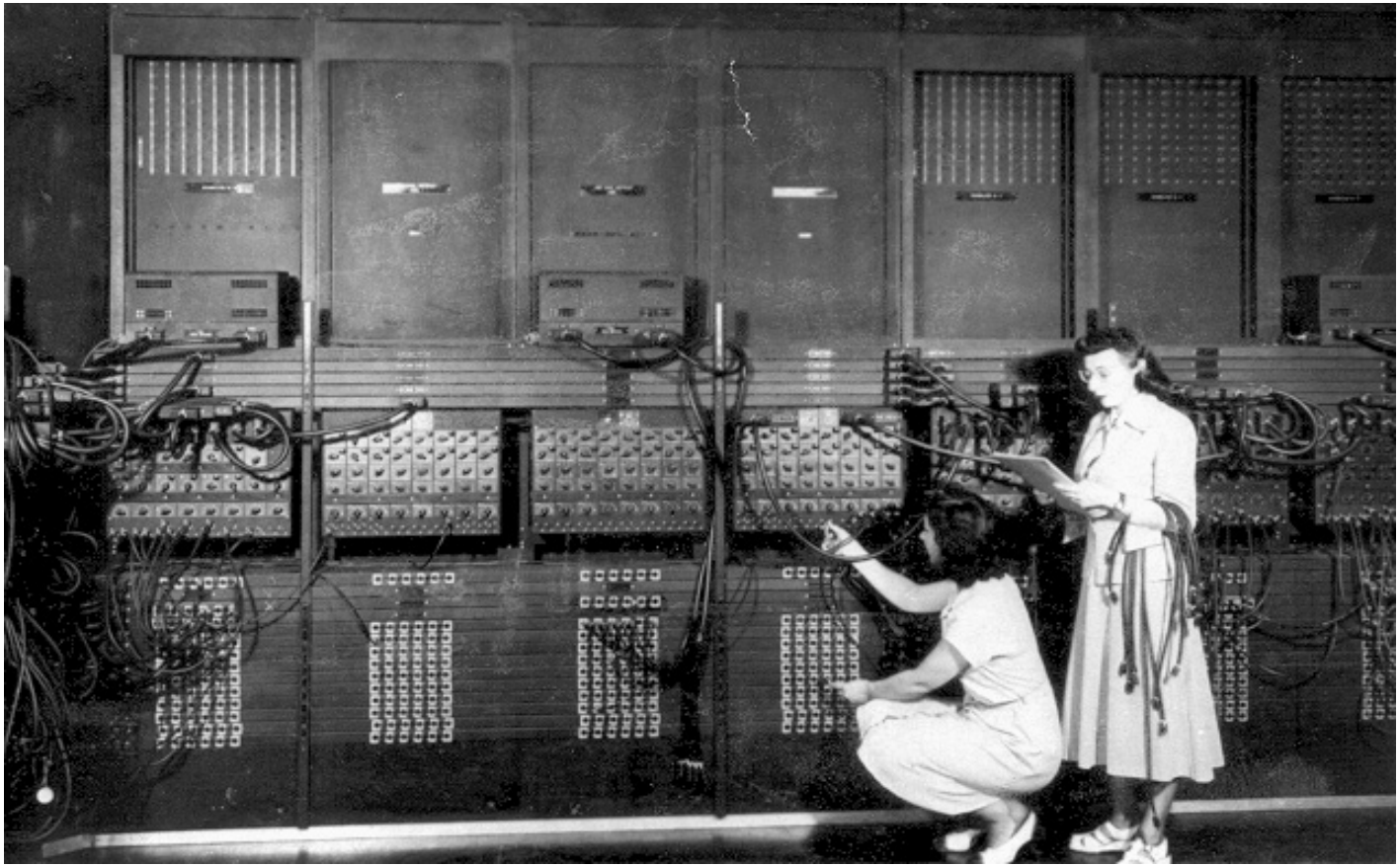
Explicações em Figuras – ALP

Claudio Cesar de Sá
claudio.sa@udesc.br

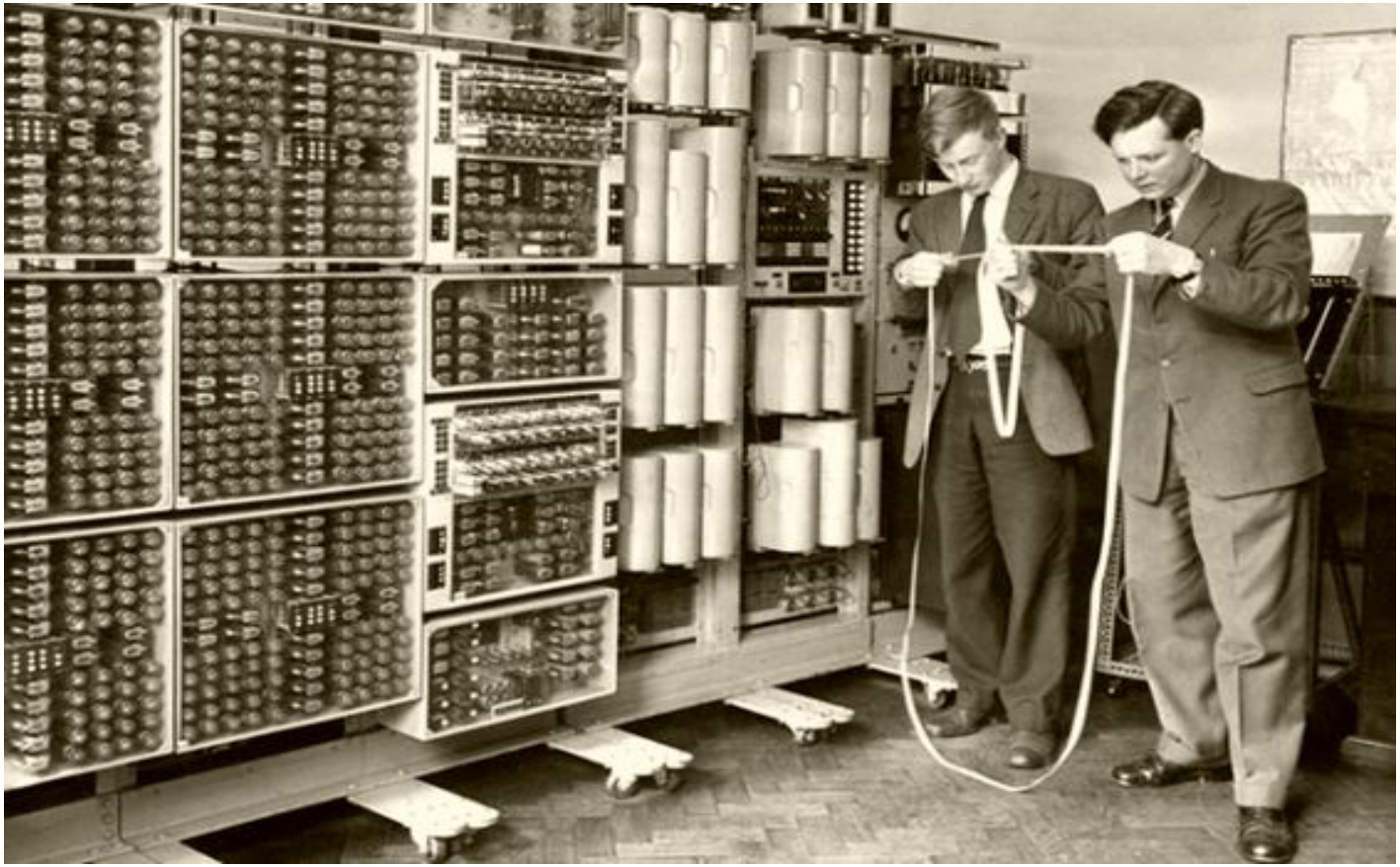
Departamento de Ciência da Computação
Centro de Ciências e Tecnológias
Universidade do Estado de Santa Catarina

