

LÓGICA DE PROGRAMAÇÃO PARA ENGENHARIA INTRODUÇÃO À ORGANIZAÇÃO DE COMPUTADORES

Prof. Dr. Daniel Caetano

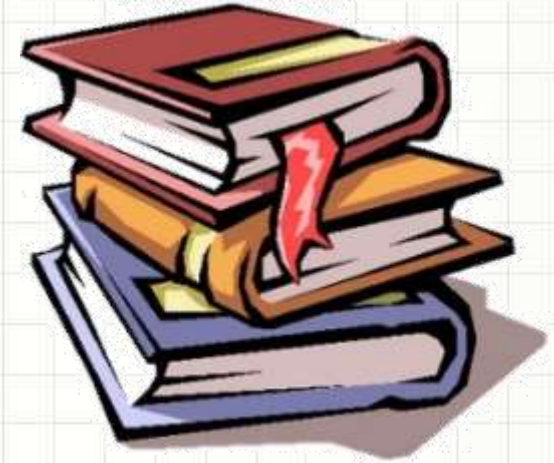
2018 - 1

Objetivos

- Apresentar o funcionamento do computador
- Apresentar a função da memória e dispositivos de entrada e saída
- Compreender o armazenamento de dados na memória
- **Atividade Aula 2 - SAVA!**



Material de Estudo



Material

Acesso ao Material

Notas de Aula e
Apresentação

<http://www.caetano.eng.br/>
(Lógica de Programação para Eng. – Aula 2)

Material Didático

Lógica de Programação, págs 19-25

Aula Online

Aula 1

Biblioteca Virtual

“Arquitetura e Organização dos Computadores”, págs
1 a 50, 99 a 121, 191 a 201 e 427 a 440.

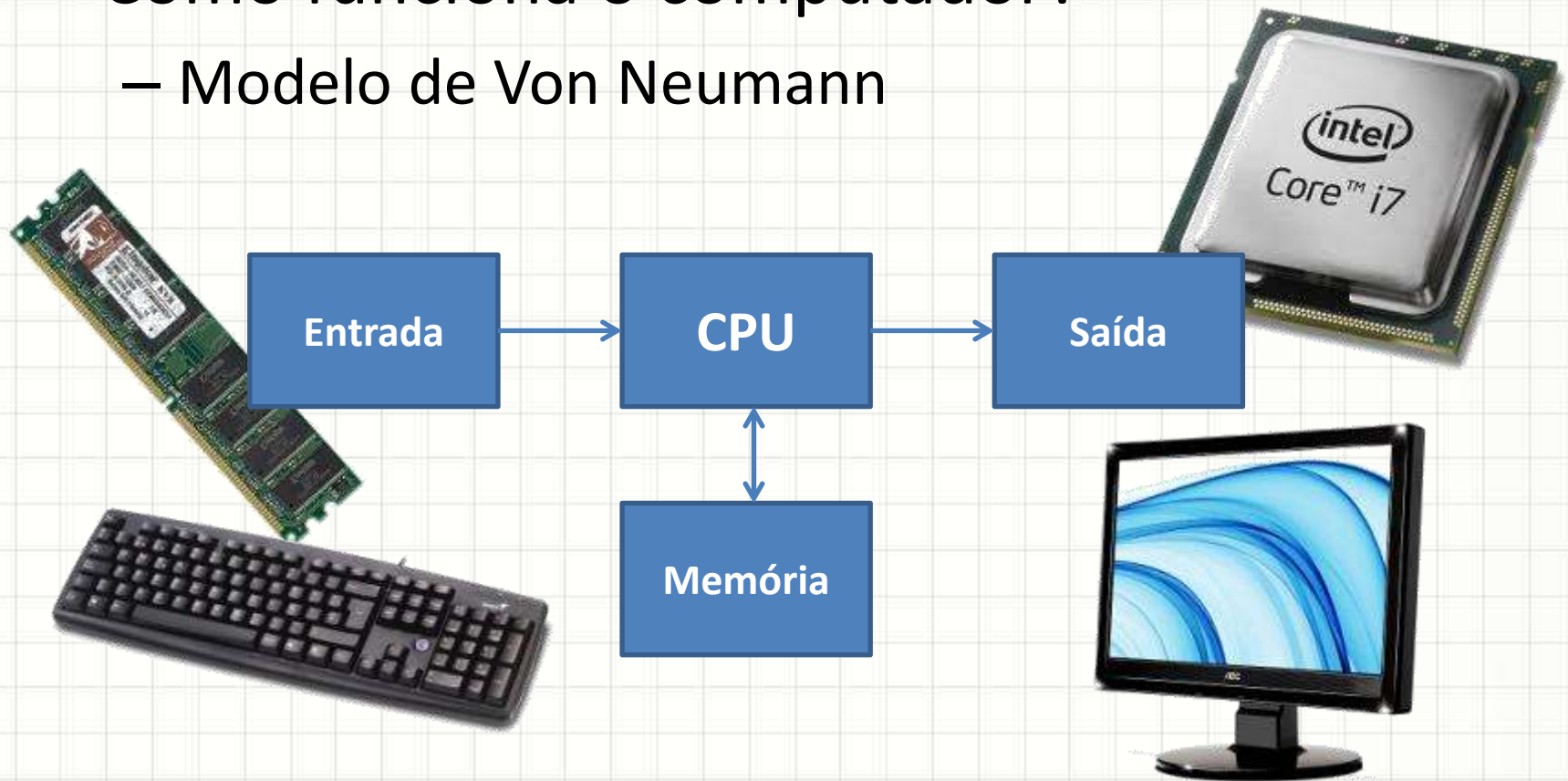
“Organização Estruturada de Computadores”, págs 1
a 7, 29 a 32, 39 a 43, 58, 73 a 74 e 397 a 408.



