Lógica de Programação para Engenharia

Introdução à Organização de Computadores

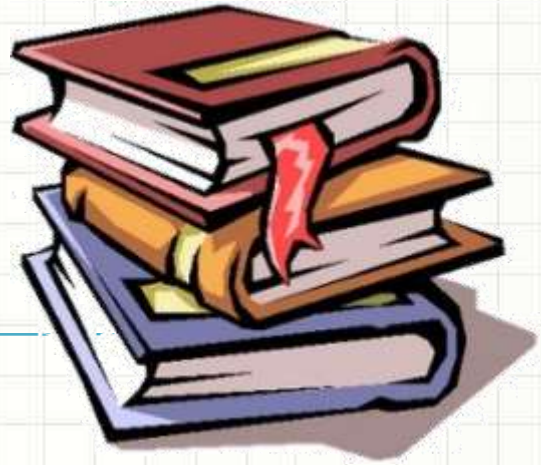
Prof. Dr. Daniel Caetano

2011 - 2

Visão Geral

1. O Computador
2. A Memória Principal
3. A Linguagem do Computador
4. Os Dados e a Memória

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.

Objetivos

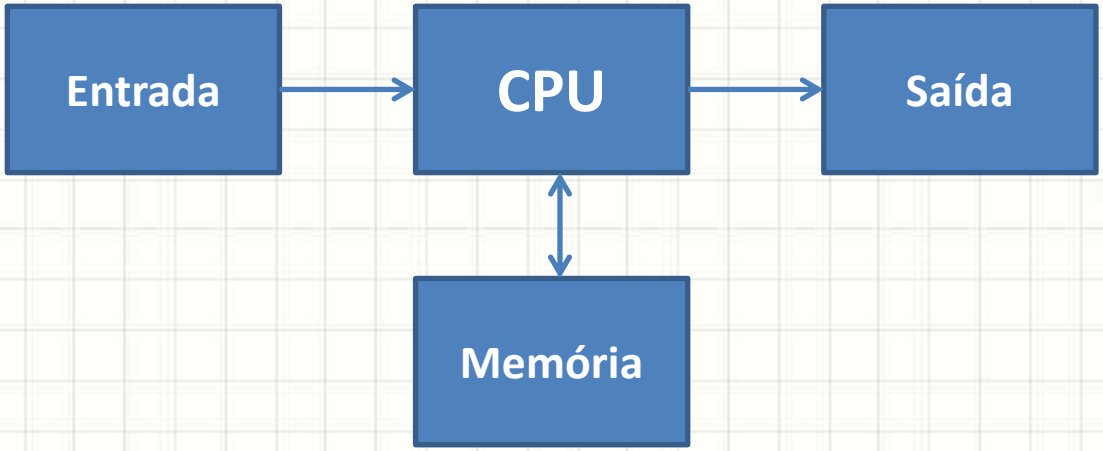
- Apresentar o funcionamento do computador
- Apresentar a função da memória e dos dispositivos de entrada e saída
- Compreender o armazenamento de dados na memória

- **GRUPOS**?
  - Até o fim da aula!