8 de março de 2017

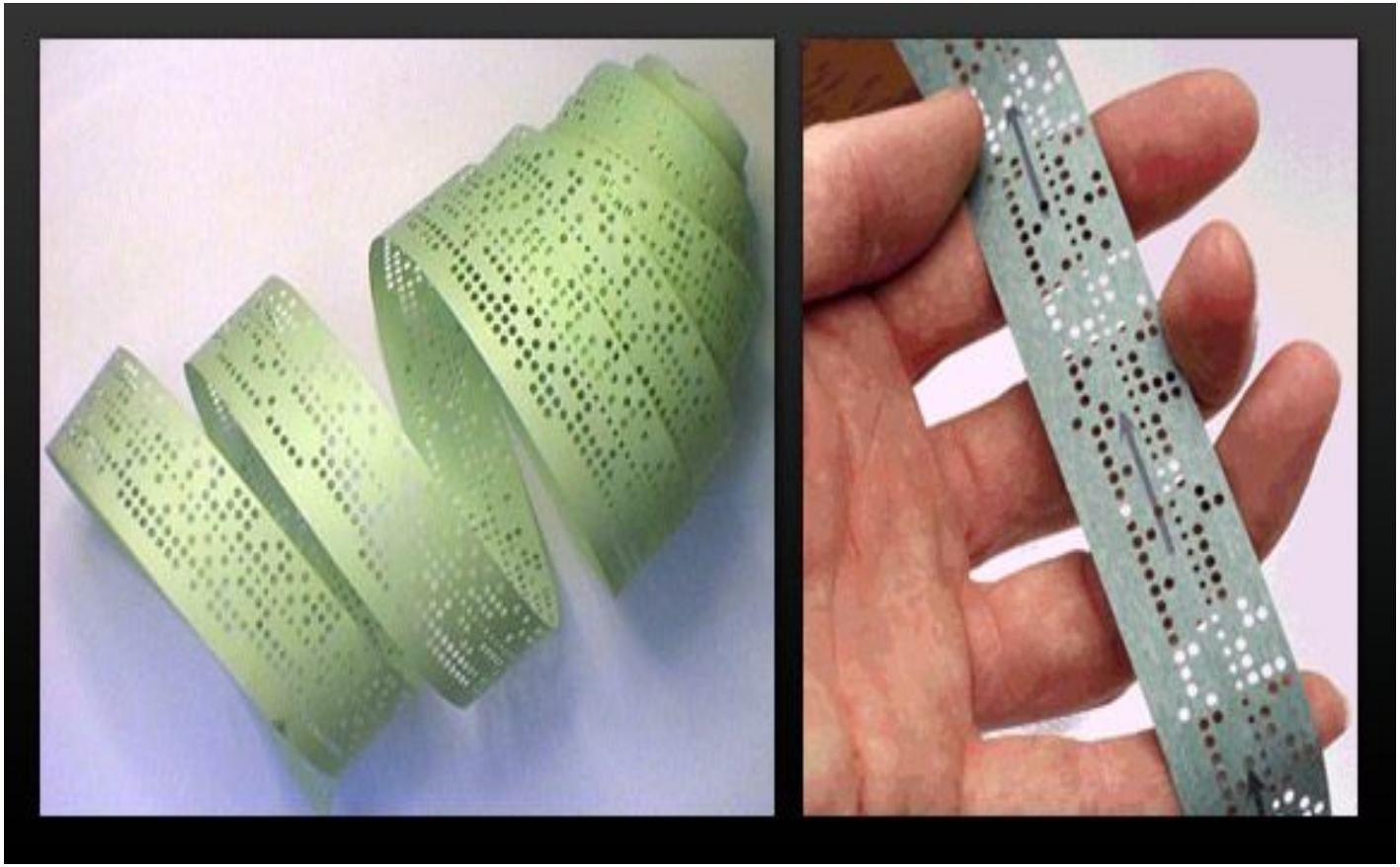
Os primeiros computadores



Computador digital – veja a fita em papel



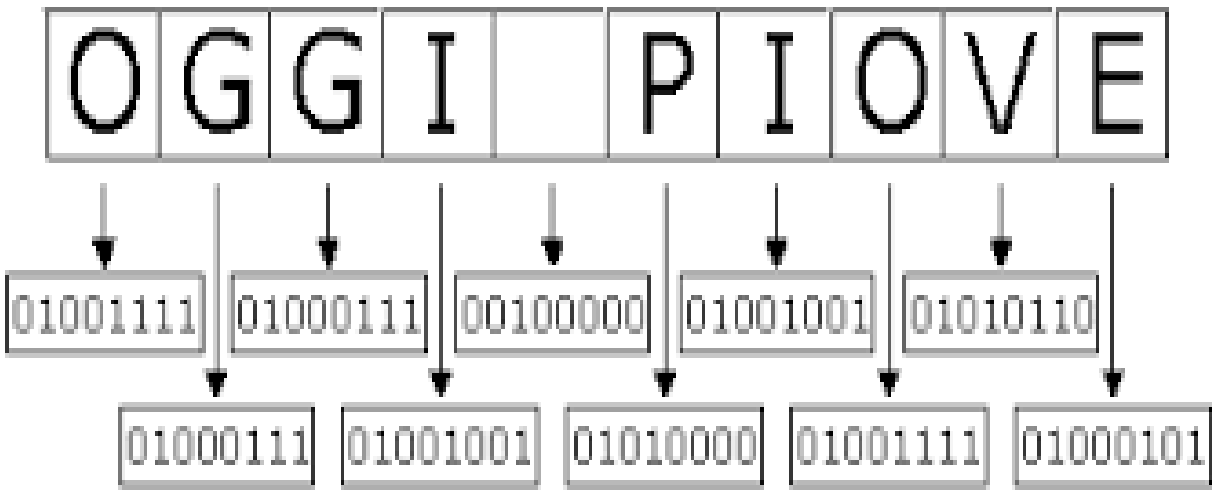
A fita perfurada = codificada



Troque *furos* por **0** e *espaços* por **1** e bem-vindo ao mundo digital!

$2^0 = 1 = \text{sim!} = 1$
$2^1 = 2 = \text{sim!} = 1$
$2^2 = 4 = \text{não!} = 0$
$2^3 = 8 = \text{sim!} = 1$
Da direita pra esquerda:
$11 = 1011$

Melhorando a leitura dos **0's** e **1's** $\rightarrow$ sistemas de codificação



Internamente ao processador temos algo *análogo* a:

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

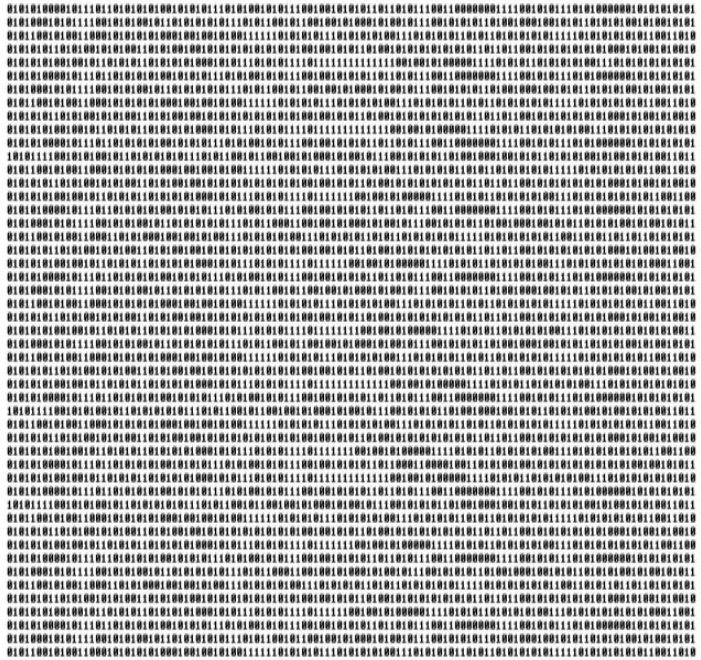
Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