O COMPUTADOR

Entendendo o Computador

- Usar ferramenta: entender a ferramenta
- Como funciona o computador?
 - Modelo de Von Neumann

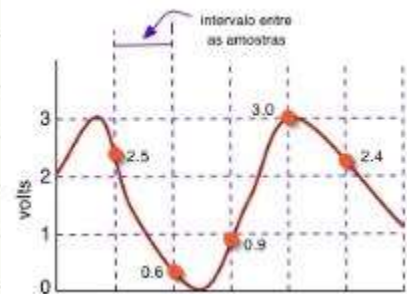


Entendendo o Computador

- **CPU**: Coordena todo o funcionamento do computador e realiza cálculos numéricos
- **Unidade de Entrada**: Recebe dados (números) externos para processamento
- **Unidade de Memória**: Armazena dados (números) para uso posterior
- **Unidade de Saída**: Exibe dados (números) para o usuário, após processamento
- **O computador só entende números!**

Dispositivos de Entrada e Saída

- **Dispositivos de Entrada**
- **Leitura:** converter informações externas (usualmente fornecidas pelo usuário) em números para o computador

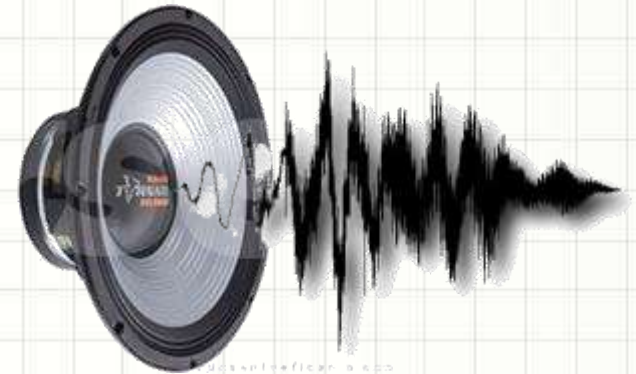
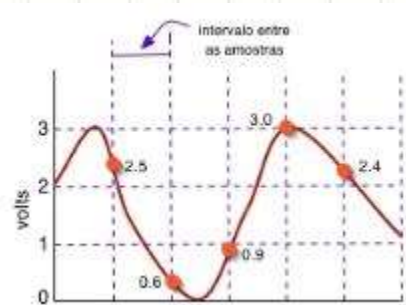


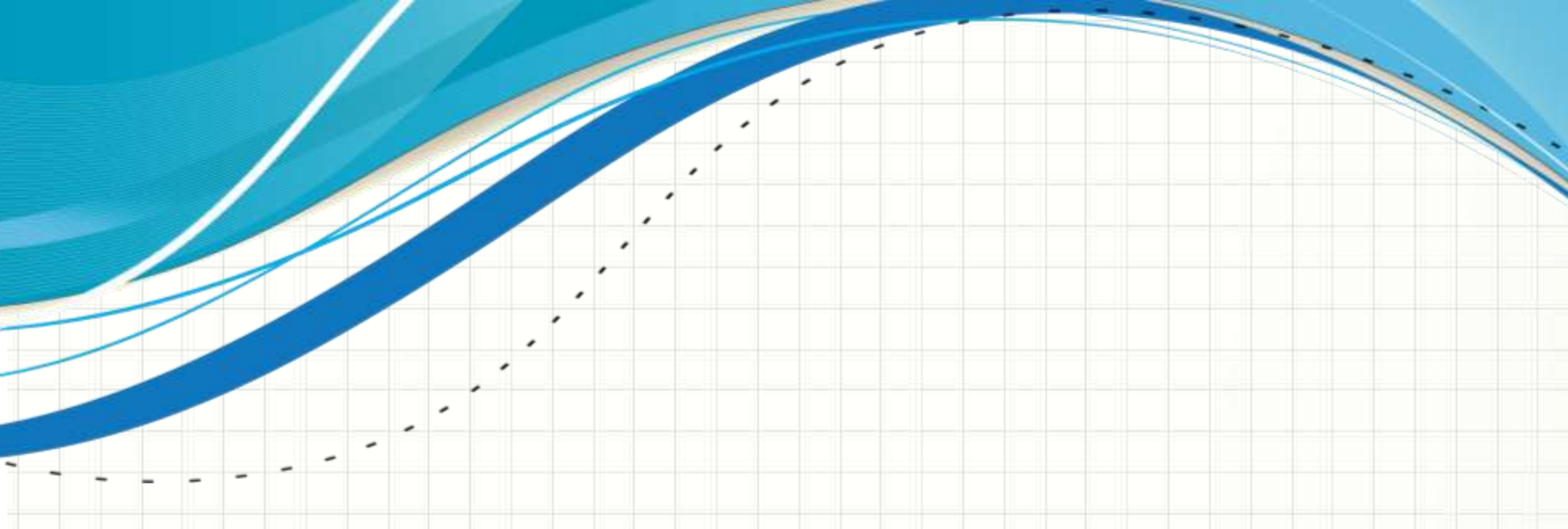
10011001
10011110
10101100
10111001
11001010
11001111
11010011
10111101

Dispositivos de Entrada e Saída

- **Dispositivos de Saída**
- **Escrita:** converter números fornecidos pelo computador em informações para o usuário