O Computador

Entendendo o Computador

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



Entendendo o Computador

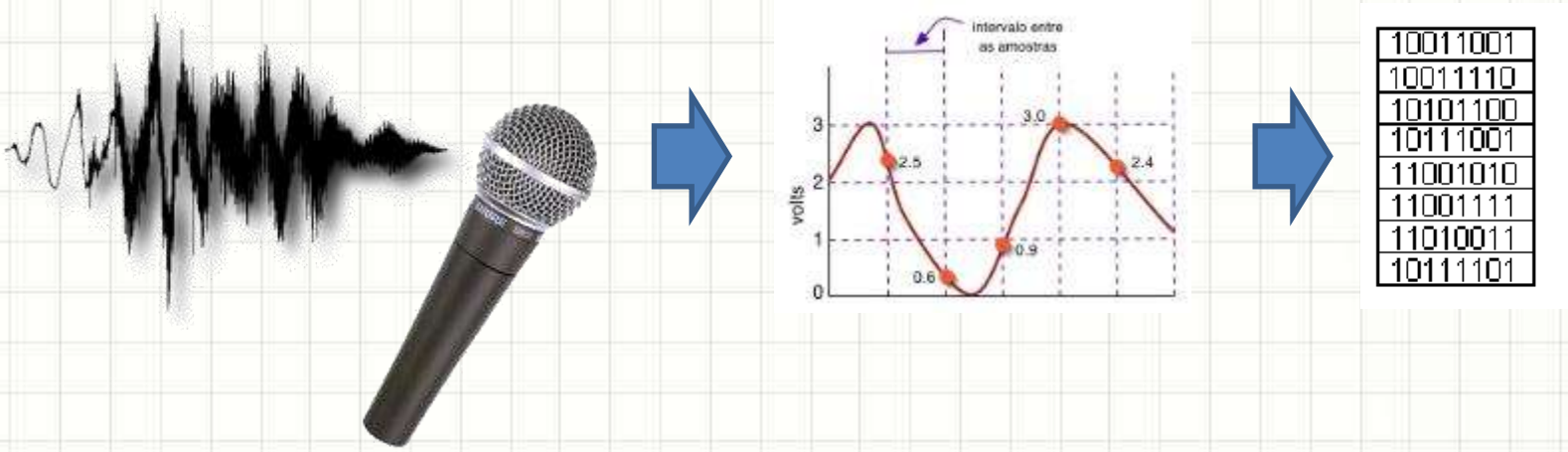
- **CPU**: Coordena todo o funcionamento do computador e realiza cálculos **numéricos**
- **Unidade de Entrada**: Recebe dados (números) externos para processamento
- **Unidade de Memória**: Armazena dados (números) para uso posterior
- **Unidade de Saída**: Exibe dados (números) para o usuário, após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

