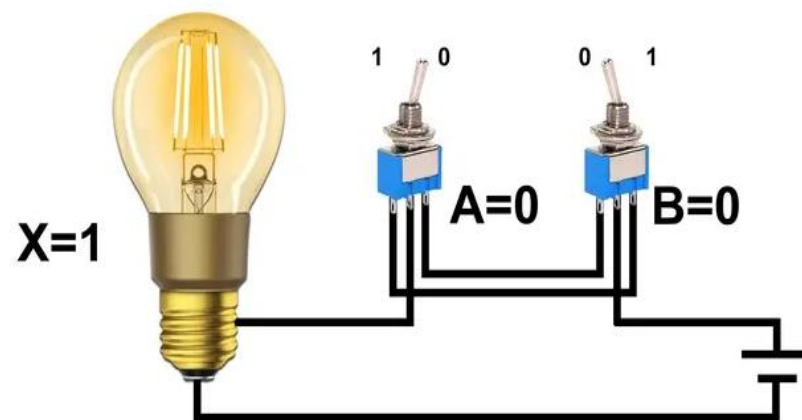
Porta Lógica XNOR



A	B	$X = (\overline{A \oplus B})$
0	0	1
0	1	0
1	0	0
1	1	1









A porta lógica **XNOR**, conhecida também como circuito de coincidência, é representada por uma porta NOR exclusiva, em que os valores de saída terão nível lógico alto sempre que as variáveis de entrada, A e B, forem iguais. O CI mais conhecido é o 74266, que possui quatro portas XNOR de duas entradas cada

Nas imagens vemos a simbologia e a expressão lógica booleana usadas para representar a porta XNOR



Portas Lógicas - RESUMO

Logic Gates - Symbols and Truth Tables

<div>BUF (Buffer)</div> <div></div>	<table><tr><th>In</th><th>Out</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	In	Out	0	0	1	1	<div>NOT (Inverter)</div> <div></div>	<table><tr><th>In</th><th>Out</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	In	Out	0	1	1	0																		
In	Out																																
0	0																																
1	1																																
In	Out																																
0	1																																
1	0																																
<div>AND</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	In1	In2	Out	0	0	0	0	1	0	1	0	0	1	1	1	<div>NAND (NOT AND)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	In1	In2	Out	0	0	1	0	1	1	1	0	1	1	1	0
In1	In2	Out																															
0	0	0																															
0	1	0																															
1	0	0																															
1	1	1																															
In1	In2	Out																															
0	0	1																															
0	1	1																															
1	0	1																															
1	1	0																															
<div>OR</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	In1	In2	Out	0	0	0	0	1	1	1	0	1	1	1	1	<div>NOR (NOT OR)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	In1	In2	Out	0	0	1	0	1	0	1	0	0	1	1	0
In1	In2	Out																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	1																															
In1	In2	Out																															
0	0	1																															
0	1	0																															
1	0	0																															
1	1	0																															
<div>XOR (Exclusive Or)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	In1	In2	Out	0	0	0	0	1	1	1	0	1	1	1	0	<div>XNOR (NOT XOR)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	In1	In2	Out	0	0	1	0	1	0	1	0	0	1	1	1
In1	In2	Out																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	0																															
In1	In2	Out																															
0	0	1																															
0	1	0																															
1	0	0																															
1	1	1																															

A circle behind a symbol indicates that the output signal is inverted.

Portas Lógicas – Circuito Somador

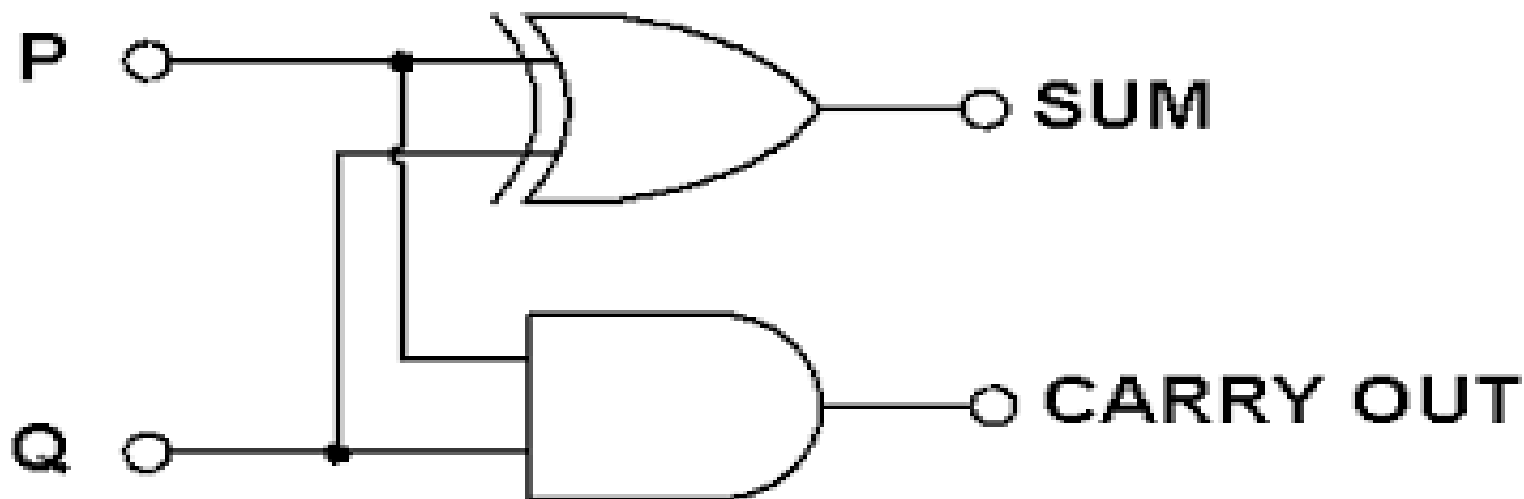
Para que o computador possa fazer tudo que conhecemos no nosso universo, deve ter uma forma de converter a matemática dele, baseada em portas lógicas, para nossa matemática baseada em somas, subtrações, multiplicações e divisões

A porta NAND é um dispositivo de computação universal. Com ela podemos criar qualquer outra coisa, inclusive a matemática que conhecemos. Mas como isto funciona?