10011001
10011110
10101100
10111001
11001010
11001111
11010011
10111101

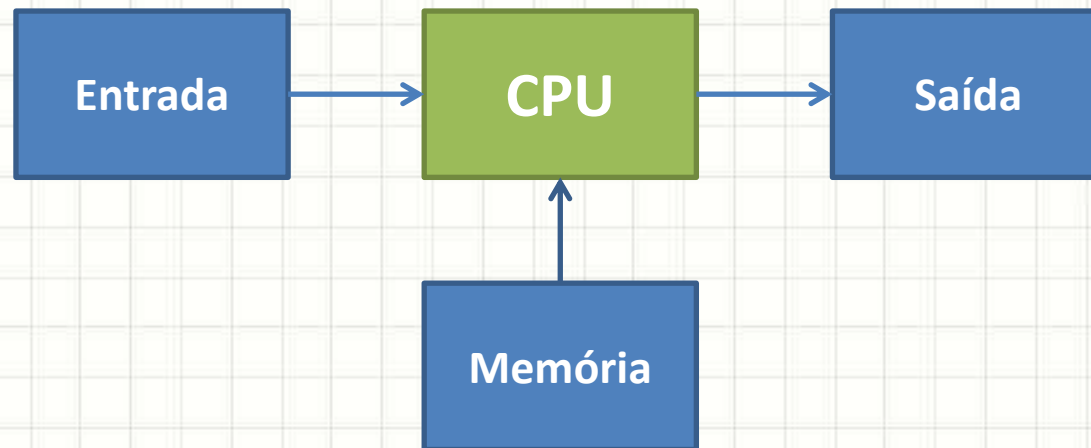




FUNCIONAMENTO DA CPU

Funcionamento da CPU

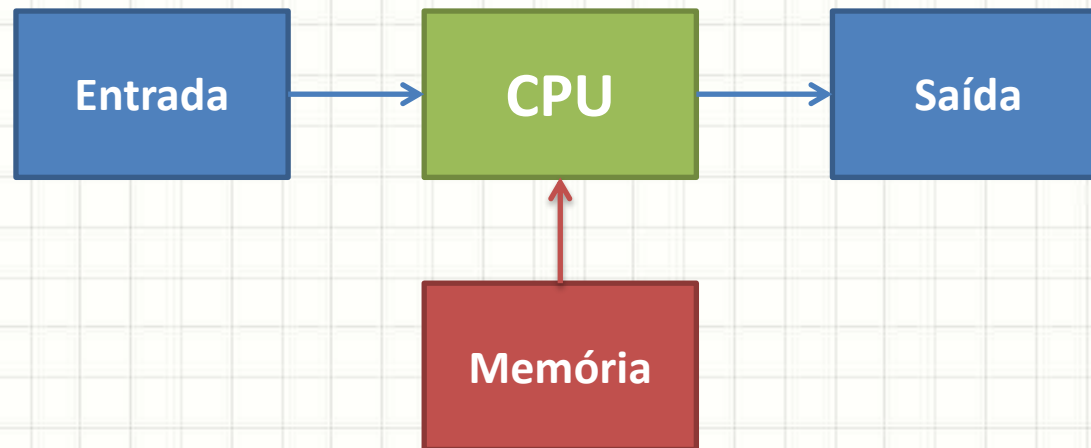
- Como a CPU coordena essas partes para produzir resultado útil?



Funcionamento da CPU

- **1. Busca Instrução**

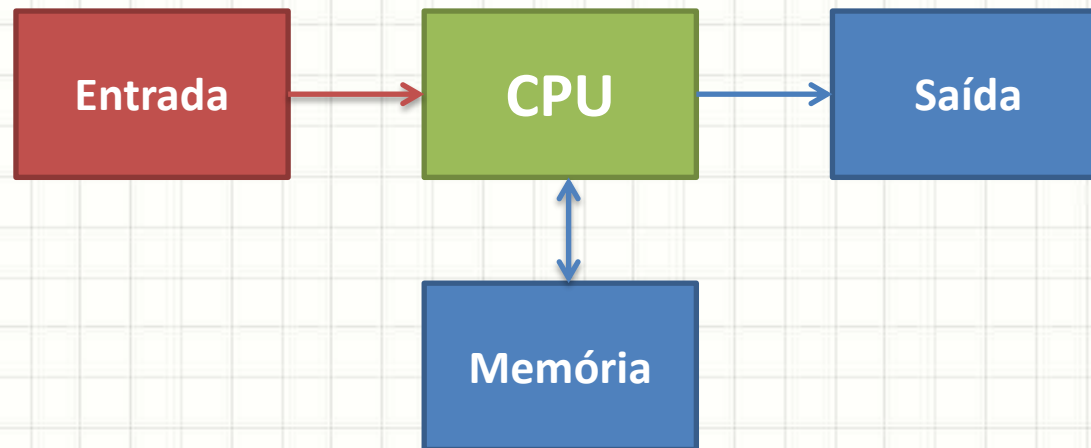
- CPU lê a memória em busca do que deve fazer
- Instrução lida: **leia entrada**