Entendendo o Computador

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

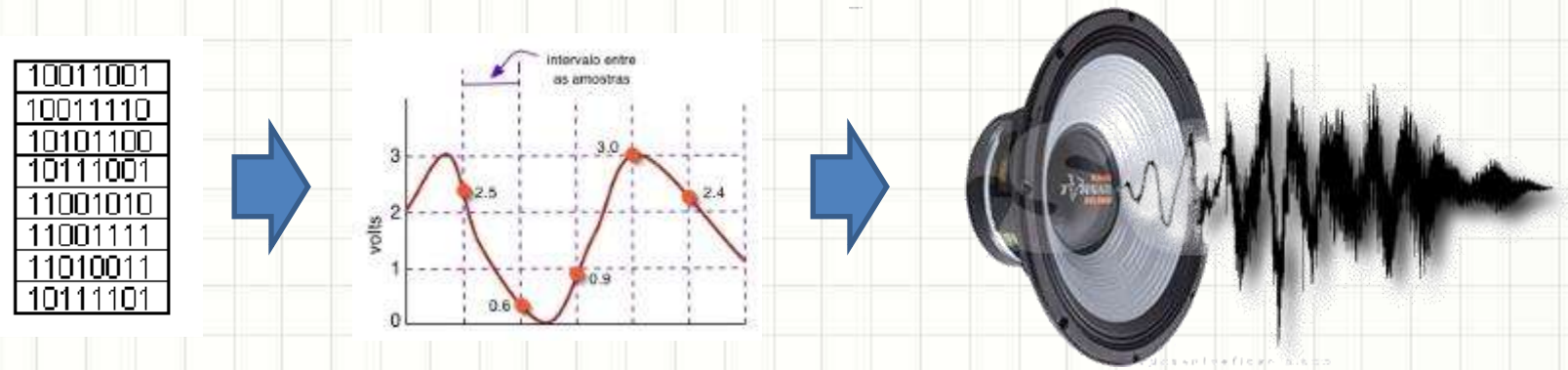
Dispositivos de Entrada e Saída

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



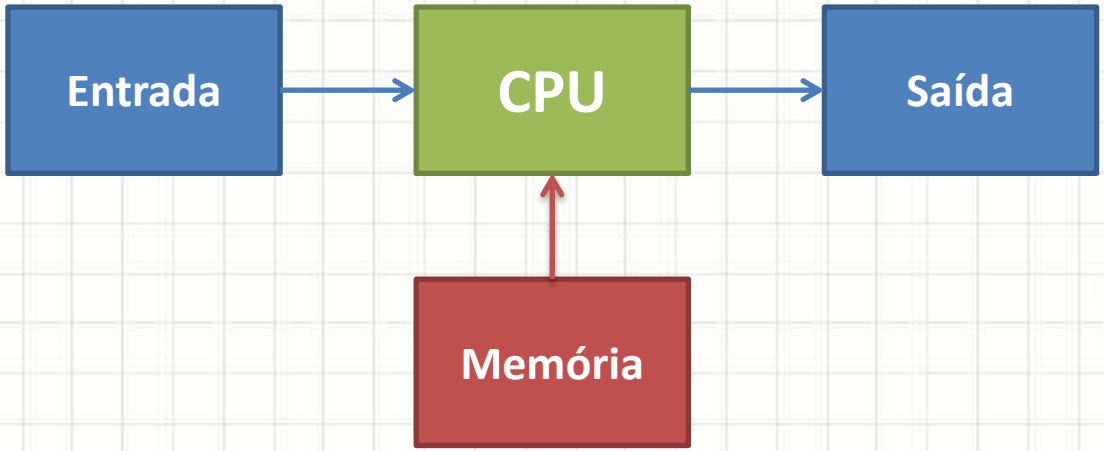
Dispositivos de Entrada e Saída

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