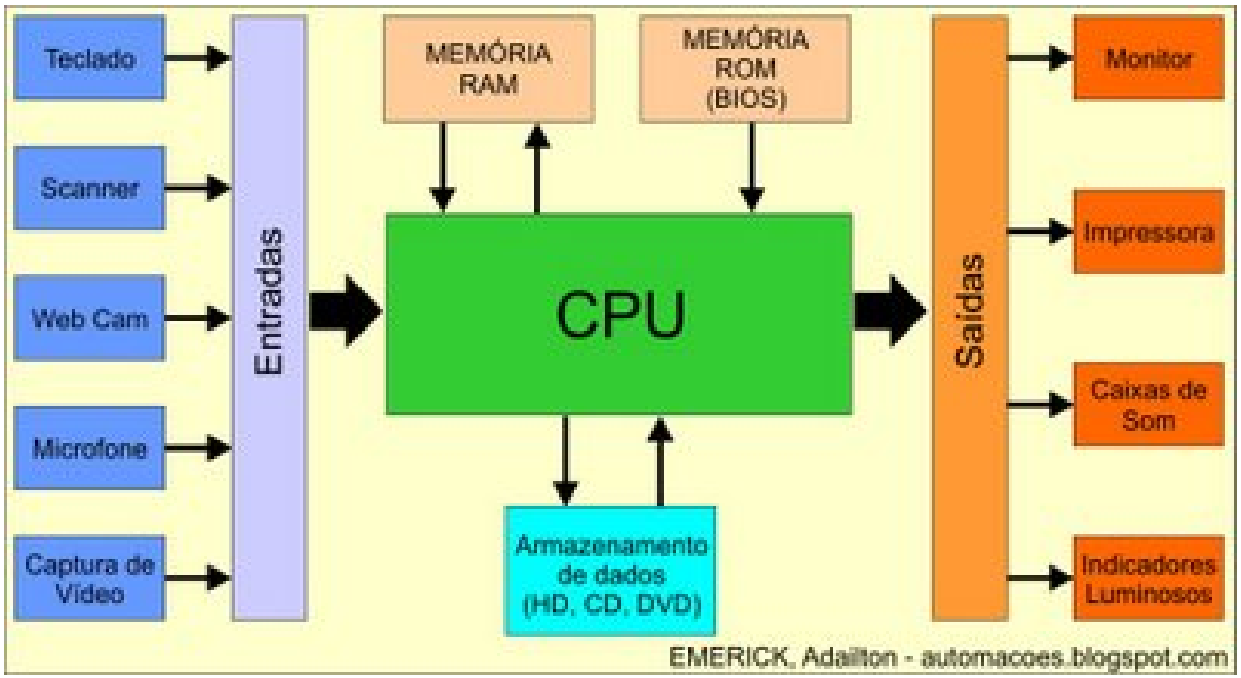
Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

Figura: Acreditem ... há sequências binárias, decimais e mnêmonicos!

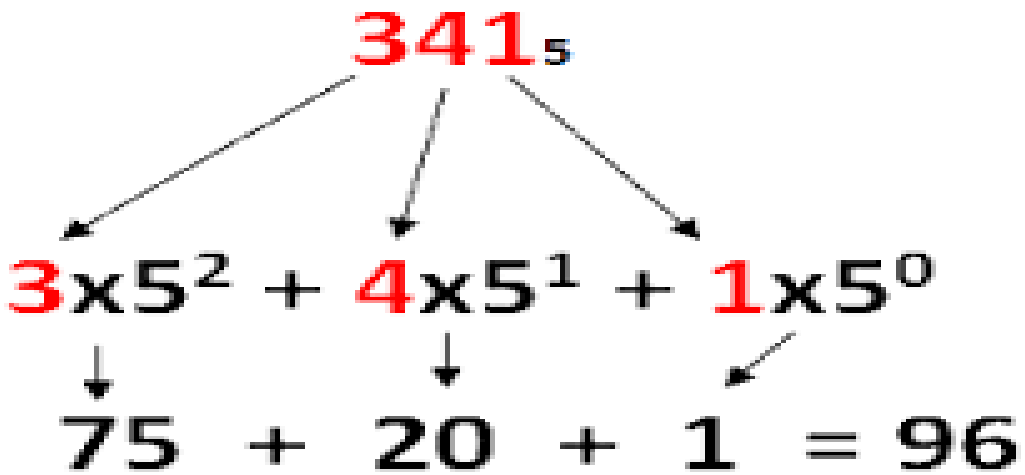
Logo, apenas binários nas memórias ...



