Lógica de Programação para Engenharia

Introdução à Organização de Computadores

Prof. Dr. Daniel Caetano

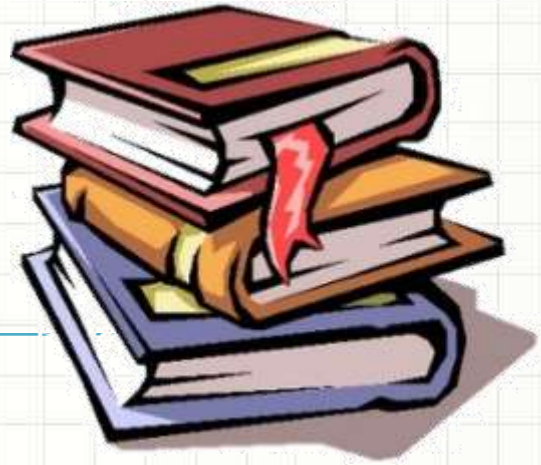
2012 - 1

Objetivos

- Apresentar o funcionamento do computador
- Apresentar a função da memória e dos dispositivos de entrada e saída
- Compreender o armazenamento de dados na memória

- **GRUPOS**?
  - Até o fim da aula!

Material de Estudo

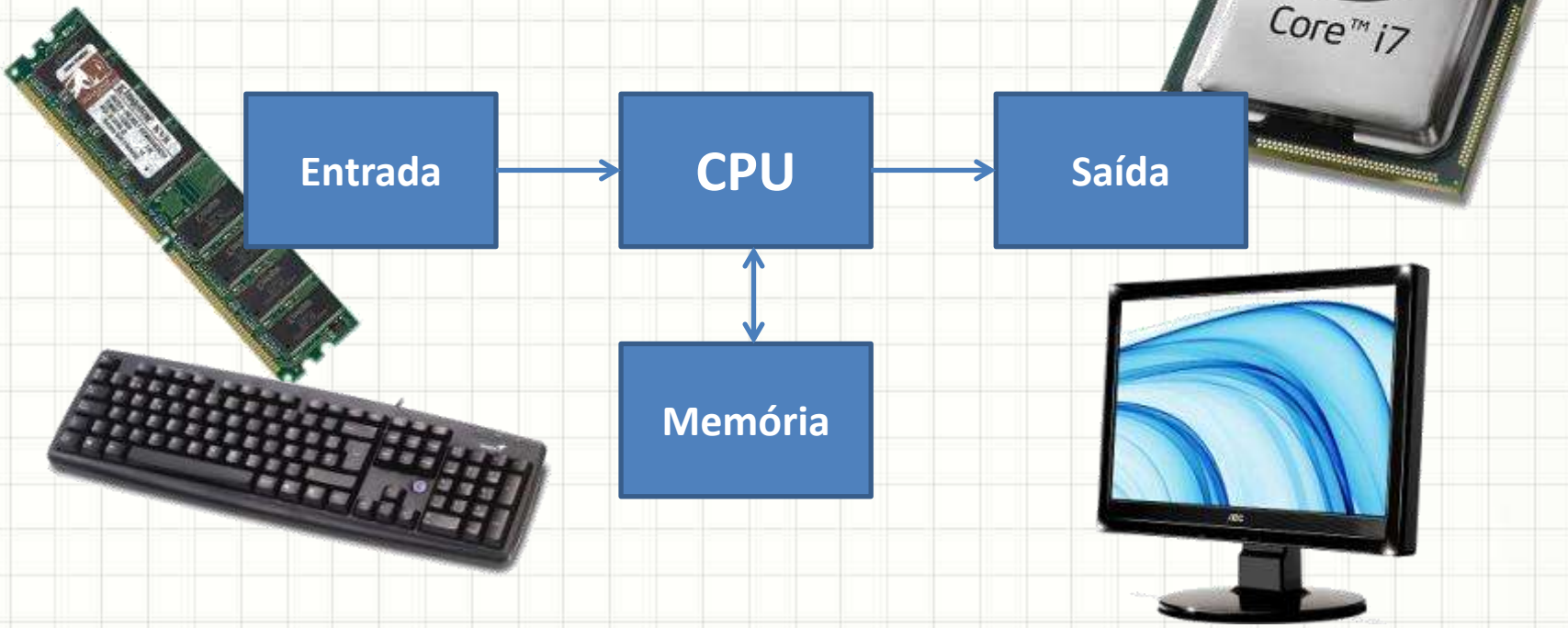


Material	Acesso ao Material
Notas de Aula	http://www.caetano.eng.br/aulas/lpe/ (Aula 2)
Apresentação	http://www.caetano.eng.br/aulas/lpe/ (Aula 2)
Material Didático	-
Arquitetura e Organização dos Computadores	Biblioteca Virtual, páginas 1 a 50, 99 a 121, 191 a 201 e 427 a 440.
Organização Estruturada de Computadores	Biblioteca Virtual, páginas 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