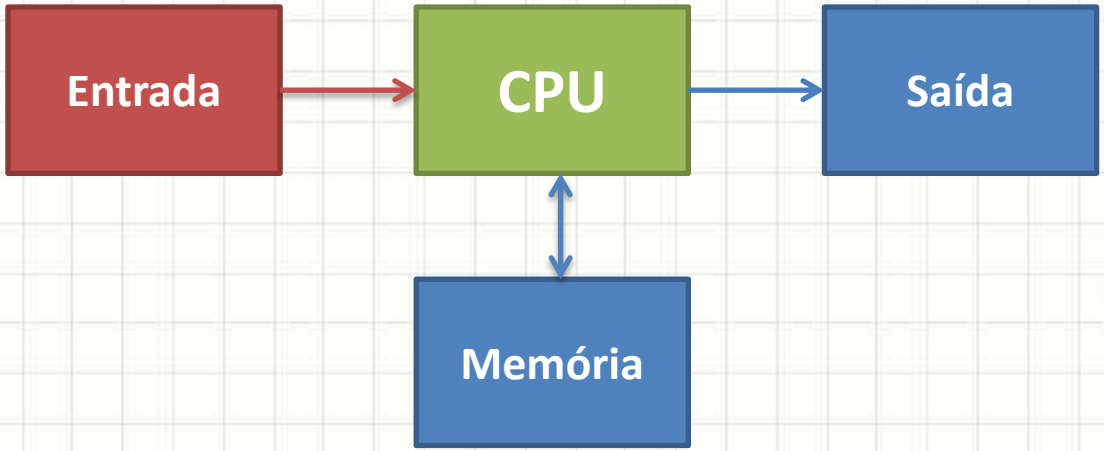
Funcionamento da CPU

- **Busca Instrução**: CPU lê a memória em busca do que deve fazer



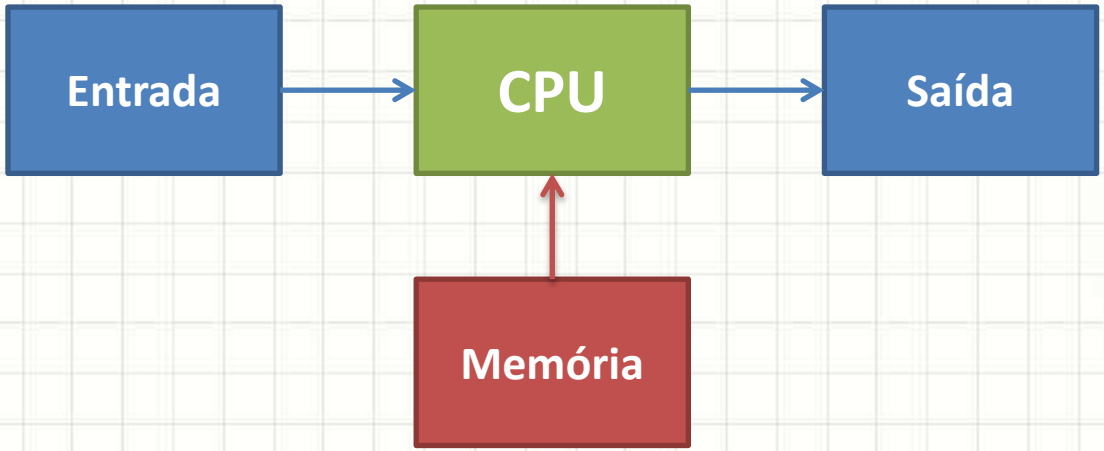
Funcionamento da CPU

- **Lê entrada**: Supondo que a instrução indica a leitura de um dado de entrada, CPU lê entrada