Relembrando um PC internamente $\equiv$ computador

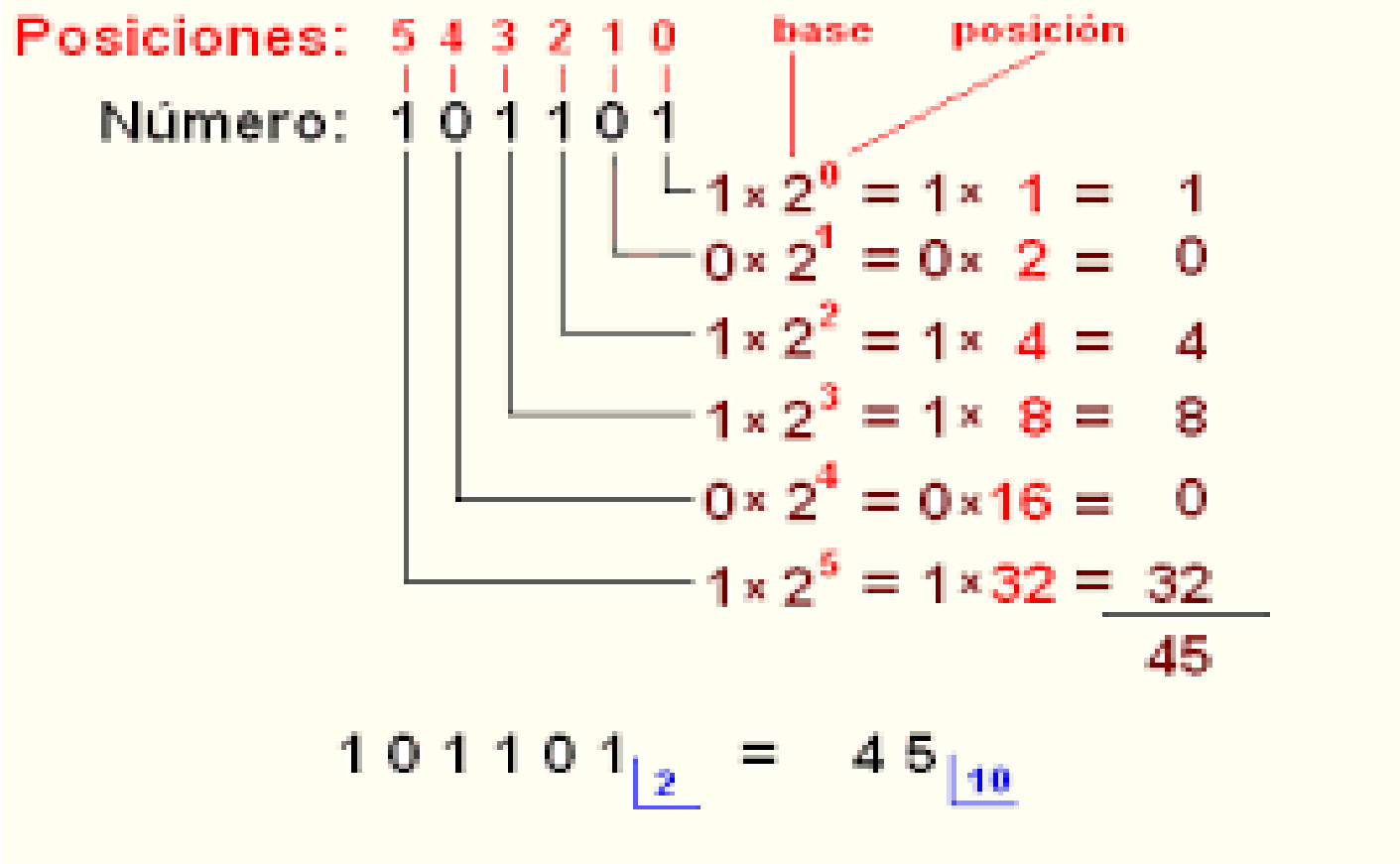


Sistema decimal (compreensível para nós) logo: decimal $\Leftrightarrow$ binário

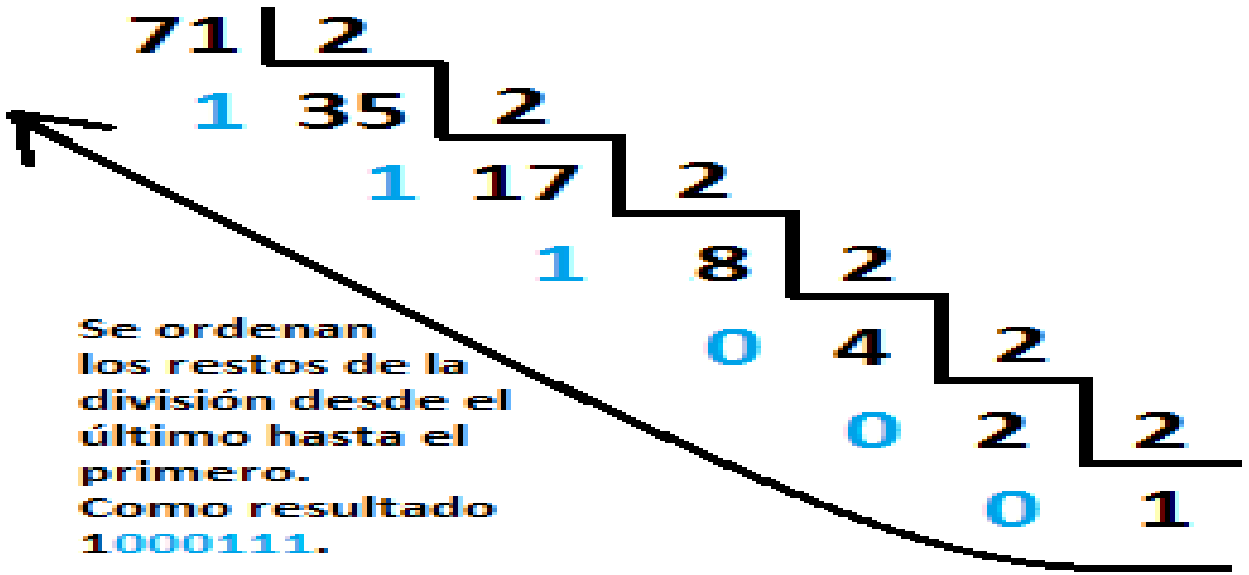


$$341_5 = 96_{10}$$

Binário para decimal



Decimal para Binário



71 se escribe en sistema binario como 1000111

O que vamos fazer no curso (**gravem esta figura**):