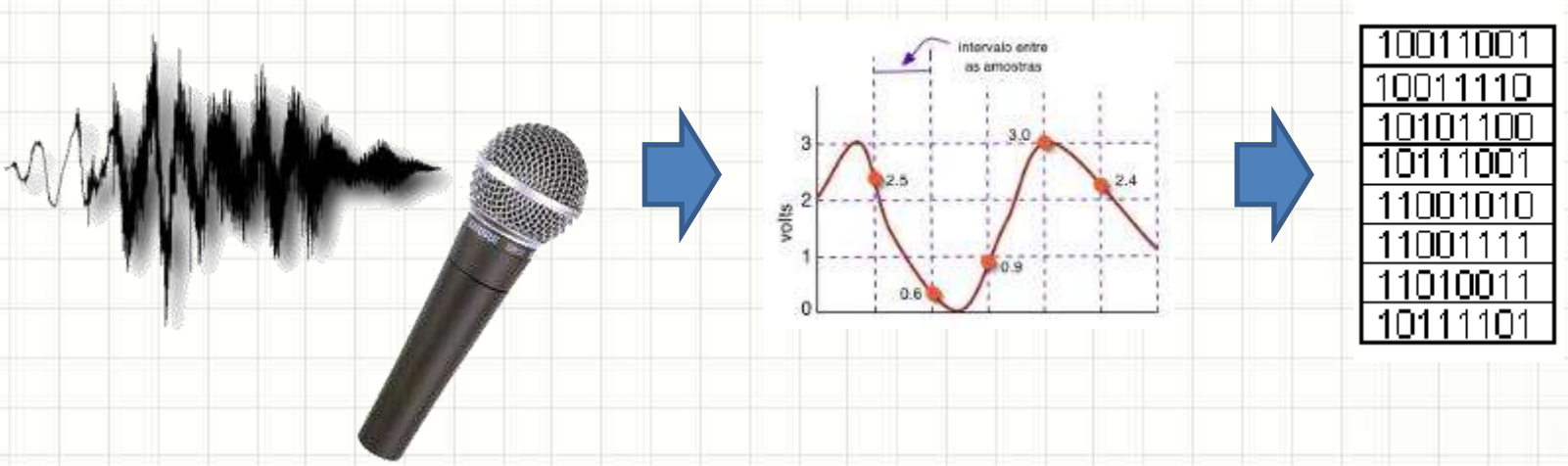
- **CPU**: Coordena todo o funcionamento do computador e realiza cálculos **numéricos**
- **Unidade de Entrada**: Recebe dados (números) externos para processamento
- **Unidade de Memória**: Armazena dados (números) para uso posterior
- **Unidade de Saída**: Exibe dados (números) para o usuário, após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

Entendendo o Computador

- Números...
- Números...
- Números...!?!?
- **O computador só entende números!**

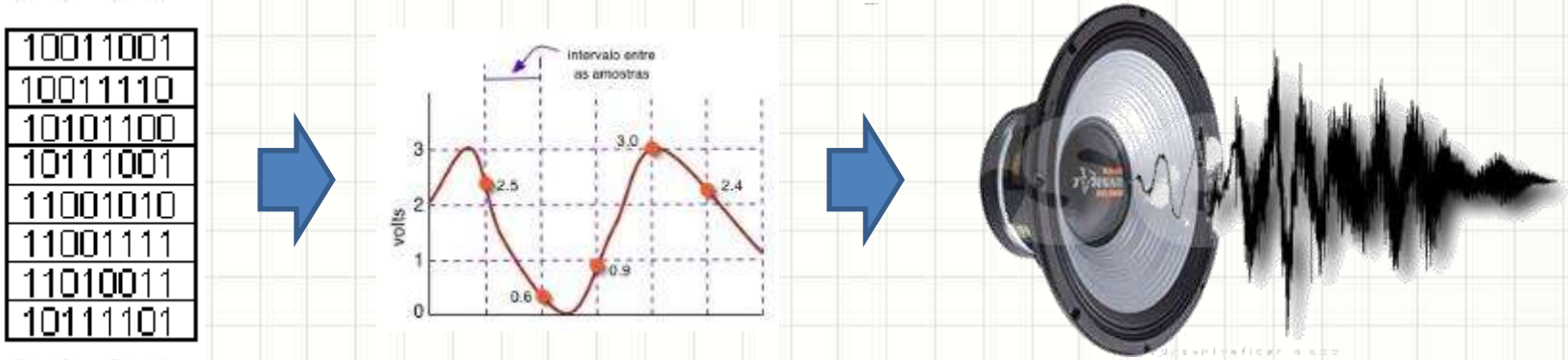
Dispositivos de Entrada e Saída

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



Dispositivos de Entrada e Saída

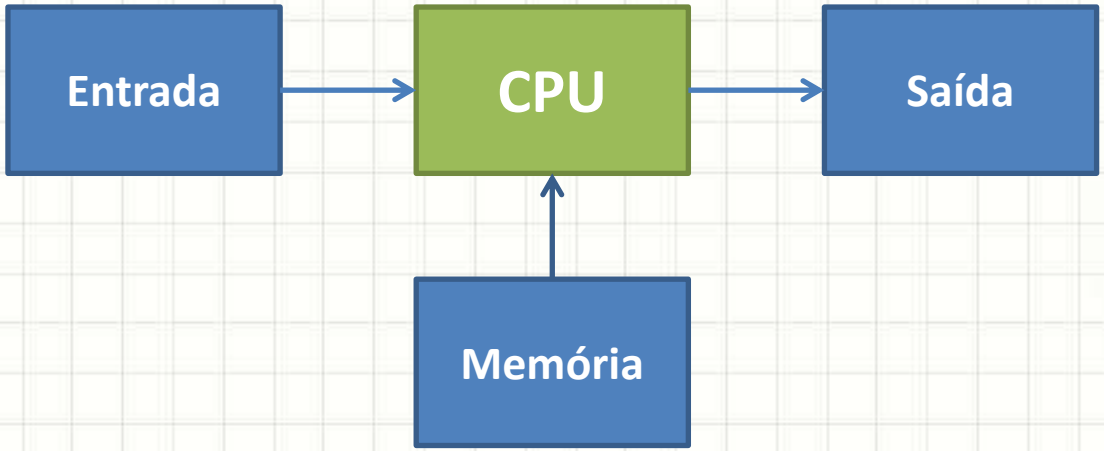
- **Dispositivos de Saída**: convertem números fornecidos pelo computador em informações para o usuário