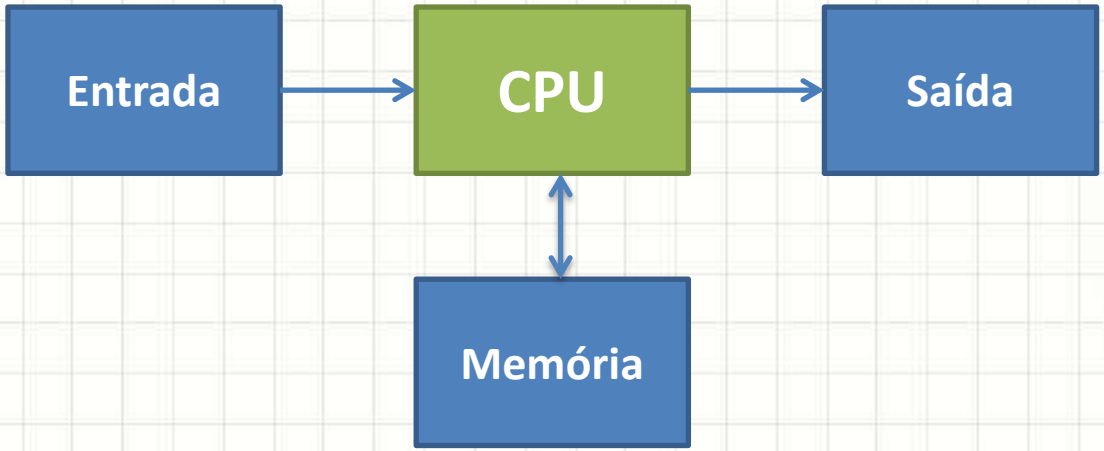
Funcionamento da CPU

- **Busca Instrução**: CPU lê a memória em busca do que deve fazer com dado lido



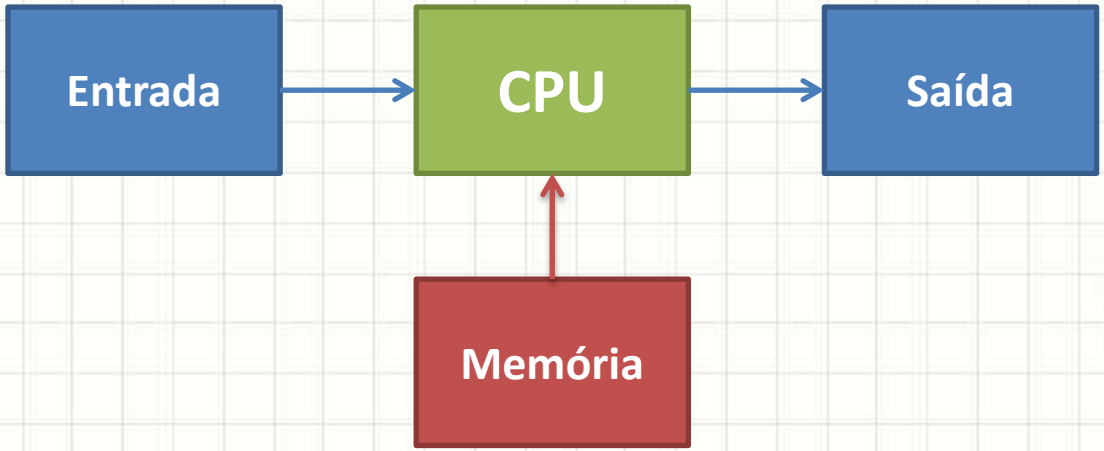
Funcionamento da CPU

- **Cálculos**: Supondo uma instrução de cálculo, a CPU realiza a operação



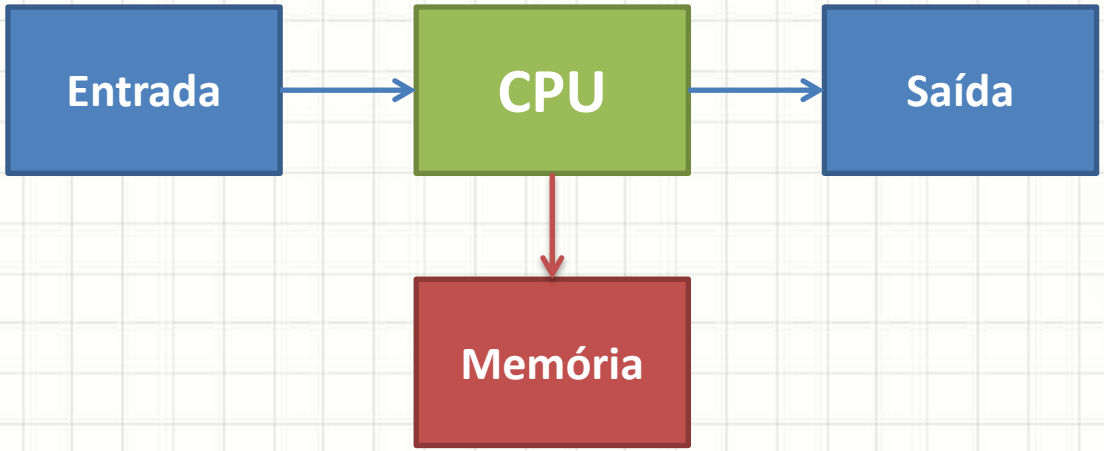
Funcionamento da CPU

- **Busca Instrução**: CPU lê a memória em busca do que deve fazer



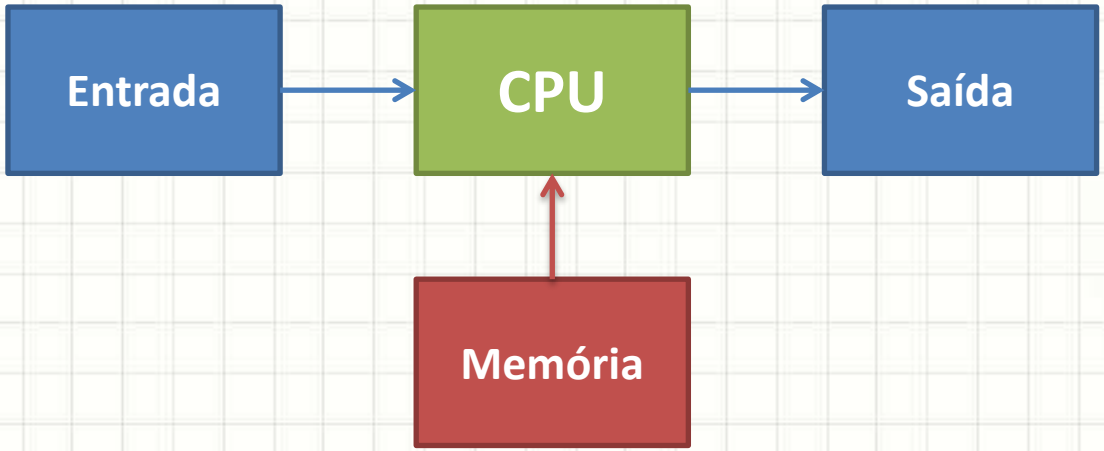
Funcionamento da CPU

- **Armazena dado**: Supondo que a instrução solicitava armazenamento do dado na memória, guarda o mesmo na memória



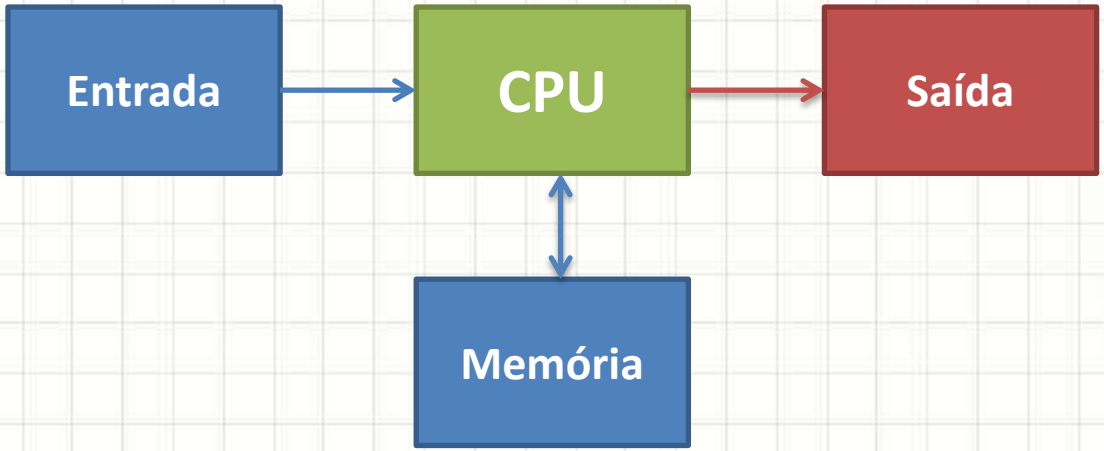
Funcionamento da CPU

- **Busca Instrução**: CPU lê a memória em busca do que deve fazer



Funcionamento da CPU

- **Apresenta saída**: Supondo que a instrução indica que um dado deve ser apresentado ao usuário, CPU escreve na saída



Funcionamento da CPU

- **Apresenta saída**: Supondo que a instrução indica que um dado deve ser apresentado

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
  - Armazena resultado em **registradores**
- A **ULA** é quem faz cálculos
  - Acessa apenas valores nos **registradores**

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, cabe um único número

A Memória Principal

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**!

A Linguagem do Computador

O Que o Computador Entende?

- O computador entende apenas números...
- ...em um dialeto chamado “binário”

  0101001010111b
- O que significa isso?
- Pode significar várias coisas, depende da interpretação!
  - Música, imagem, números...
- Comecemos com os números!
  - Mas para entender os números do computador, precisamos primeiro entender os nossos!