Tendo-se os valores que desejamos nas entradas, podemos gerar qualquer valor que quisermos na saída, usando apenas portas lógicas. Vamos pensar em uma soma simples de um dígito. Se somarmos $1+0=1$ e $0+1=1$. Ou seja, os valores das duas entradas precisam ser diferentes (XOR), para que a saída seja 1

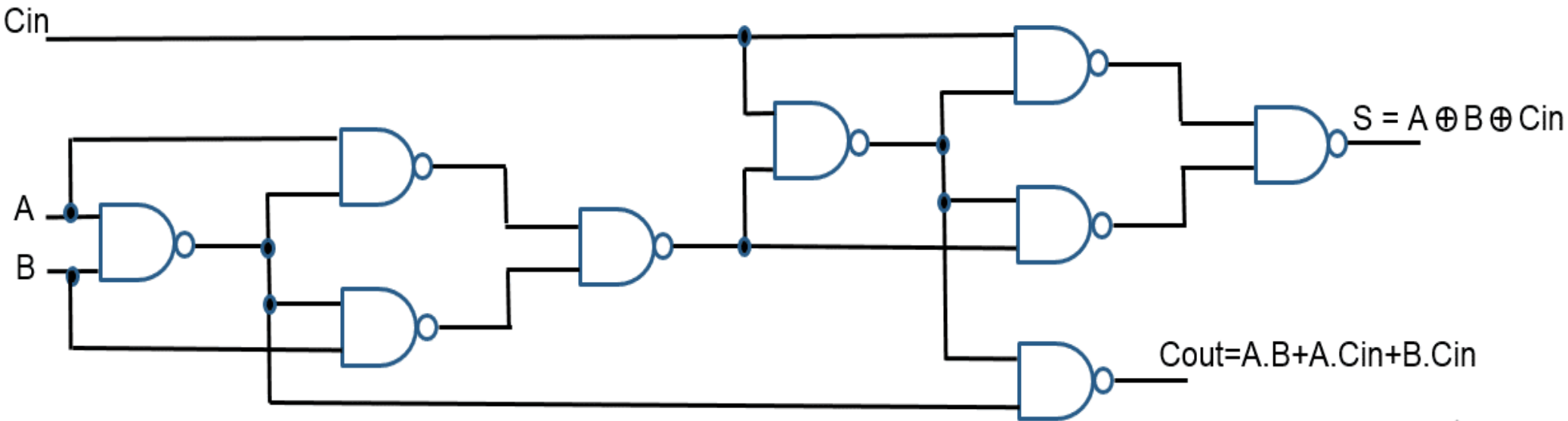
Se as duas entradas forem 0, teremos $0+0=0$. E se as duas entradas forem 1? Bem, $1+1=2$, e precisamos de dois dígitos em binário para representar o dois. Portanto $1+1=0$ "e vai um"

Portas Lógicas – Circuito Somador



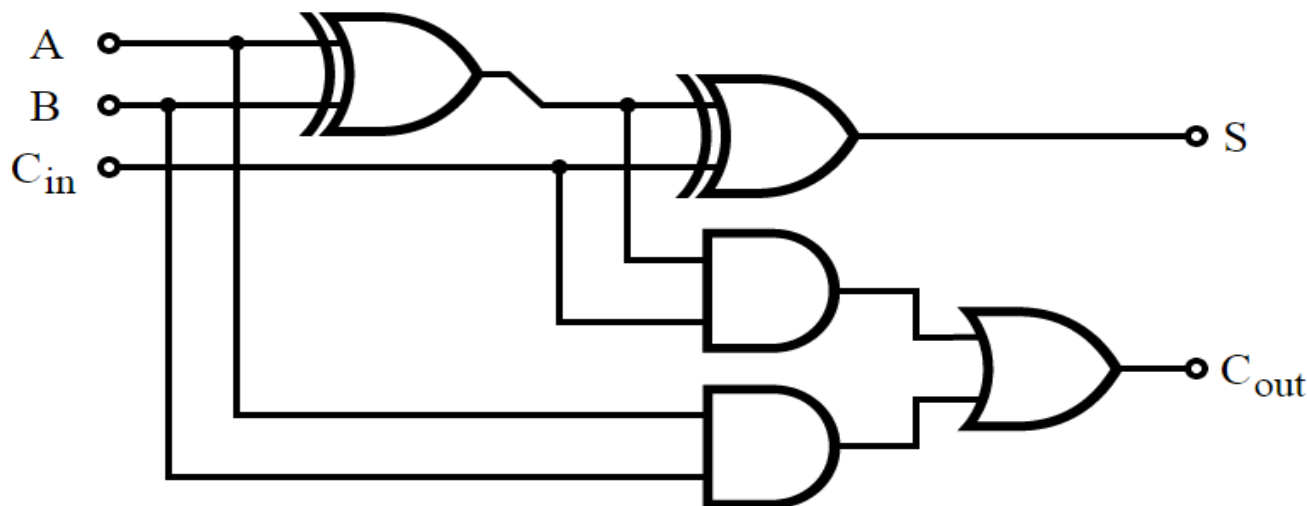
Dá para entender como ele funciona? Vimos que o XOR (que pode ser feito com portas NAND) gera 1 na saída se as duas entradas forem diferentes, uma sendo 0 e a outra sendo 1. E o "vai um" da matemática só acontece quando as duas entradas forem iguais a 1. **Mas este circuito ainda é pouco útil, porque para ligar vários em sequência, precisamos fornecer duas entradas e O VALOR DO VAI UM da operação anterior.** Precisamos de um circuito um pouco mais elaborado que este