FUNCIONAMENTO DA CPU

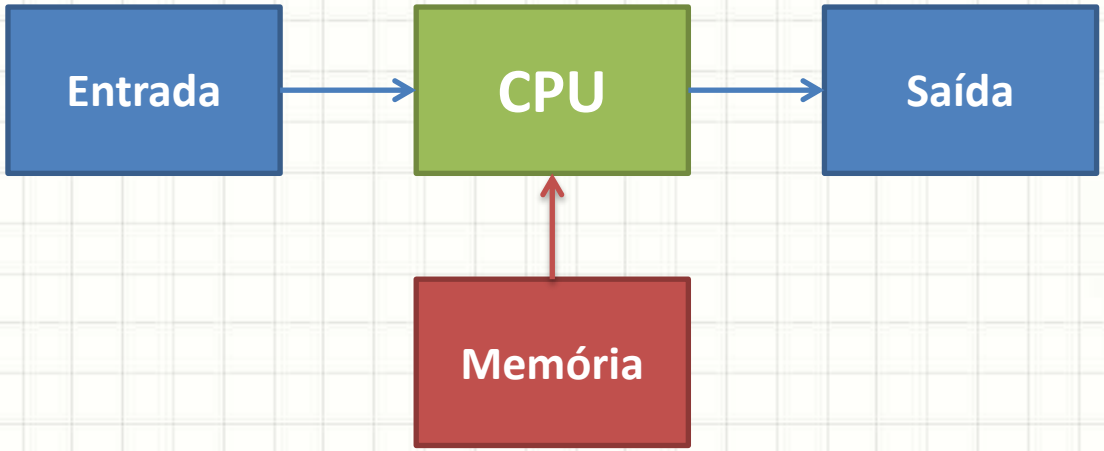
Funcionamento da CPU

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



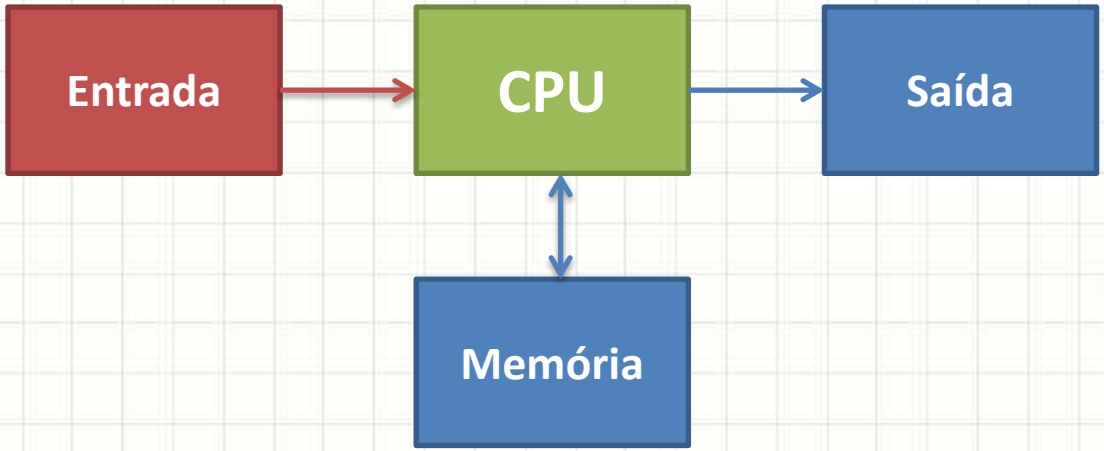
Funcionamento da CPU

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



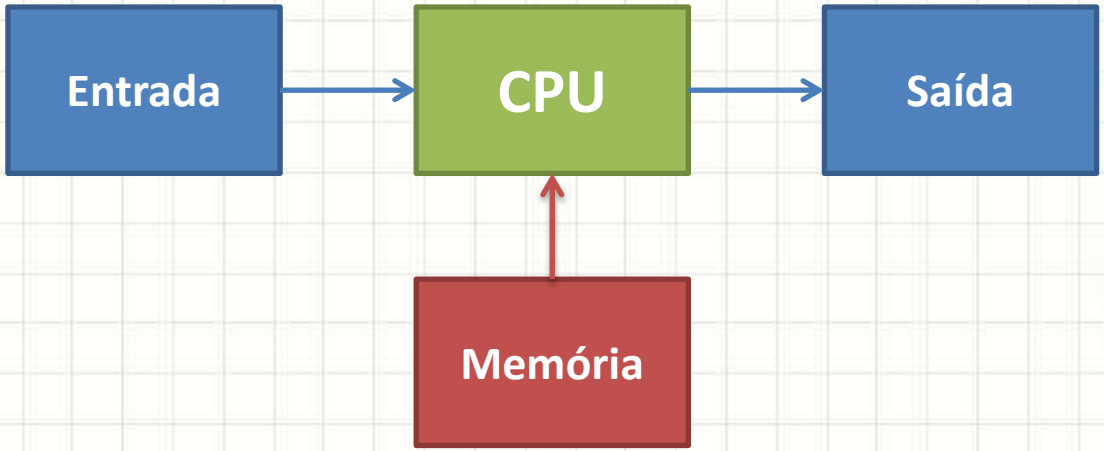
Funcionamento da CPU

- **2. Lê entrada**
  - CPU lê entrada, recebendo um dado numérico
  - Dado lido: 33