Os Números Decimais

- Temos **DEZ** dedos, contando ambas a mãos
- Como consequência, lidamos com números DECIMAIS
- Isso significa que cada um dos dígitos de um número pode ser preenchido com um de 10 símbolos diferentes:

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

Os Números Decimais

- Quando precisamos representar um número maior que 9, acrescentamos um dígito 1 à esquerda... E o dígito da direita volta a zero:

0
1
2
3
4
5
6
7
8
9
10

Os Números Decimais

- Assim, o número à esquerda indica quantas vezes o número à direita já superou 10 símbolos de contagem.
- Por exemplo:

17 = 1* 10 + 7

Os Números Binários

- Para representar números, o computador conta apenas com sinais elétricos
- Por motivo de precisão, apenas **DOIS** estados elétricos são usados para indicar números: **ligado** e **desligado**.
- Cada "fio" do computador – pelo qual pode estar passando corrente elétrica ou não – representa um dígito para ele – chamado **bit**
- Sendo assim, os números são compostos por apenas dois "símbolos":

0, 1

Os Números Binários

- Quando precisamos representar um número maior que 1, acrescentamos um dígito 1 à esquerda... E o dígito da direita volta a zero:

0

1

10

11

100

Os Números Binários

- Assim, o número à esquerda indica quantas vezes o número à direita já superou 2 símbolos de contagem.
- Por exemplo:

11 = 1* 2 + 1

- A regra está parecida com a do decimal, não?

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!

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

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 de Decimal para Binário

- Regra prática: converter 13 para binário
- 13/2 = 6,5

Fracionário!

Conversão de Decimal para Binário

- Regra prática: converter 13 para binário

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

Exato!

Conversão de Decimal para Binário

- Regra prática: converter 13 para binário

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

Fracionário!

101b

Conversão de Decimal para Binário

- Regra prática: converter 13 para binário

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

Fracionário!

1101b

Conversão de Decimal para Binário

- Regra prática: converter 13 para binário

- 13/2 = 6,5
- 6/2 = 3,0
- 3/2 = 1,5
- 1/2 = 0,5
- 0 **Fim!**

1101b

Conversão de Decimal para Binário

- Regra prática: converter 13 para binário

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

13 = 1101b

1101b

Os Dados e a Memória

O que cabe em uma “gaveta”?

- Depende do computador/memória
- Memória de 8 bits tem 8 bits por “gaveta”
  - Cada posição tem até 8 “fios” para guardar os dados
- Memória de 16 bits tem 16 bits por “gaveta”
  - Cada posição tem até 16 “fios” para guardar os dados

- Qual o maior número inteiro sem sinal que cabe em uma dessas gavetas?

O que cabe em uma “gaveta”?

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

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

Quantas “gavetas” há?

- Depende do número de bits do endereço!
- Regra prática!
  - Número de posições = 2^m
  - **m** é o número de bits do endereço

- Exemplo:
  - 8 bits → $2^8 = 256$
  - 16 bits → $2^{16} = 65.536$ (64 KB)
  - 32 bits → $2^{32} = 4.294.967.296$ (4 GB)

O Que Está na Memória?

- Números binários...
- O que eles significam?

  01010010101111b

- Depende da interpretação!
  - Números inteiros, sem sinal, como já vimos...
  - Números inteiros com sinal
  - Números reais
  - Letras...

O Que Está na Memória?

- 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

O Que Está na Memória?

- Números Reais (simplificado)
- Mantissa * 2^{Exponte}
  - 0100b * 2^{-11b} =
  - 4 * 2^{-3} =
  - 4 / 8 =
  - 0,5

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

O Que Está na Memória?

- Números Maiores...?

- Divididos em várias “gavetas”
  - **byte** → 8 bits → 1 posição de memória
  - **word** → 16 bits → 2 posições de memória
  - **dword** → 32 bits → 4 posições de memória

O Que Está na Memória?

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

Binário	Decimal	Hexa	Glifo
0010 0000	32	20	
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	+
0010 1100	44	2C	,
0010 1101	45	2D	-

Binário	Decimal	Hexa	Glifo
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
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

Binário	Decimal	Hexa	Glifo
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
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

O Que Está na Memória?

- Letras?
- Padrões de codificação

Tabela completa nas notas de aula!

											Glifo
											`
											a
											b
											c
											d
											e
											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

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!