Funcionamento da CPU

- **2. Lê entrada**

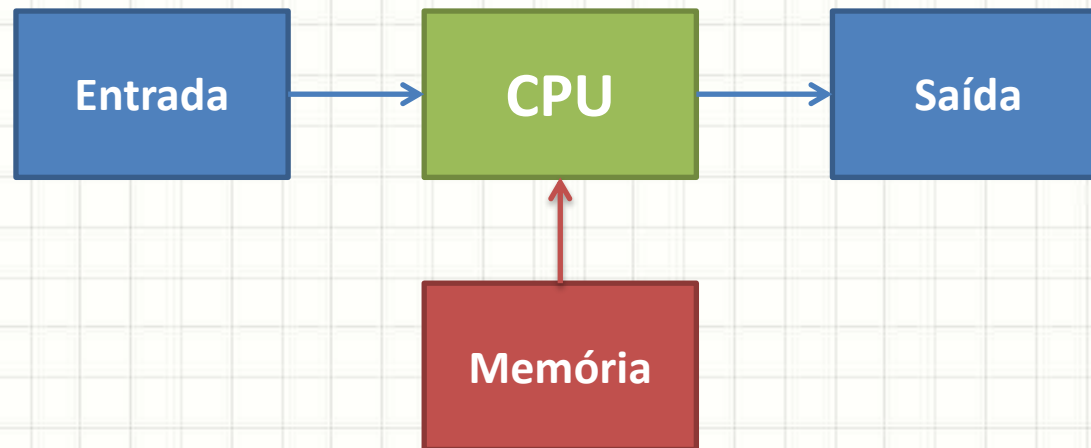
- CPU lê entrada, recebendo um dado numérico
- Dado lido: 33



Funcionamento da CPU

- **3. Busca Instrução**

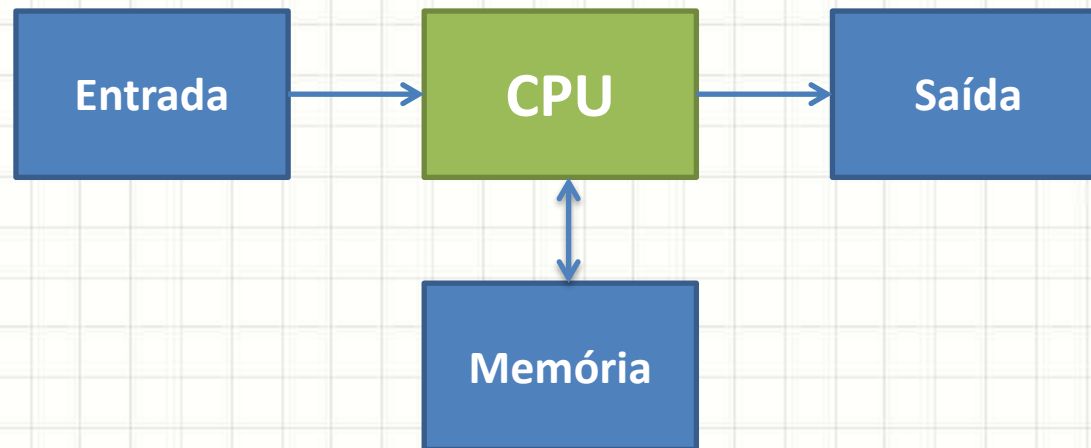
- CPU lê a memória em busca do que deve fazer
- Instrução lida: **multiplique por dois**



Funcionamento da CPU

- **4. Cálculo**

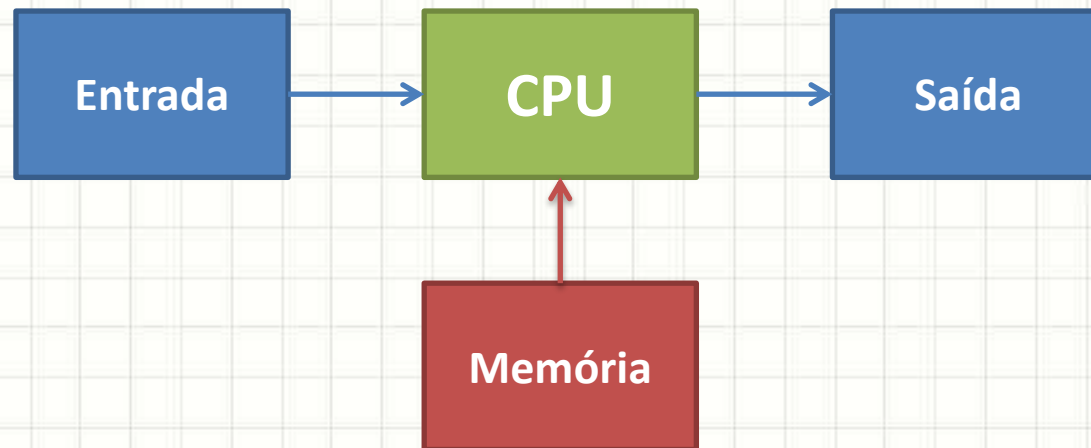
- CPU realiza operação de cálculo
- $33 * 2 = 66$



Funcionamento da CPU

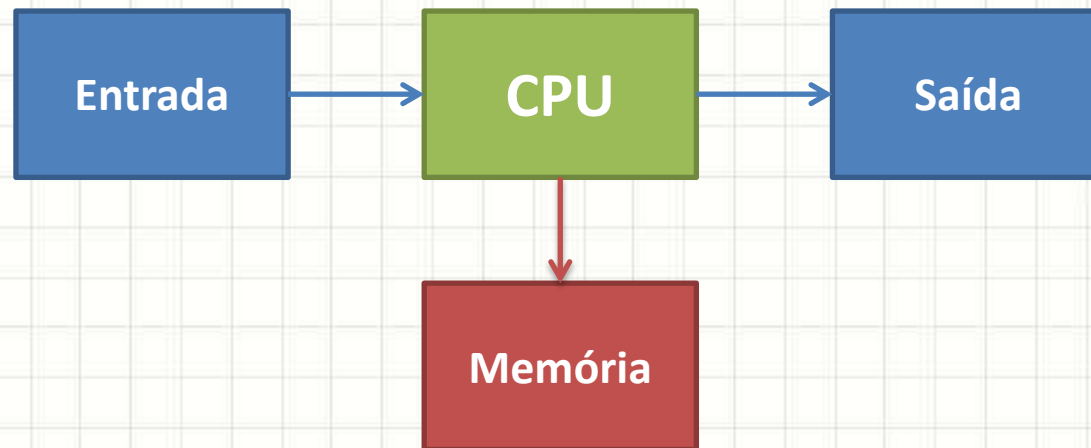
- **5. Busca Instrução**

- CPU lê a memória em busca do que deve fazer
- Instrução lida: **armazenar resultado**