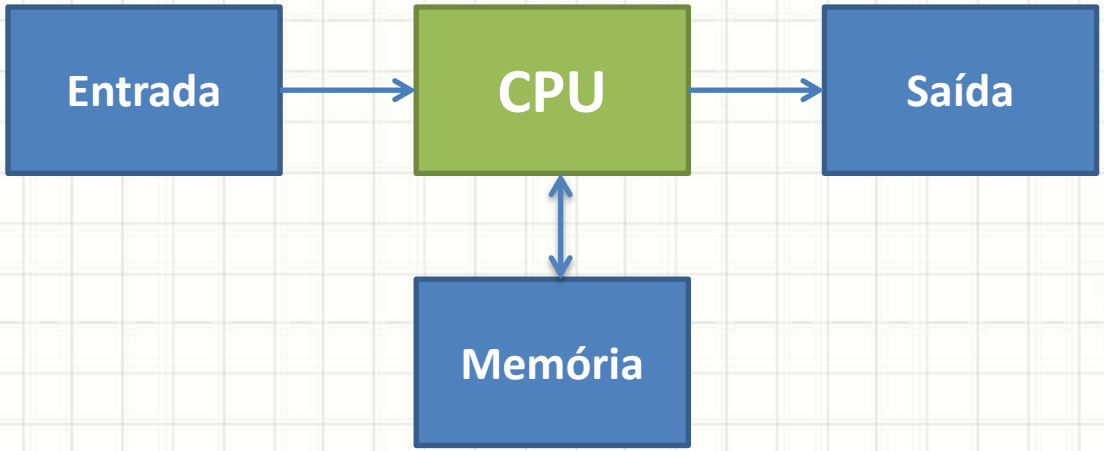
Funcionamento da CPU

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



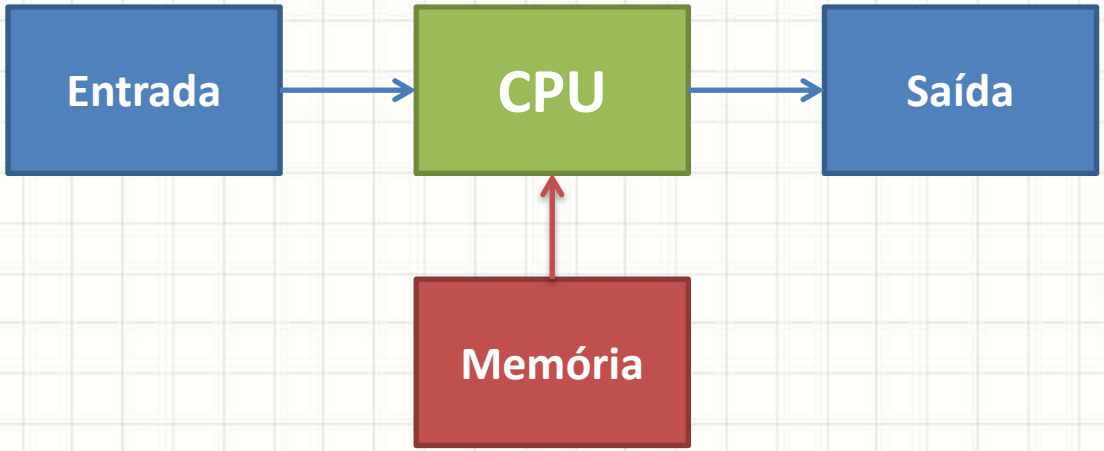
Funcionamento da CPU

- **4. Cálculo**
  - CPU realiza operação de cálculo
  - 33 * 2 = 66



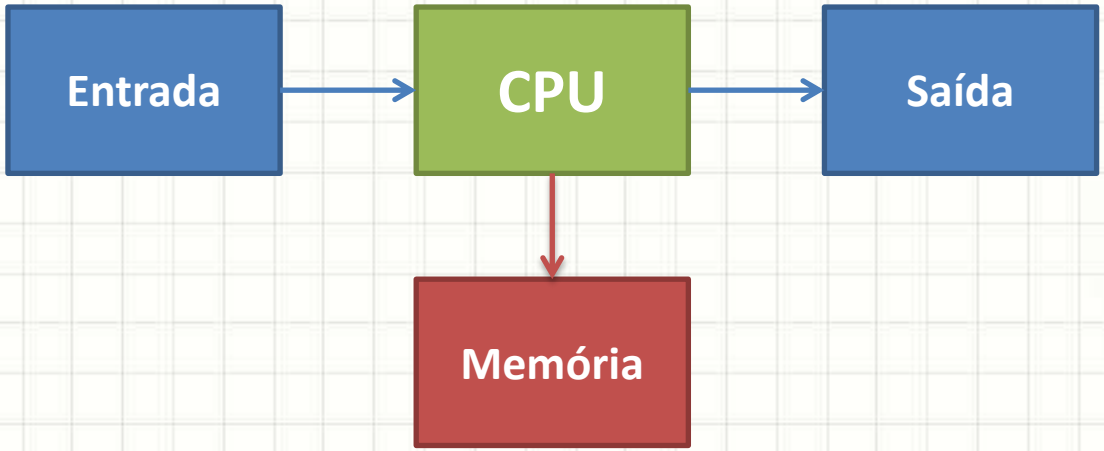
Funcionamento da CPU

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



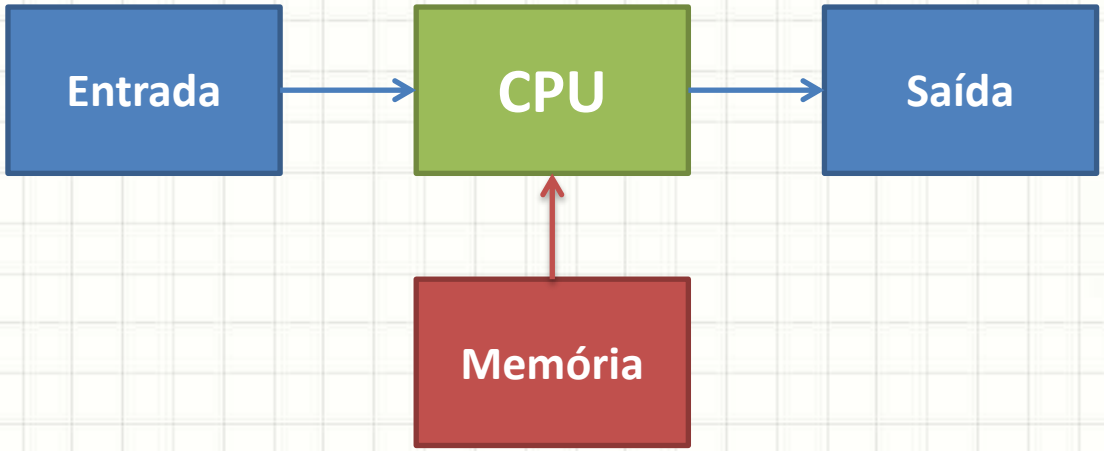
Funcionamento da CPU

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



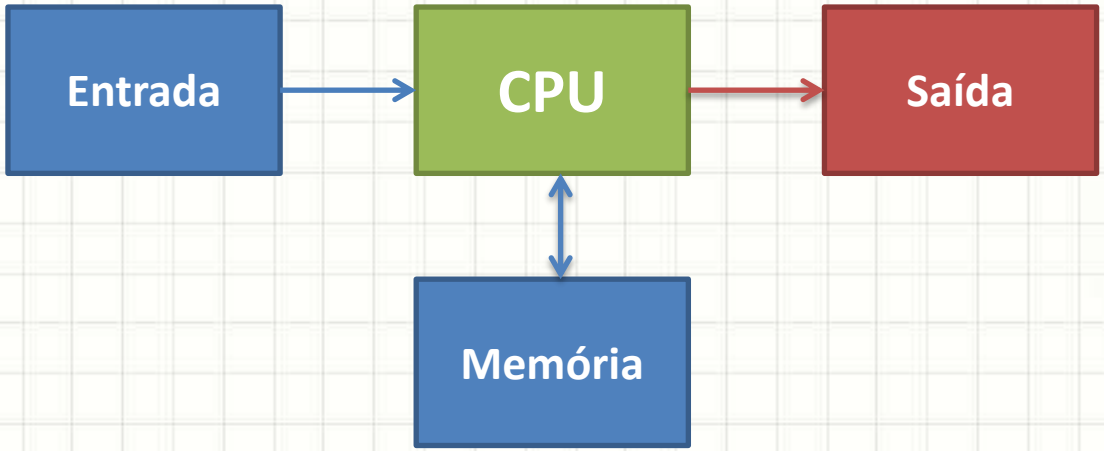
Funcionamento da CPU

- **7. Busca Instrução**
  - CPU lê a memória em busca do que deve fazer
  - Intrusção 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 único número

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**

Endereço	0	1	2	3	4	5	6	7
Valor								

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**

Endereço	0	1	2	3	4	5	6	7
Valor								

- 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**

Endereço	0	1	2	3	4	5	6	7