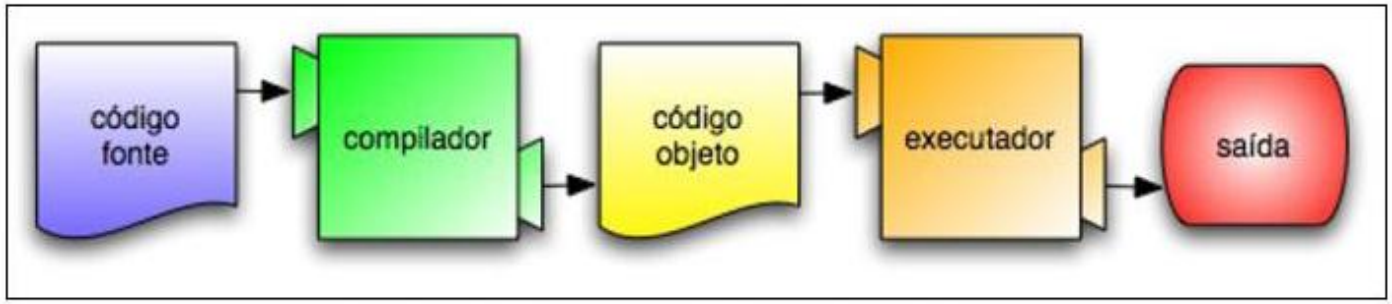


Figura 01 – Processo simprificado de compilação