Portas Lógicas – Circuito Somador



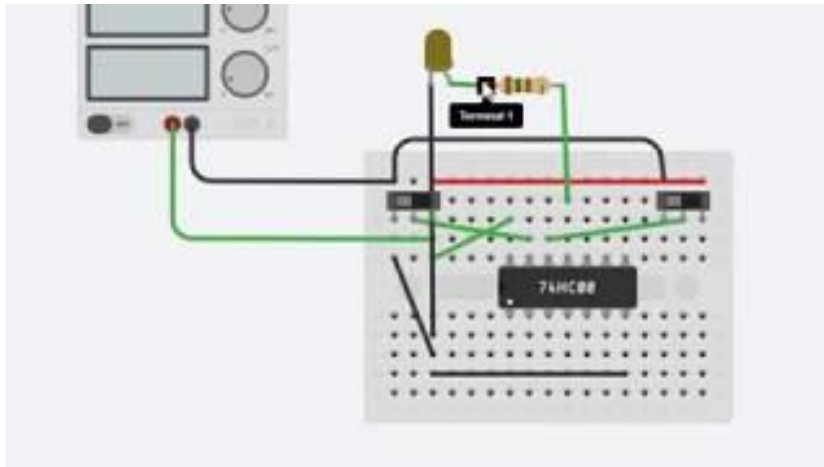
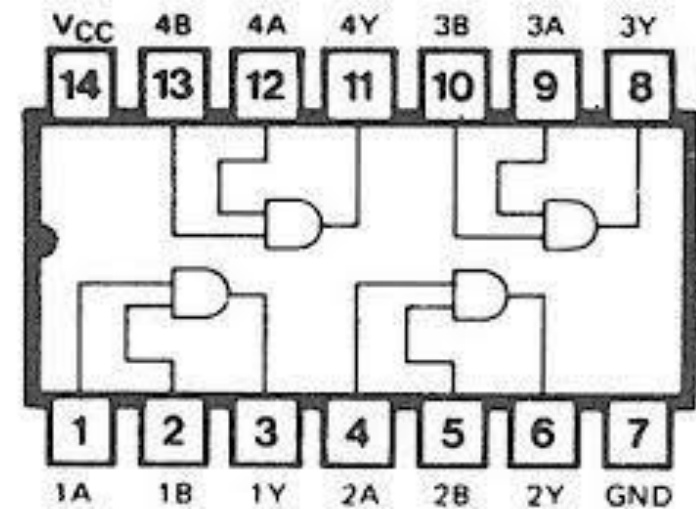
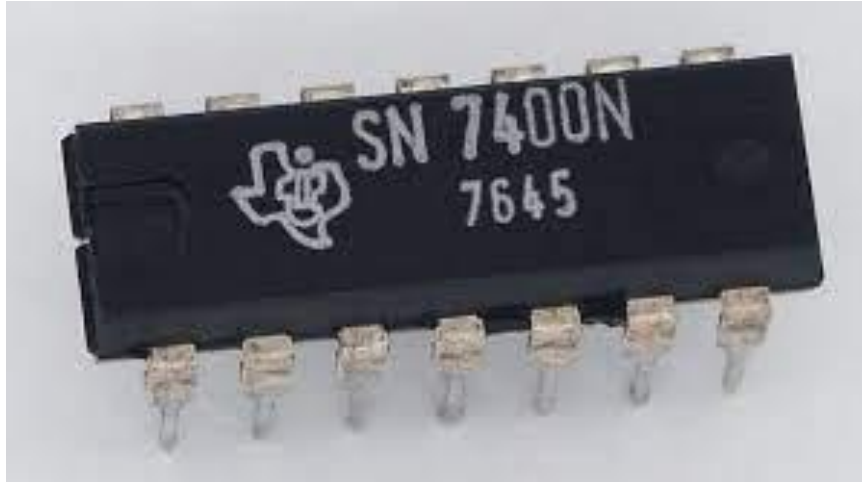
Este circuito aí acima, construído apenas com portas NAND, é o que chamamos de um somador completo

Portas Lógicas – Circuito Somador



Ele é como que o meio somador, mas ao invés de somente duas entradas, temos os dois valores e mais um valor que é o "vai um" do bit anterior (chamamos de **carry**). Se não tivermos o bit anterior, porque não houve um bit menos significativo, basta jogar 0 em **C_{in}**. Ligando vários destes em sequência, podemos efetuar somas com quantas entradas quisermos, porque não há limite de somadores que podem ser ligados em sequência. Lembre-se que na matemática o "vai um" da unidade para a dezena funciona da mesma forma que o "vai um" da dezena para centena, e vai assim até o infinito. Ou seja, basta ligar o **C_{out}** de um circuito deste acima no **C_{in}** de outro, para termos um somador de dois números de dois bits. E podemos juntar vários para fazer somas de quantos bits quisermos

Portas Lógicas – Circuitos Integrados



Portas Lógicas

Prática: <https://academo.org/demos/logic-gate-simulator/>

Logic Gate Simulator

A free, simple, online logic gate simulator. Investigate the behaviour of AND, OR, NOT, NAND, NOR and XOR gates. Select gates from the dropdown list and click "add node" to add more gates. Drag from the hollow circles to the solid circles to make connections. Right click connections to delete them. See below for more detailed instructions.