Valor								

- 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**

Endereço	0	1	2	3	4	5	6	7
Valor				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**

Endereço	0	1	2	3	4	5	6	7
Valor				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**

Endereço	0	1	2	3	4	5	6	7
Valor				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**

Endereço	0	1	2	3	4	5	6	7
Valor				255		7		

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

A Memória Principal

- Tomemos, agora, uma memória cheia

Endereço	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Valor	10	57	0	255	100	7	10	2

A Memória Principal

- Tomemos, agora, uma memória cheia

Endereço	0	1	2	3	4	5	6	7
Valor	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

Endereço	0	1	2	3	4	5	6	7
Valor	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

Endereço	0	1	2	3	4	5	6	7
Valor	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”:

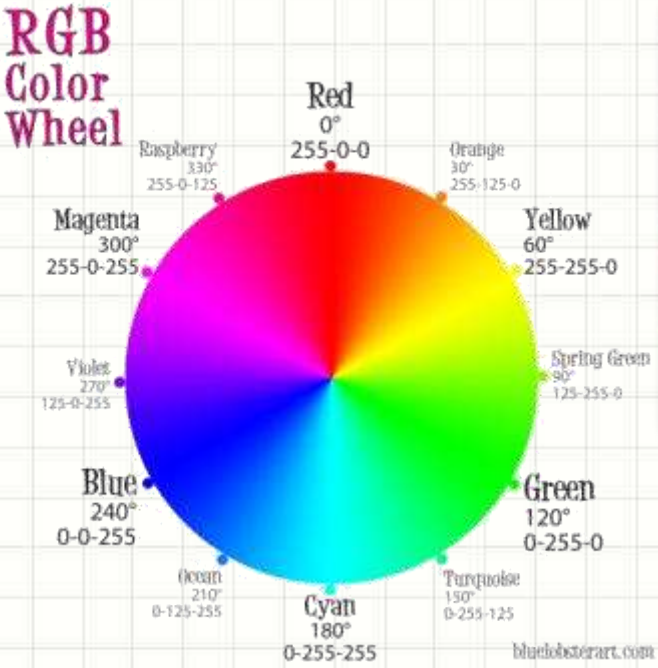
  01010010101111b

- Mas o que isso significa?