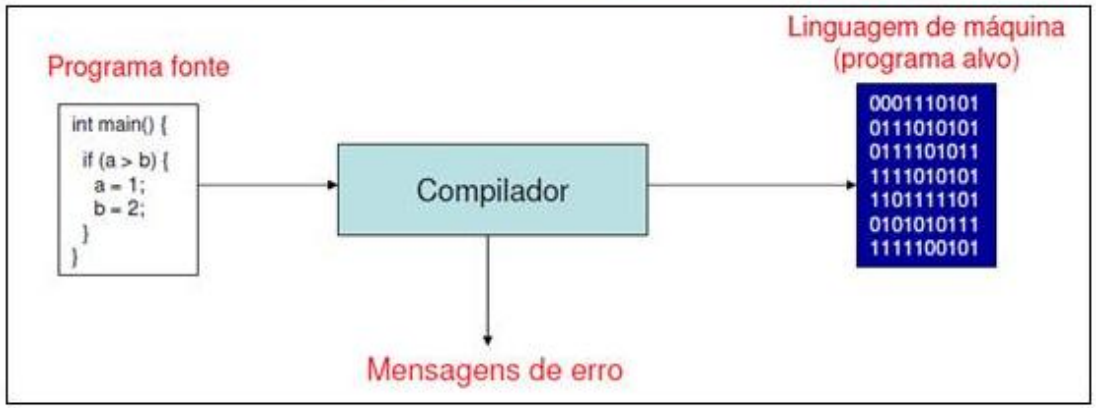
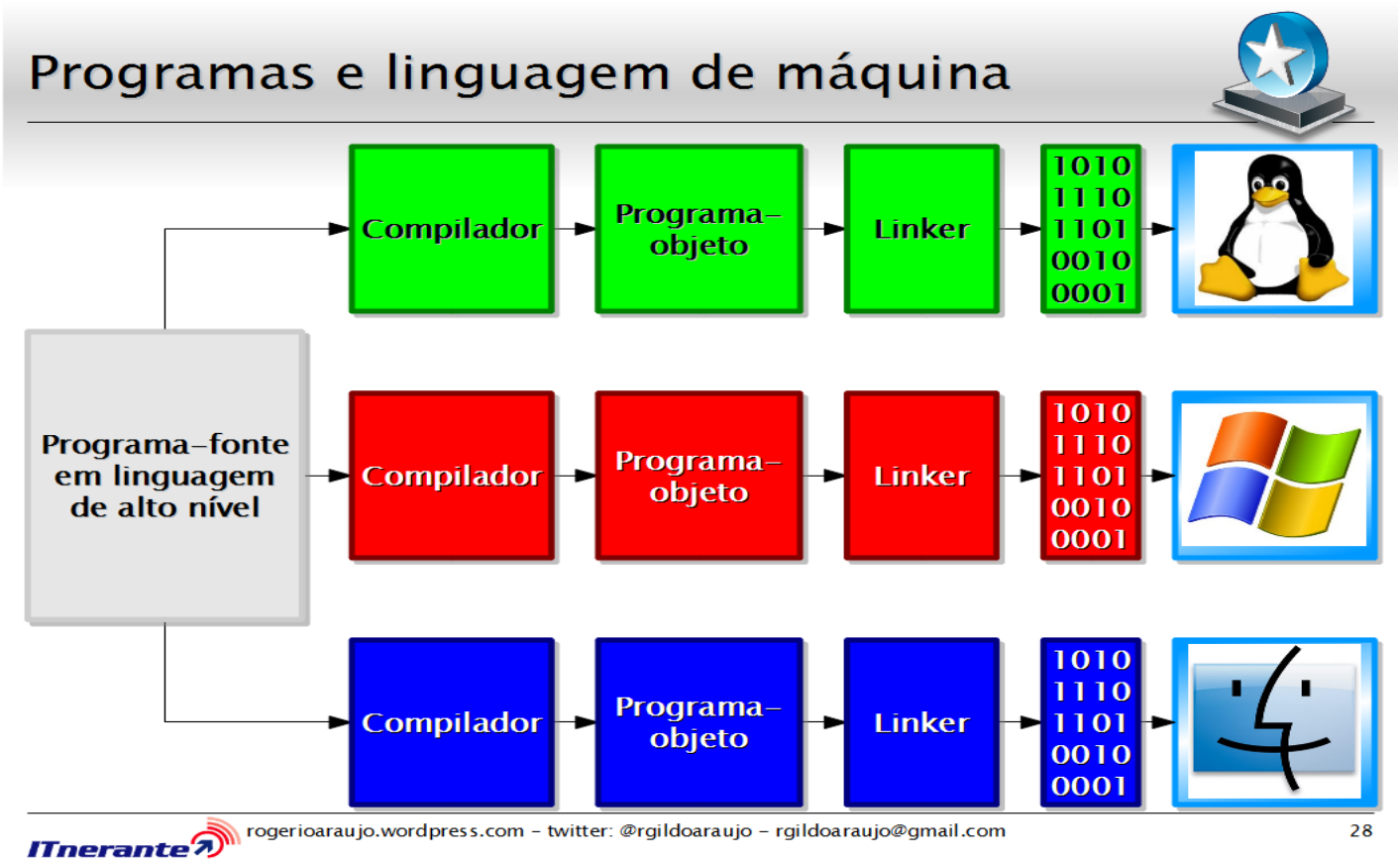
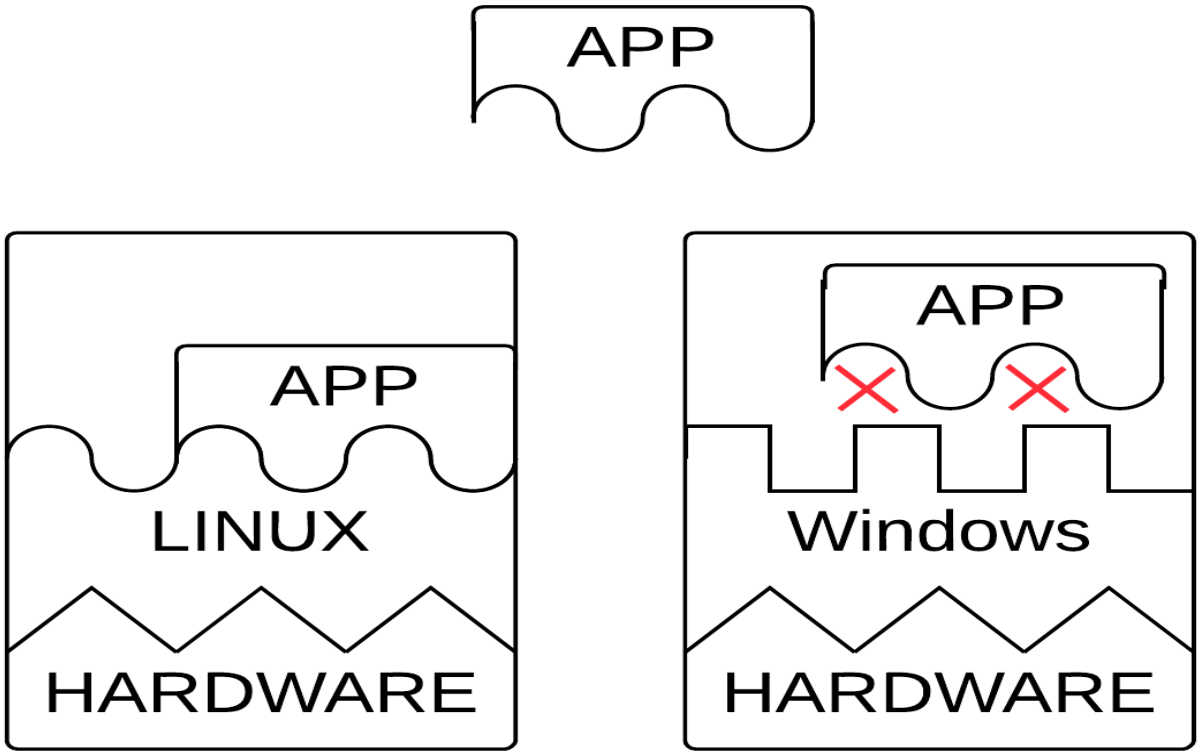