Funcionamento da CPU

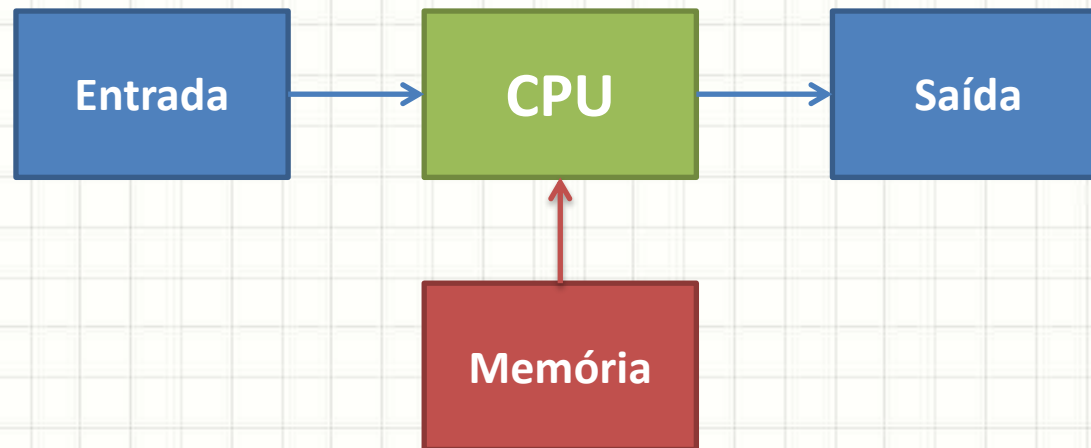
- **6. Armazena dado**
 - Guarda o dado na memória
 - Dado armazenado: 66



Funcionamento da CPU

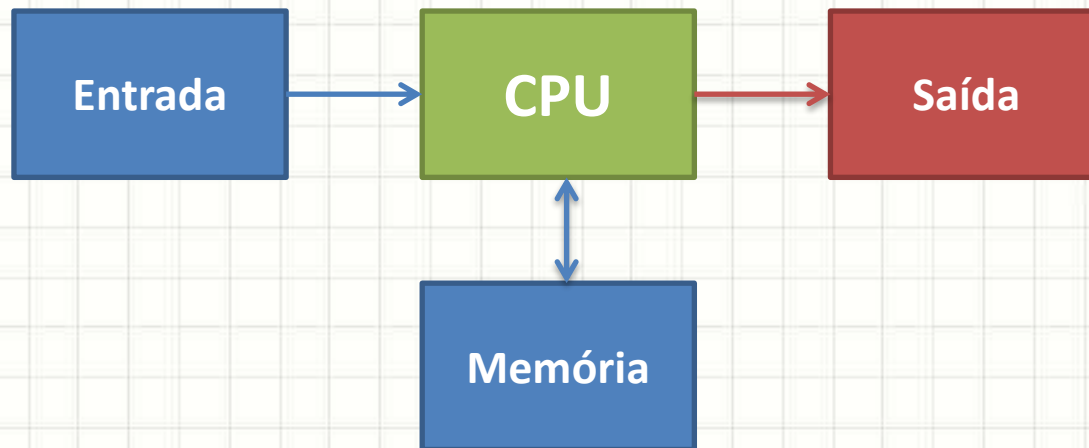
- **7. Busca Instrução**

- CPU lê a memória em busca do que deve fazer
- Instrução lida: **imprime dado**



Funcionamento da CPU

- **8. Saída de Dados**
 - CPU escreve na saída
 - O número 66 é impresso



Funcionamento da CPU

- **8. Saída de Dados**
 - CPU escreve na saída

E assim sucessivamente...

Memória

Funcionamento da CPU

- A CPU tem duas partes principais:
 - Unidade de Controle: coordena a execução
 - Unidade Lógica Aritmética: realiza os cálculos
- A UC é quem acessa a memória RAM
 - Analogia: usuário de uma calculadora
- A ULA é quem faz cálculos
 - Analogia: a calculadora em si



FUNCIONAMENTO DA MEMÓRIA PRINCIPAL

A Memória Principal

- Cada gaveta é chamada **posição de memória**
- Cada gaveta possui um número que a identifica, chamado **endereço de memória**
- Em cada uma das gavetas, podemos guardar um número fixo de “dígitos”



A Memória Principal

- A memória principal (RAM) é...



A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>								

A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>								

- Armazenemos o valor **255** na posição de memória cujo endereço é **3**

A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>								

- Armazenemos o valor **255** na posição de memória cujo endereço é **3**

A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>				255				

- Armazenemos o valor **255** na posição de memória cujo endereço é **3**

A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>				255				

- Agora, armazenemos o valor **7** na posição de memória cujo endereço é **5**

A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>				255				

- Agora, armazenemos o valor **7** na posição de memória cujo endereço é **5**

A Memória Principal

- Quando queremos guardar um número na memória, temos dizer em qual **posição de memória** ele deve ser armazenado, usando para isso o **endereço de memória**

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>				255		7		

- Agora, armazenemos o valor **7** na posição de memória cujo endereço é **5**

A Memória Principal

- Tomemos, agora, uma memória cheia

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>	10	57	0	255	100	7	10	2

A Memória Principal

- Tomemos, agora, uma memória cheia

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>	10	57	0	255	100	7	10	2

- Qual é o valor na posição de memória cujo endereço é **7**?