O Que o Computador Entende?

0101001010111b

- Isso pode significar várias coisas...
  - Música, imagem, quantidades...



Humanos x Processadores

- Desde muito cedo aprendemos a contar com nossos dedos;
- Como temos **DEZ** dedos nas mãos, usamos naturalmente os números DECIMAIS
- Isso significa que cada “casa” do nosso número será preenchida com um de 10 símbolos diferentes:

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

Humanos x Processadores

- Como indicar números decimais para um processador?

Humanos x Processadores

- Como indicar números decimais para um processador?

FIOS

Humanos x Processadores

- Como indicar números decimais para um processador?

0

Humanos x Processadores

- Como indicar números decimais para um processador?

1

Humanos x Processadores

- Como indicar números decimais para um processador?

2

Humanos x Processadores

- Como indicar números decimais para um processador?

3

Humanos x Processadores

- Como indicar números decimais para um processador?

4

Humanos x Processadores

- Como indicar números decimais para um processador?

5

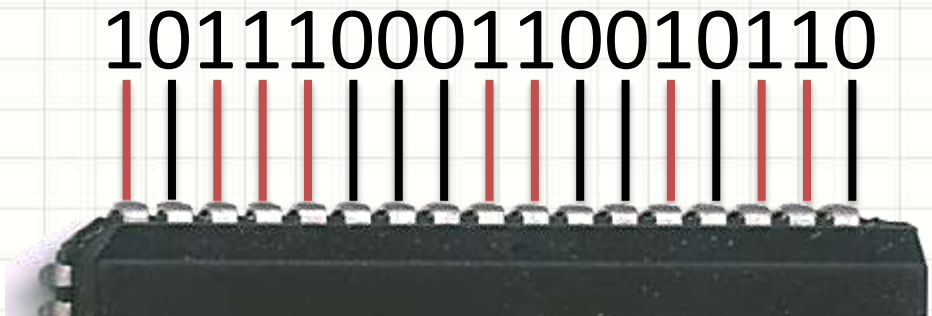
Humanos x Processadores

- Como indicar números decimais para um 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:

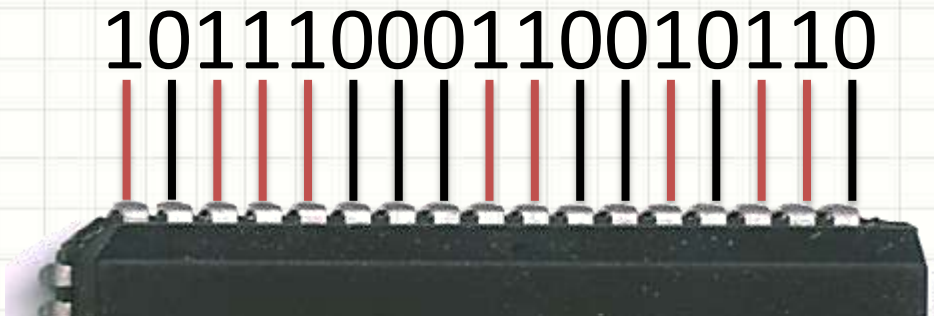


94.614

Como saber isso?

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

1011100011001010110b = 94.614



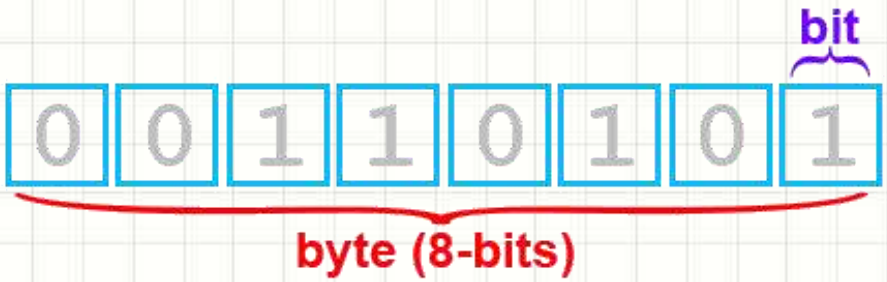
94.614

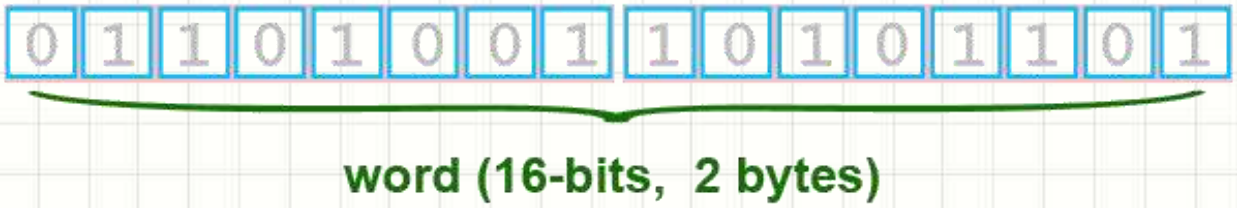
Os Números Binários

- 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

Dígito	5	4	3	2	1	0
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

Dígito	5		3		1	0
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→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→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?

- Toda quantidade inteira decimal pode ser representada exatamente em binário
- Nem toda quantidade **fracionária** decimal pode ser representada exatamente em binário
- 0,1 decimal, ao ser convertido para binário se torna:

  0,0001100110011001100110011001100110011...
- Mas o computador guarda infinitas casas?
- **NÃO!**

Qual o Problema?

- Se ele guardar apenas 16 bits, por exemplo...