Figura 02 – Exemplo de compilação – programa na linguagem C

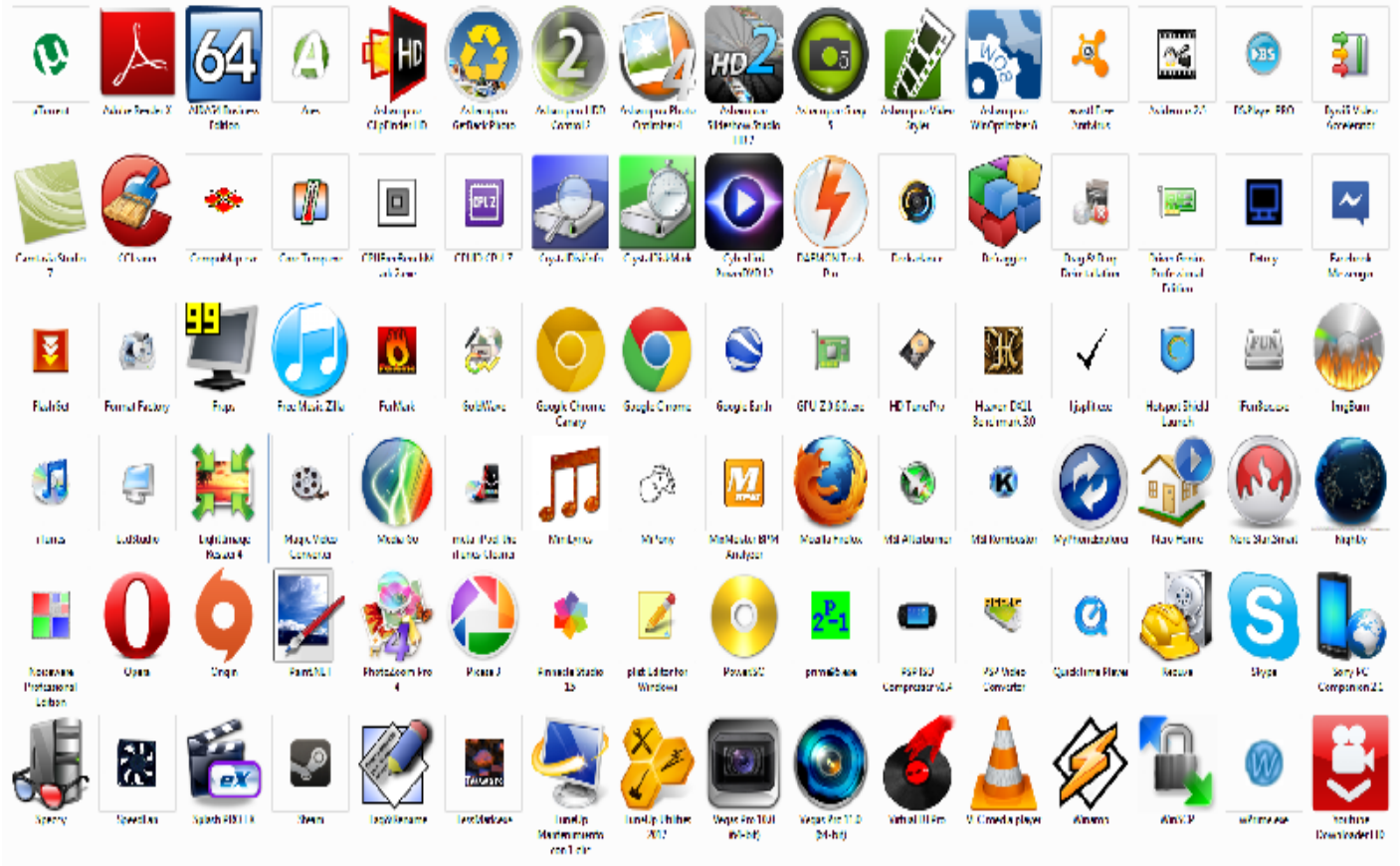
Válido para qualquer Sistema Operacional (SO):