A Memória Principal

- Tomemos, agora, uma memória cheia

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>	10	57	0	255	100	7	10	2

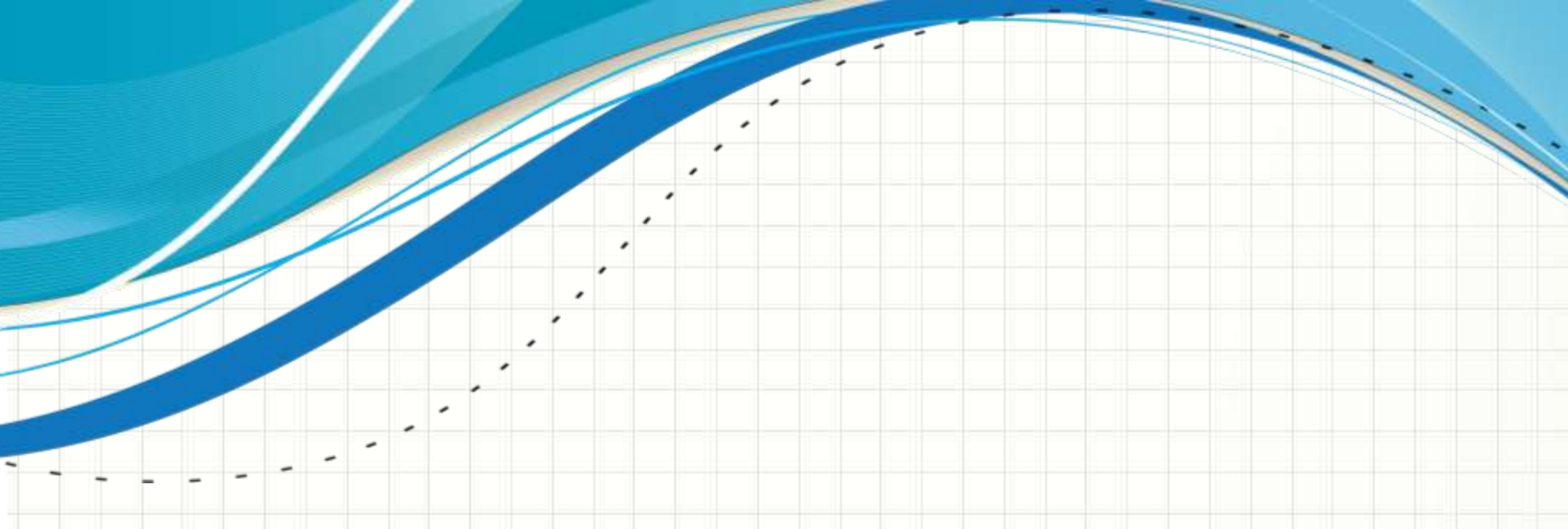
- Qual é o valor na posição de memória cujo endereço é **7**?

A Memória Principal

- Tomemos, agora, uma memória cheia

<u>Endereço</u>	0	1	2	3	4	5	6	7
<u>Valor</u>	10	57	0	255	100	7	10	2

- Qual é o valor na posição de memória cujo endereço é **7**?
- O valor é **2**!
- **Atenção!**
O computador **NÃO** apaga a memória sozinho!



**COMO OS NÚMEROS SÃO
ARMAZENADOS: A LINGUAGEM
DO COMPUTADOR**

O Que o Computador Entende?

- Já vimos que o computador entende apenas números...
- ...mas ele entende os nossos números?
- Infelizmente... não.
- O computador um *dialeto* chamado “binário”:
0101001010111b
- Mas o que isso significa?

Humanos x Processadores

- Humanos aprendem a contar com os dedos;
- Como temos DEZ dedos nas mãos, usamos naturalmente os números DECIMAIS
- Isso significa que cada **dígito** do número será “ocupado” com um de 10 símbolos diferentes:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



Humanos x Processadores

- Como indicar n^{os} decimais para o processador?

FIOS



Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



0

Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



2

Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



4

Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



5

Humanos x Processadores

- Como indicar n^{os} decimais para o processador?



94.614

Como saber isso?

- Imaginemos que cada fio representa um “dígito” numérico, chamado **bit**
- Esse fio pode estar **desligado ou ligado**
- Associando o símbolo **0** ao fio “desligado” e **1** ao fio “ligado”, representa-se o “número” em um formato que o computador entende:

10111000110010110

94.614



Como saber isso?

- Imaginemos que cada fio representa um “dígito” numérico, chamado **bit**
- Esse fio pode estar **desligado** ou **ligado**