  0,0001100110011001100110011001100110011...
- Vai virar isso:

  0,0001100110011001
- Que ao ser convertido para decimal é...

  0,099976
- 0,1 ≠ 0,09976
- Usar números com vírgula pode não ser uma boa ideia em programas de engenharia!

Como Guardar Outros Dados na Memória?

Outros Dados na Memória

- Só números binários... 0101001010111b
- Seu significado depende da interpretação!
  - Já vimos números inteiros “sem sinal”
- E outras coisas?
  - Números inteiros com sinal
  - Letras...
  - Imagens...
- Interpretação depende de uma **convenção**

Representação com Sinal

- Números com sinal

Bit	7 (Sinal)	6	5	4	3	2	1	0
Valor	1	0	0	0	0	1	0	0

  - 1: Negativo
  - 0: Positivo
- 100b = 4
  - Então este número é o -4

Representação de Caracteres

- Letras: Padrões de codificação
  - ASCII
  - UTF-8
  - UTF-16

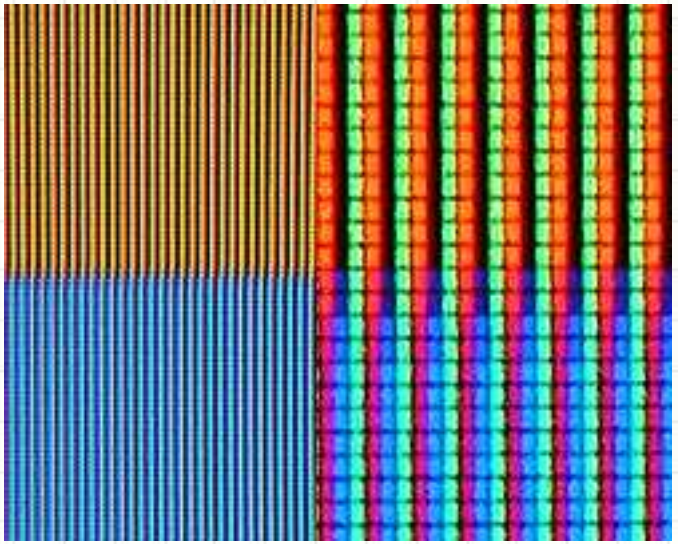
Código	Caracteres	Código	Caracteres	Código	Caracteres
32	[space]	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	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)

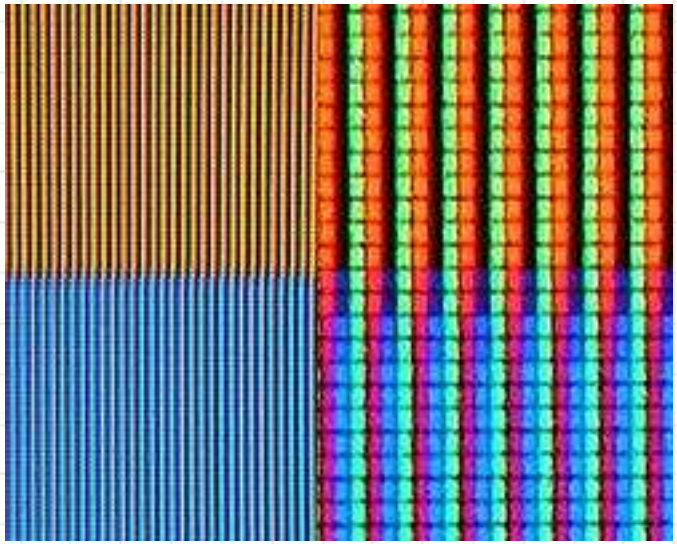
Representação de Imagens

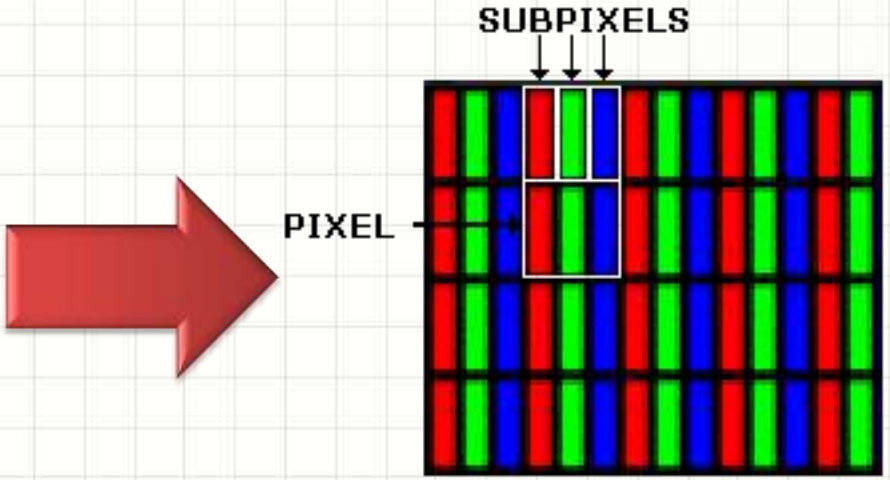
- Pontos: RGB (Vermelho, Verde, Azul)