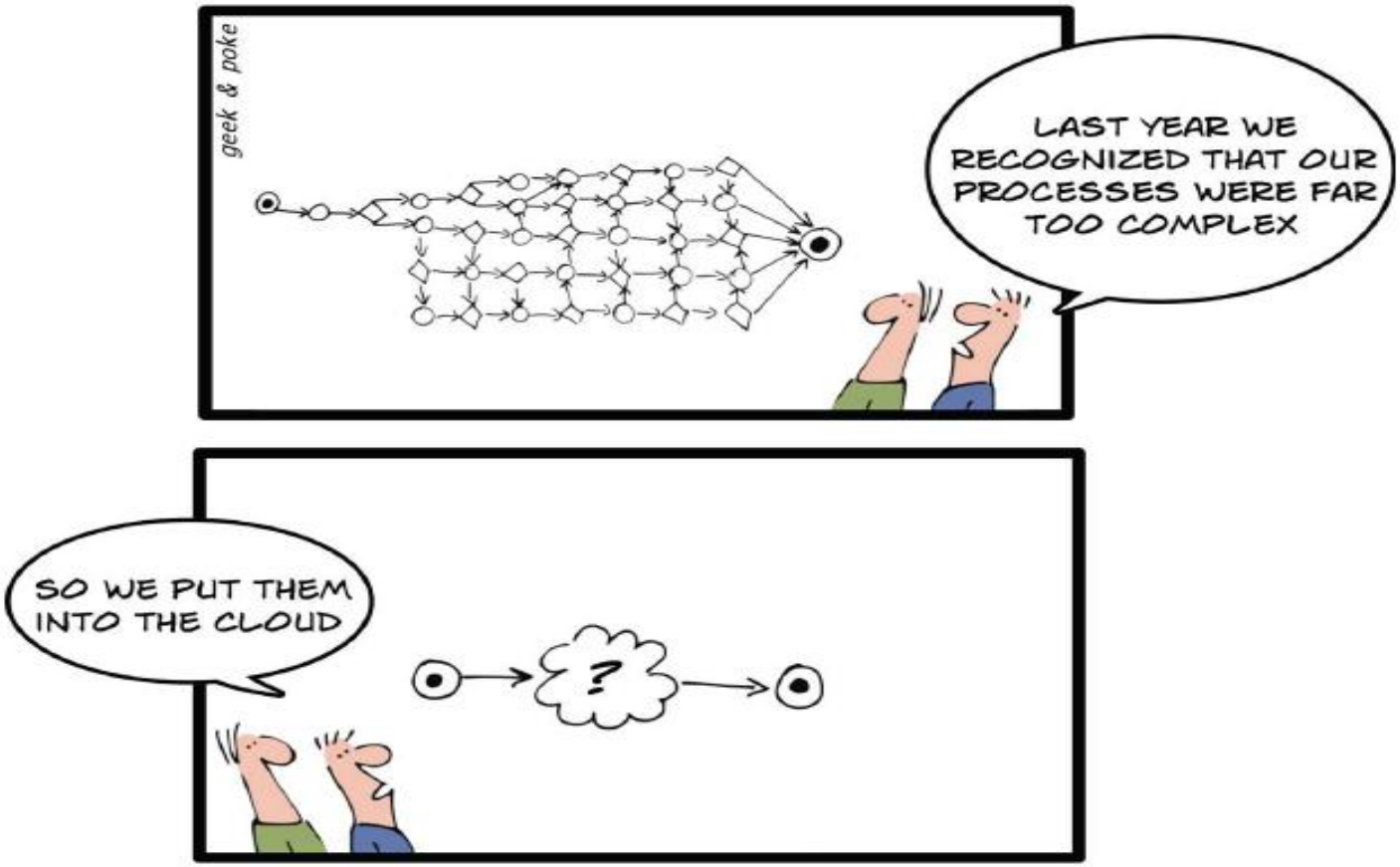
Pois vamos fazer aplicações (fontes e executáveis) usuários!



Programas aos usuários!



Epílogo


geek & poke
LAST YEAR WE RECOGNIZED THAT OUR PROCESSES WERE FAR TOO COMPLEX
SO WE PUT THEM INTO THE CLOUD
?