Engineering

Electronics

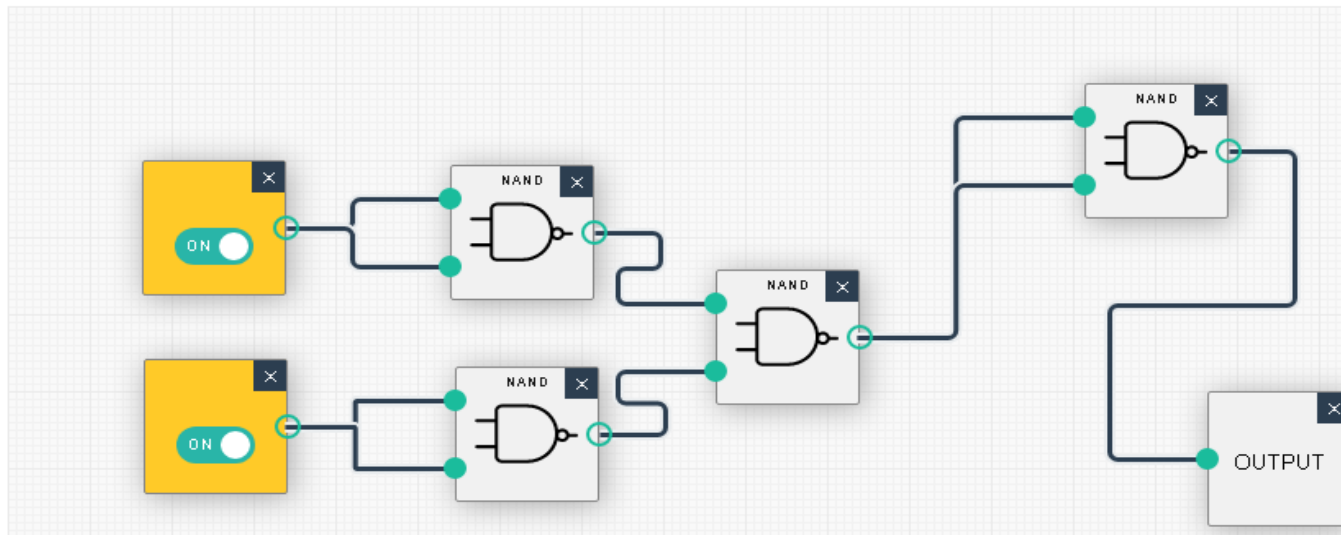
Logic



Share

Tweet

BECOME A PATRON

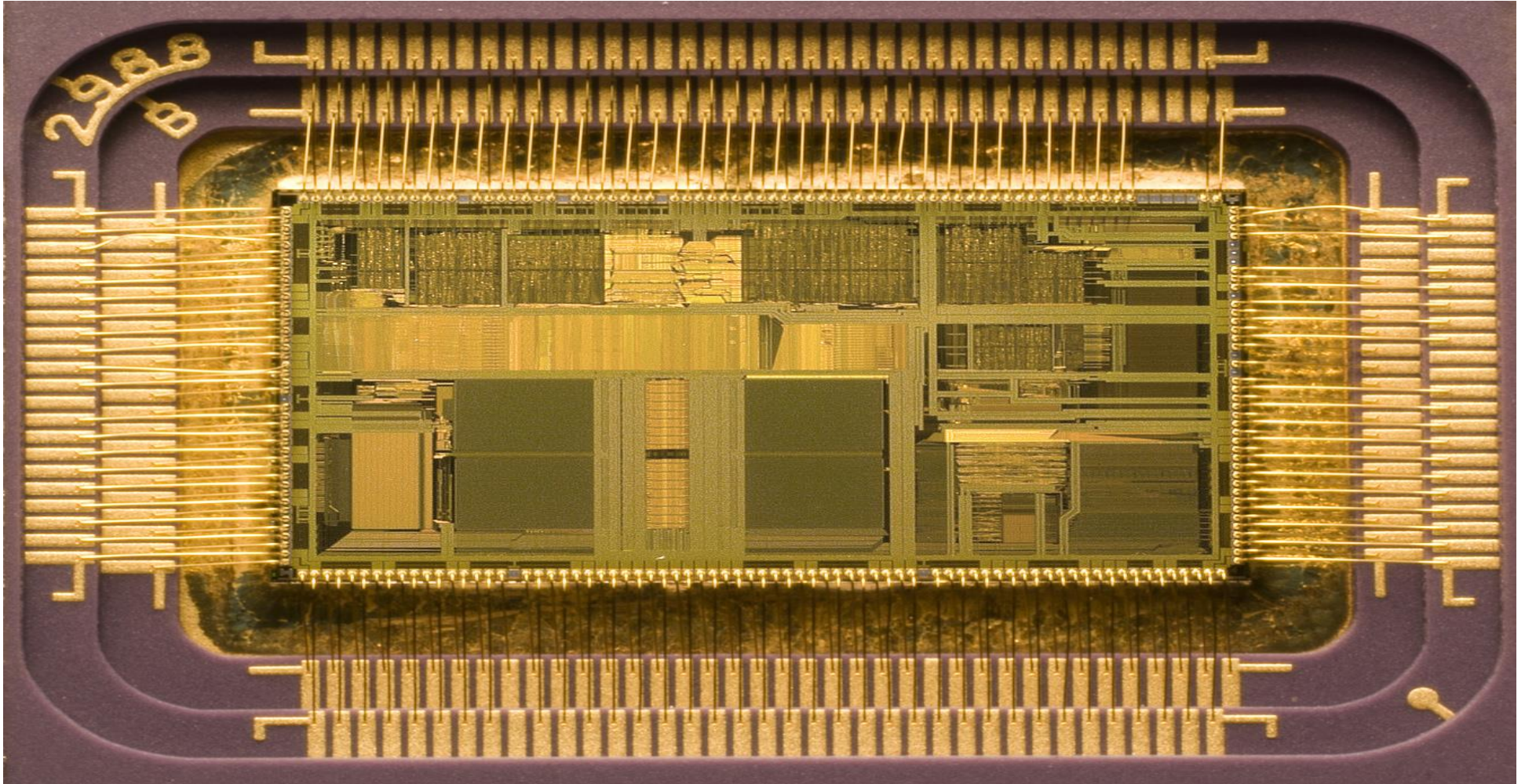


NAND

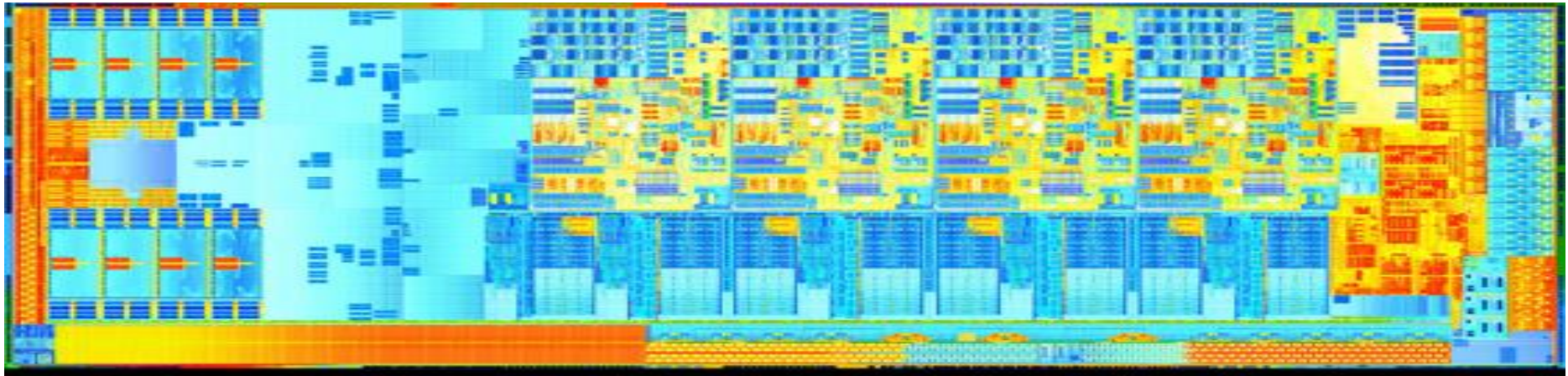
Add node

Full screen mode

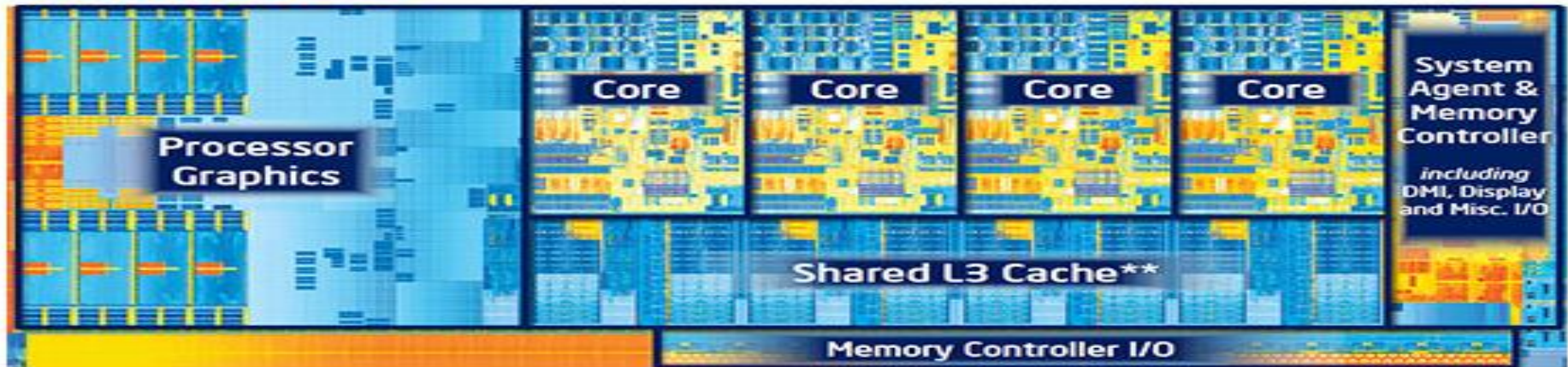
Por dentro da CPU - CHIP



Por dentro da CPU - CHIP



**3rd Generation Intel® Core™ Processor:
22nm Process**

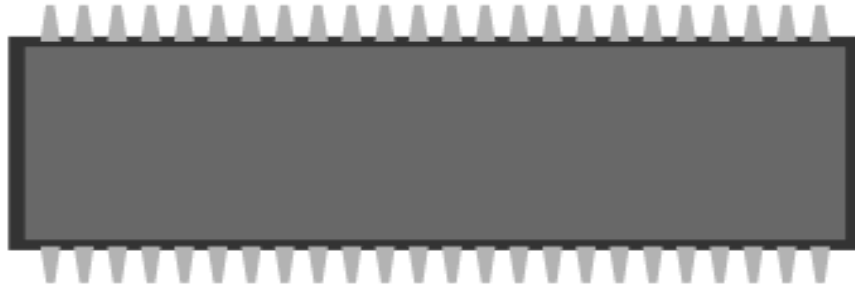


New architecture with shared cache delivering more performance and energy efficiency