Representação de Imagens

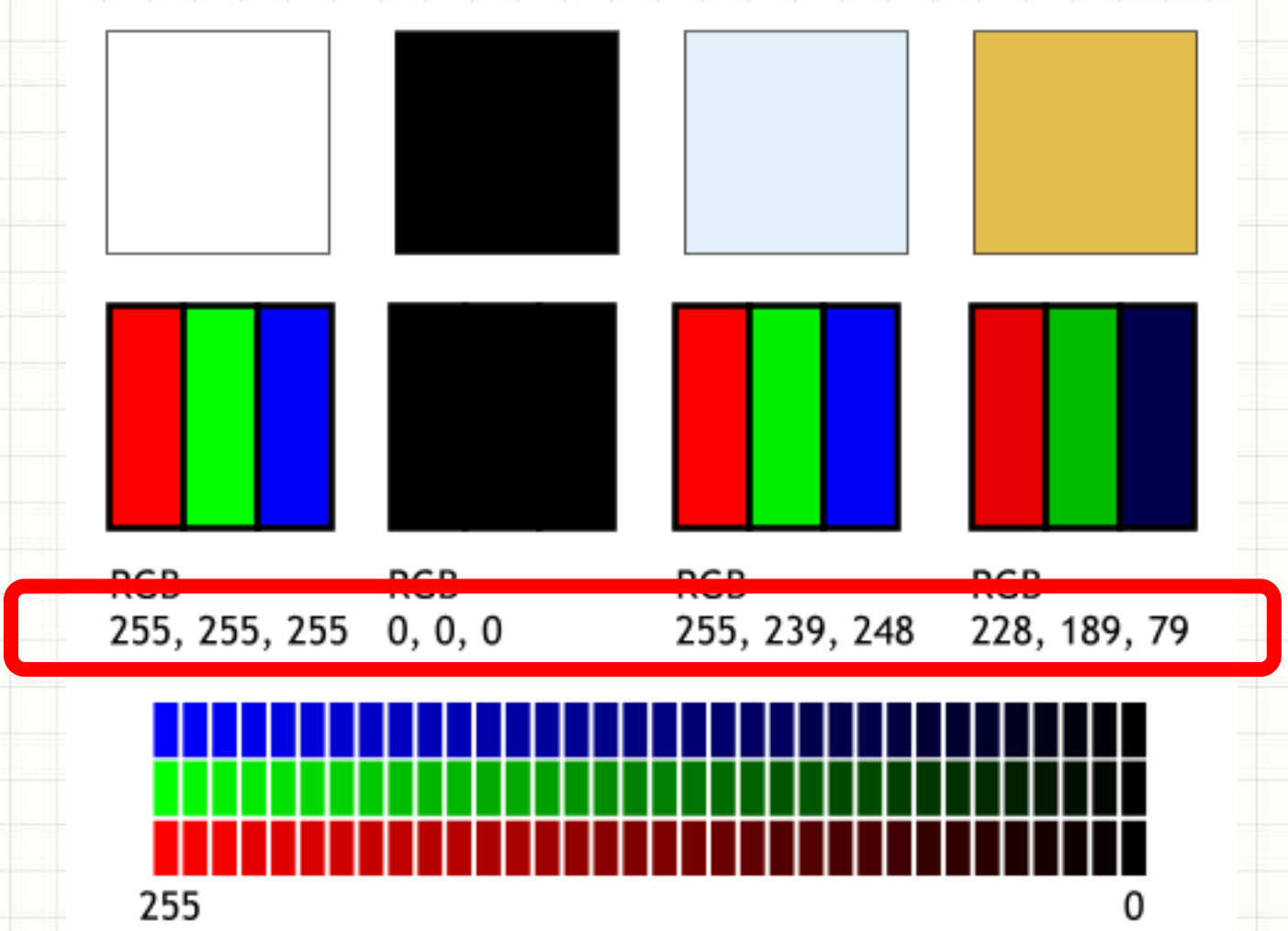
- Pontos: RGB (Vermelho, Verde, Azul)





Representação de Imagens

- Pontos: RGB (Vermelho, Verde, Azul)



Quantos Dados Cabem na Memória?

Quantas “gavetas” há?

- Gaveta = Posição de Memória
- Cada gaveta tem um **endereço numérico**
- O endereço é definido por um conjunto de fios
- Quantos bits tem o endereço?!
- Regra prática!
- Número de posições de memória = 2^n
  - **n** é o número de bits do endereço

- Exemplo:
  - Endereço de 16 bits → 2^{16} = 65.536 (64 KB)
  - Endereço de 32 bits → 2^{32} = 4.294.967.296 (4 GB)

Quanto cabe em uma “gaveta”?

- Em geral, cada “gaveta” tem **um byte** (8 bits)
- Números grandes: usam várias “gavetas”
  - **byte** → 8 bits → 1 posição de memória
  - **word** → 16 bits → 2 posições de memória
  - **dword** → 32 bits → 4 posições de memória
- Alguns computadores possuem “gavetas” maiores...

Quanto cabe em uma “gaveta”?

- Regra prática!
  - Maior valor = $2^m - 1$
  - **m** é o número de bits da posição de memória

- Exemplo:
  - 8 bits → $2^8 - 1 = 256 - 1 = 255$
  - 16 bits → $2^{16} - 1 = 65.536 - 1 = 65.535$

EXERCÍCIOS

Exercícios

1. Para que serve a CPU e a Memória?
2. Onde normalmente o computador busca instruções?
3. O que é preciso saber para compreender o significado de um número binário?
4. Qual o problema em usar números fracionários no computador?
5. Converta:
   a) 11101b para decimal
   b) 72 para binário

Entrega dos Grupos de Trabalho

Conclusões

Resumo

- Elementos e lógica básica do computador
- Organização da memória
- Números binários e conversão B→D e D→B
- Codificações mais complexas

- **TAREFA PARA PRÓXIMA AULA**
  - **Estudar!**

Próxima Aula

- O que é lógica?
  - Como transformar problemas reais em uma sequência lógica de solução?

PERGUNTAS?

Bom Descanso a Todos!