10111000110010110b = 94.614

10111000110010110

94.614



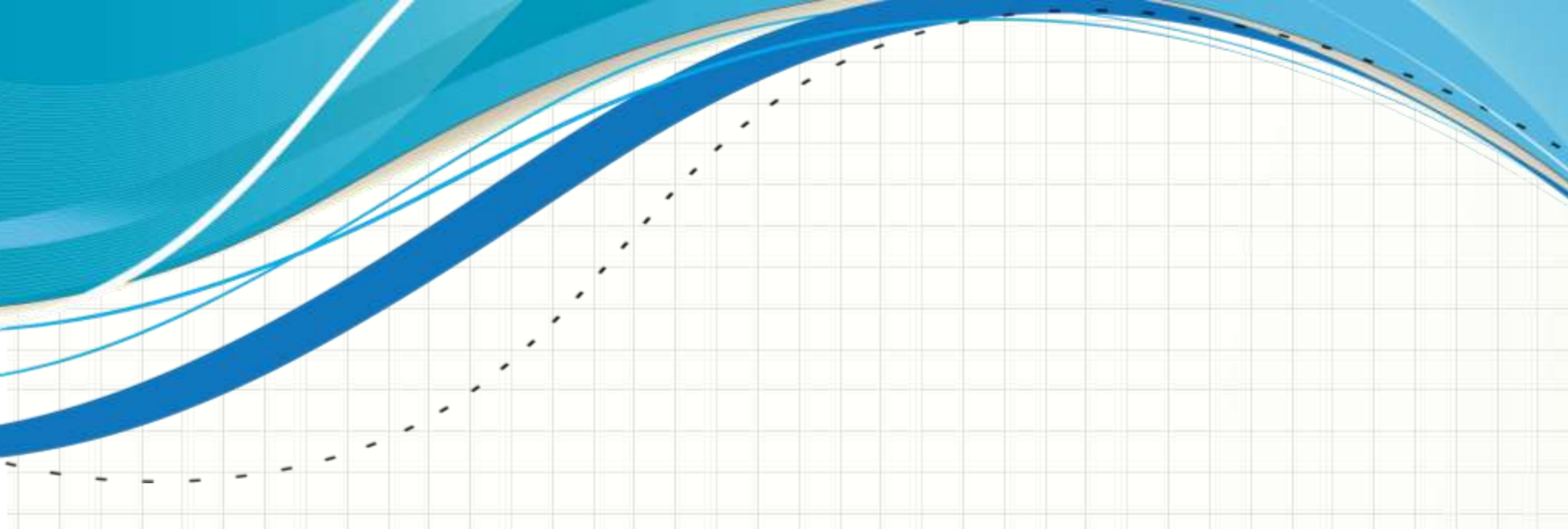
Os Números Binários

- Como cada **bit** pode ser apenas 0 ou 1...
...o nome dessa representação é “**binária**”.
- Um único bit armazena pouca informação
- Usualmente, os bits aparecem agrupados



Os Números Binários

- Agrupamentos comuns e seus nomes
 - 4 bits: **nibble**
 - Suficiente para dígito de 0 a 9
 - 8 bits: **byte**
 - Suficiente para guardar os 256 símbolos mais comuns (incluindo letras de um texto)
 - 16 bits: **word** (palavra)
 - Suficiente para armazenar a maioria das instruções de um computador)
 - 32 bits: **double word** (palavra dupla)
 - Suficiente para guardar endereços de memória comuns



CONVERTENDO DE BINÁRIO PARA QUANTIDADE DECIMAL

Conversão de Binário para Decimal

- Regra prática: construa essa tabela

Multiplicador	32	16	8	4	2	1
101011b	1	0	1	0	1	1

- Limpe os multiplicadores para os quais o valor do dígito é igual a zero

Conversão de Binário para Decimal

- Regra prática: construa essa tabela

Multiplicador	32		8		2	1
101011b	1		1		1	1

- Limpe os multiplicadores para os quais o valor do dígito é igual a zero
- Some os multiplicadores que sobraram!

$$32 + 8 + 2 + 1 = \mathbf{43}$$



**CONVERTENDO DE DECIMAL
PARA REPRESENTAÇÃO
BINÁRIA**

Conversão de Decimal para Binário

- Regra prática: divida sucessivamente por 2, construindo o número binário da direita para a esquerda.
- Se a divisão for exata, acrescenta-se 0 ao número binário
- Se a divisão for “quebrada”, acrescenta-se 1 ao número binário e “joga-se fora” a parte
- Repete-se até que o valor a dividir seja 0
- Observe!

Conversão D→B

- Regra prática: converter 13 para binário

1b

- $13/2 = 6,5$ **Fracionário!**

Conversão $D \rightarrow B$

- Regra prática: converter 13 para binário

01b

- $13/2 = 6,5$
- $6/2 = 3,0$

Exato!

Conversão D→B

- Regra prática: converter 13 para binário

101b

- $13/2 = 6,5$
- $6/2 = 3,0$
- $3/2 = 1,5$

Fracionário!

Conversão $D \rightarrow B$

- Regra prática: converter 13 para binário

1101b

- $13/2 = 6,5$
- $6/2 = 3,0$
- $3/2 = 1,5$
- $1/2 = 0,5$

Fracionário!

Conversão D→B

- Regra prática: converter 13 para binário

1101b

- $13/2 = 6,5$
- $6/2 = 3,0$
- $3/2 = 1,5$
- $1/2 = 0,5$
- 0

Fim!

Conversão D→B

- Regra prática: converter 13 para binário

1101b

- 13/2
- 6/2 :
- 3/2 :
- 1/2 :
- 0

$$13 = 1101b$$

Qual o Problema?

- **Inteiros:** representados em binário exato
- **Fracionários:** nem sempre binários exatos
- Ex.: 0,1 decimal, em binário fica...:

0,000110011001100110011001100110011001100110011...

- Mas o computador guarda infinitas casas?
- **NÃO!**



COMO GUARDAR OUTROS DADOS NA MEMÓRIA?