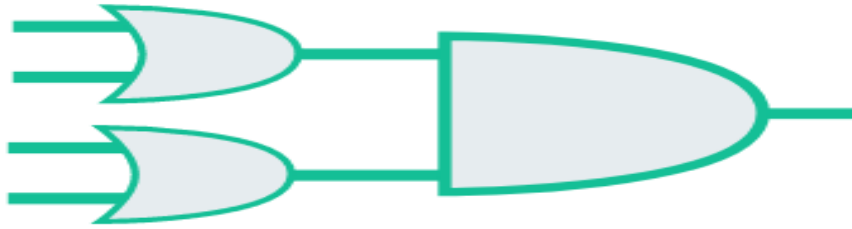
Quad Core die with Intel® HD Graphics 4000 shown above
 Transistor count: 1.4Billion Die size: 160mm²

** Cache is shared across all 4 cores and processor graphics

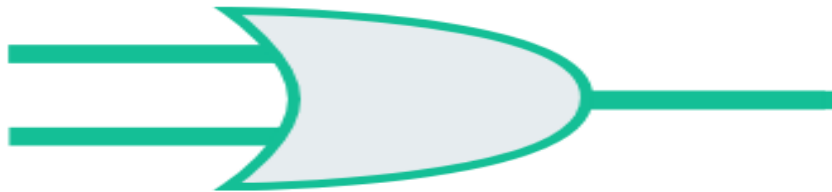
Por dentro da CPU - CHIP



CHIP



CIRCUIT

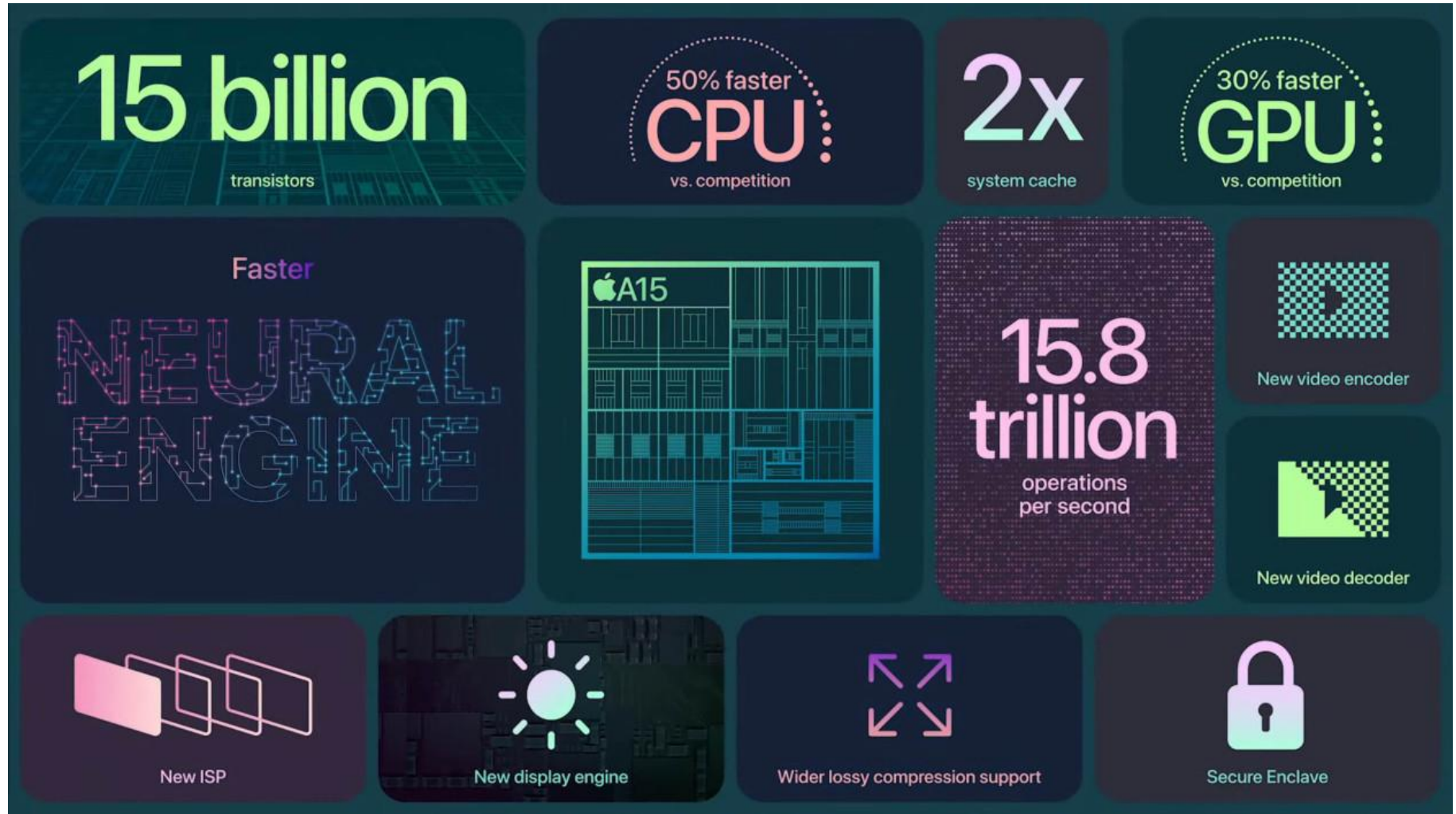


LOGIC GATE

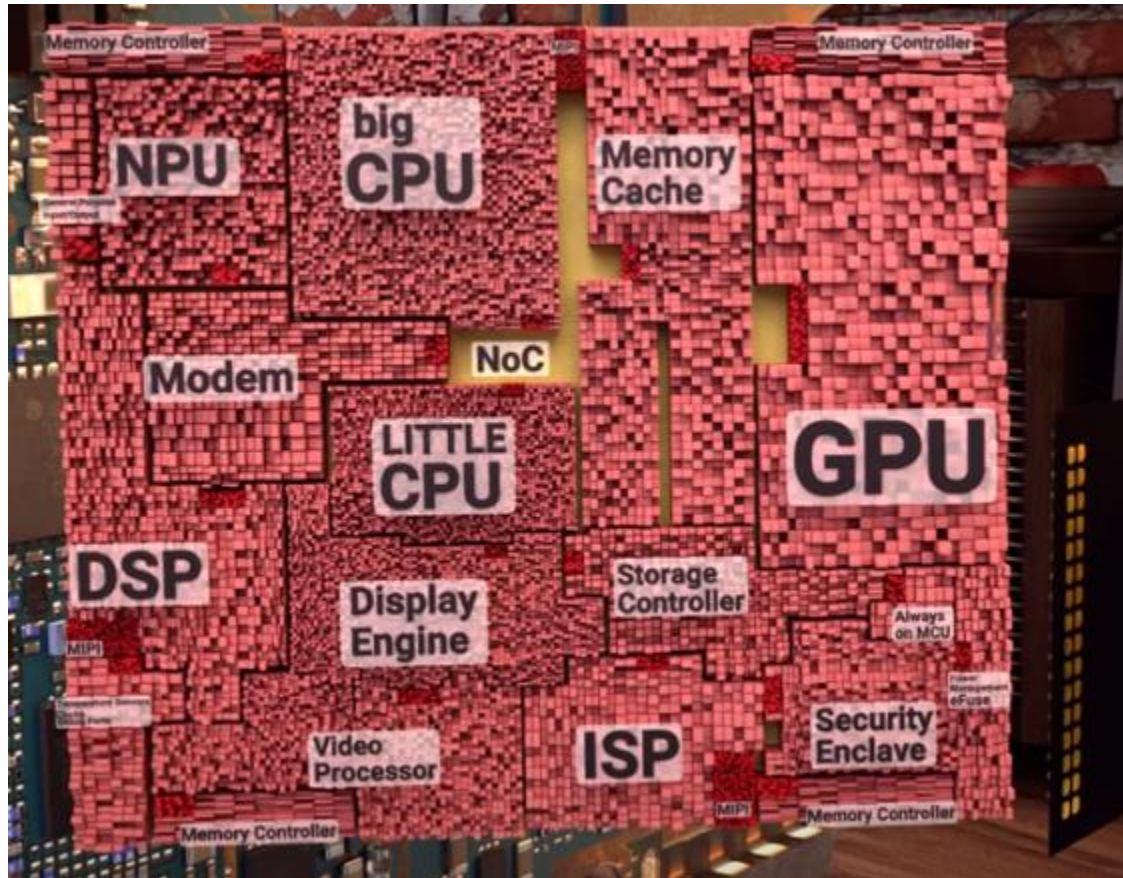


TRANSISTORS

SOC – System On a Chip



SOC – System On a Chip



<https://www.youtube.com/watch?v=NKfW8ijmRQ4&t=447s>

Microcontroladores



(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
VCC	7	22	GND
GND	8	21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 (\overline{SS} /OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

Power
Ground
Programming/debug
Digital
Analog
Crystal/Osc

Muitas vezes confundido com “microprocessador”, talvez pela semelhança de nomes, o microcontrolador consiste em um único circuito integrado que reúne um núcleo de processador, memórias voláteis e não voláteis e diversos periféricos de entrada e de saída de dados. Ou seja, ele nada mais é do que um computador muito pequeno capaz de realizar determinadas tarefas de maneira eficaz e sob um tamanho altamente compacto

Um microprocessador, por sua vez, contém apenas um processador de tamanho bastante pequeno no circuito integrado. Dessa maneira, ele não dispõe de periféricos tais como contadores, conversores e memórias variadas. Sendo assim, ele é capaz de executar apenas funções lógicas e aritméticas definidas pelo programa

Dentro do mercado atual, existem três grandes marcas que são bastante populares dentro do nicho de microcontroladores. São elas: a linha PIC da Microchip, a Intel MCS da Intel e o Atmel AVR da Atmel. Pode-se dizer também que a mais popular dentre as três seria a Atmel, uma vez que é a linha utilizada nas placas de Arduino, plataforma bastante difundida entre estudantes devido à sua simplicidade e acessibilidade

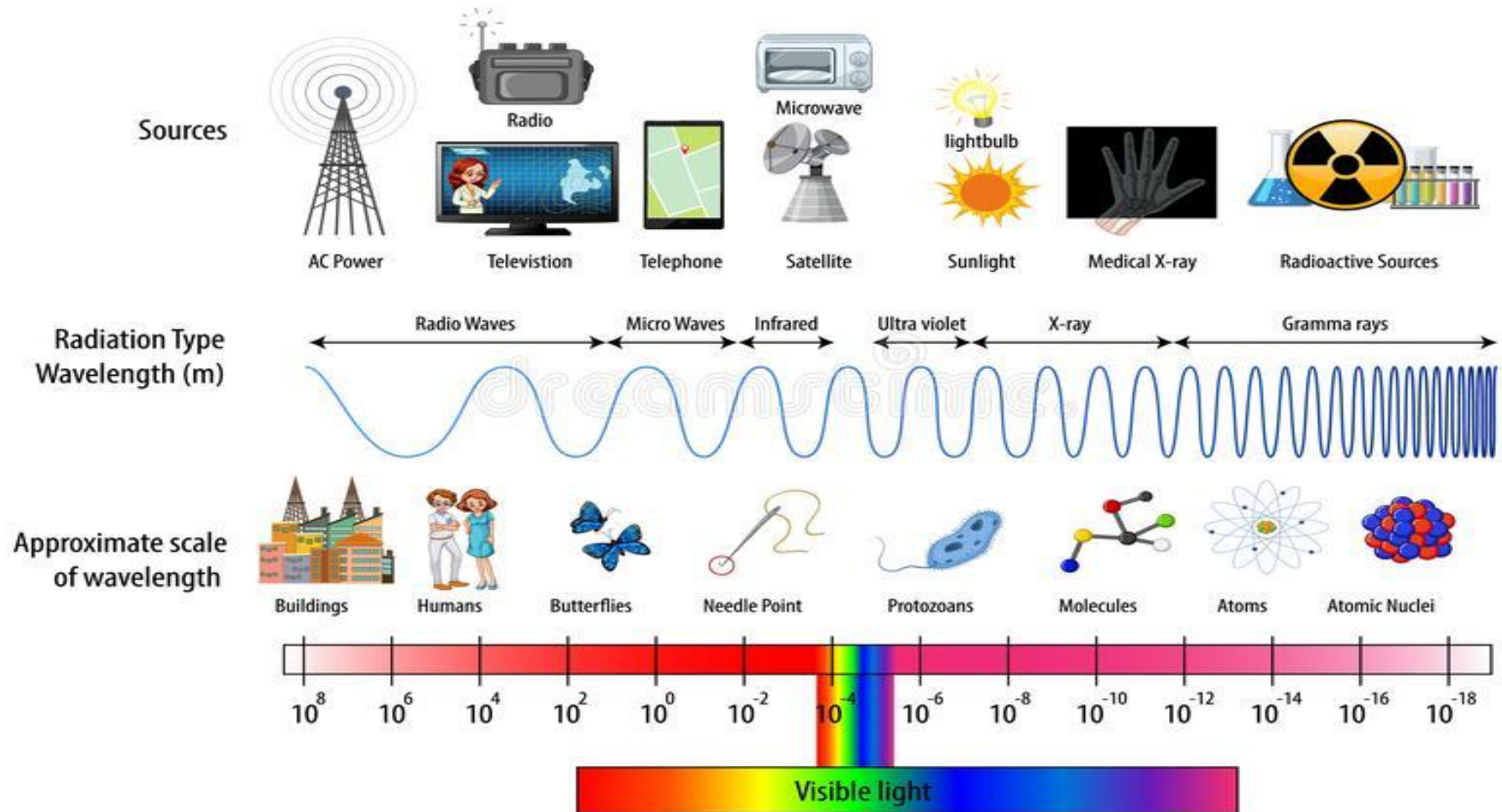
As Escalas da Natureza

O ângstron é um submúltiplo do metro e Vale $10^{-10} \text{ m} = 0,0000000001 \text{ m}$ um n° muito pequeno usado para medir átomos, micro organismos etc... Cada unidade de medida de comprimento é usada para determinado fim.

Múltiplos do metro	submúltiplos do metro
Yottametro (Ym) : 10^{24} m	decímetro (dm) : 10^{-1} m
Zettametro (Zm) : 10^{21} m	centímetro (cm) : 10^{-2} m
Exametro (Em) : 10^{18} m	milímetro (mm) : 10^{-3} m
Petametro (Pm) : 10^{15} m	micrometro (um) : 10^{-6} m
Terametro (Tm) : 10^{12} m	nanometro (nm) : 10^{-9} m
Gigametro (Gm) : 10^9 m	Ângstron (A) : 10^{-10} m
Quilômetro (km) : 10^3 m	picometro (pm) : 10^{-12} m
Hectômetro (Hm) : 10^2 m	femtômetro(fm) : 10^{-15} m
Decâmetro (Dm) : 10^1 m	attômetro (am) : 10^{-18} m
Metro (m) : 10^0 m	zeptômetro(zm) : 10^{-21} m
	yoctômetro (ym) : 10^{-24} m
Prof.Estevam * 2010 *	

As Escalas da Natureza

THE ELECTROMAGNETIC SPECTRUM



Tarefa: Leitura e Discussão em Sala de Aula

1- Faça leitura do artigo “ A Máquina mais Valiosa do Mundo” – Revista Super Interessante – Ed. Abril

<https://super.abril.com.br/tecnologia/a-maquina-mais-valiosa-do-mundo/#:~:text=Ela%20custa%20US%24%20150%20milh%C3%B5es,que%20a%20China%20n%C3%A3o%20tem>





SÃO
PAULO
TECH
SCHOOL