Outros Dados na Memória

- Só números binários... 0101001010111b
- Seu significado depende da interpretação!
 - Números inteiros
 - Números fracionários
 - Letras
 - Imagens
 - ...
- Interpretação depende de uma **convenção**

Representação de Caracteres

- Letras: Padrões de codificação

- ASCII
- UTF-8
- UTF-16



Código	Caracteres	Código	Caracteres	Código	Caracteres
32	[space]	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[123	{
60	<	92	\	124	
61	=	93]	125	}
62	>	94	^	126	~
63	?	95	_	127	[backspace]

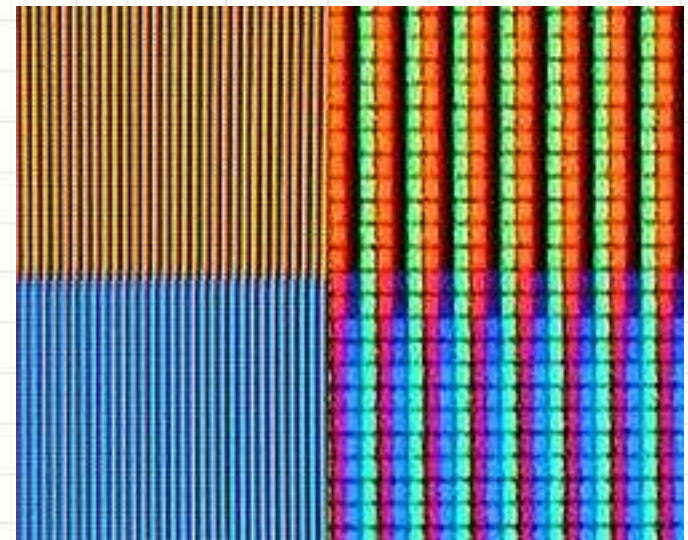
Representação de Imagens

- Pontos: RGB (Vermelho, Verde, Azul)



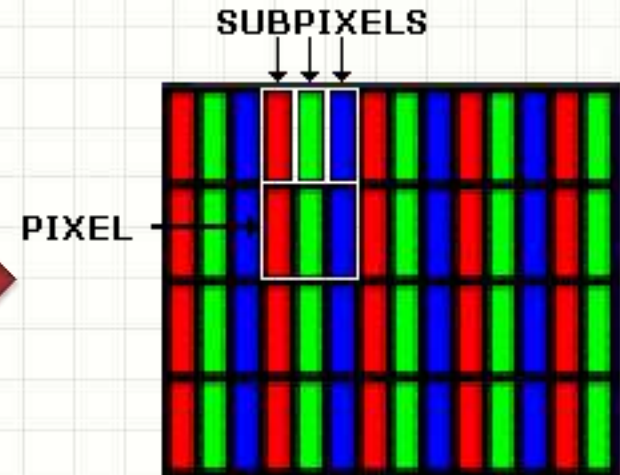
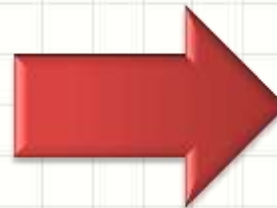
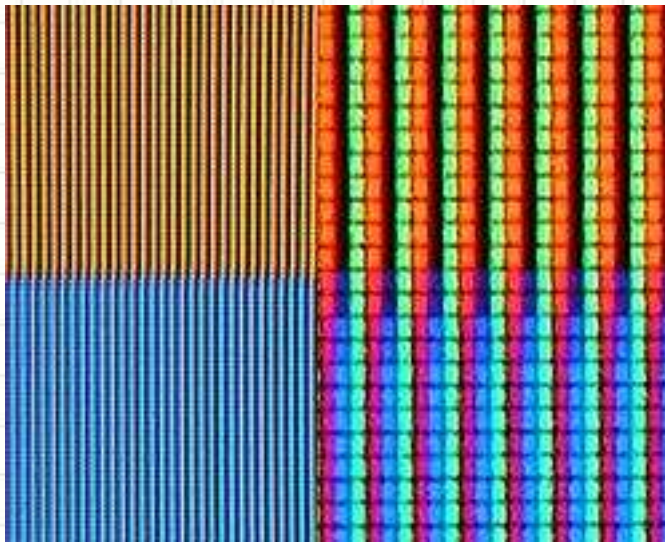
Representação de Imagens

- Pontos: RGB (Vermelho, Verde, Azul)



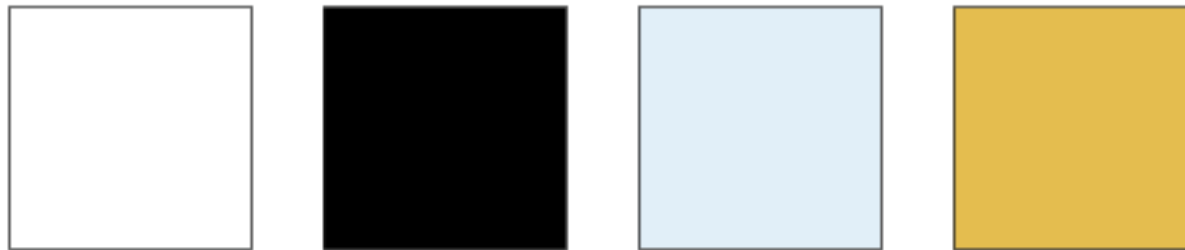
Representação de Imagens

- Pontos: RGB (Vermelho, Verde, Azul)



Representação de Imagens

- Pontos: RGB (Vermelho, Verde, Azul)



RGB 255, 255, 255 RGB 0, 0, 0 RGB 255, 239, 248 RGB 228, 189, 79





CONCLUSÕES

Resumo

- Elementos e lógica básica do computador
 - Organização da memória
 - Números binários e conversão $B \rightarrow D$ e $D \rightarrow B$
 - Codificações mais complexas
 - **TAREFA**: Exercícios Aula 2
-
- Formalizando a lógica
 - Como sistematizar soluções para problemas



PERGUNTAS?