



INFORMÁTICA PARA ENGENHARIA

INTRODUÇÃO À ORGANIZAÇÃO DE COMPUTADORES

Prof. Dr. Daniel Caetano

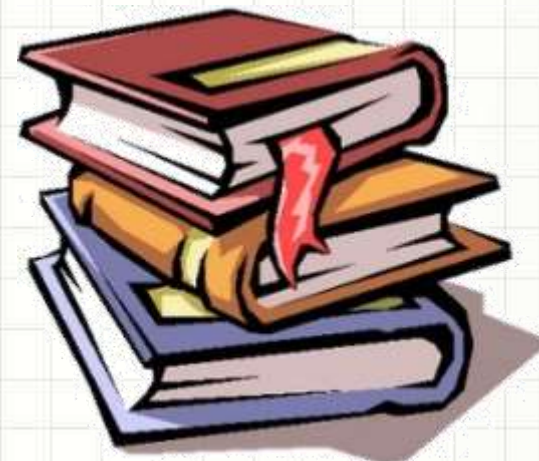
2019 - 1

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/>
(Informática para Engenharia – Aula 2)

Material Didático

Lógica de Programação, págs 19-25

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.

LEMBRETE: CONSULTAR O “ANTES” DA AULA 3 NO SAVA!

Antes de Mais nada...

- **Não deixe de consultar o material da 1ª Aula!**
- **Disciplina Híbrida: Presencial + EAD**
 - Toda semana acessar o SAVVA!
 - Se preparar para conteúdo da semana seguinte!
- **Exercícios Semanais**
 - Exercícios propostos a cada aula: SAVVA
- **Será controlada a presença**
 - Chamada ocorrerá sempre às 20:30 / 22:25
 - Nome fora da lista = falta

- **Contato**

Professor

Informações de Contato

Daniel Caetano

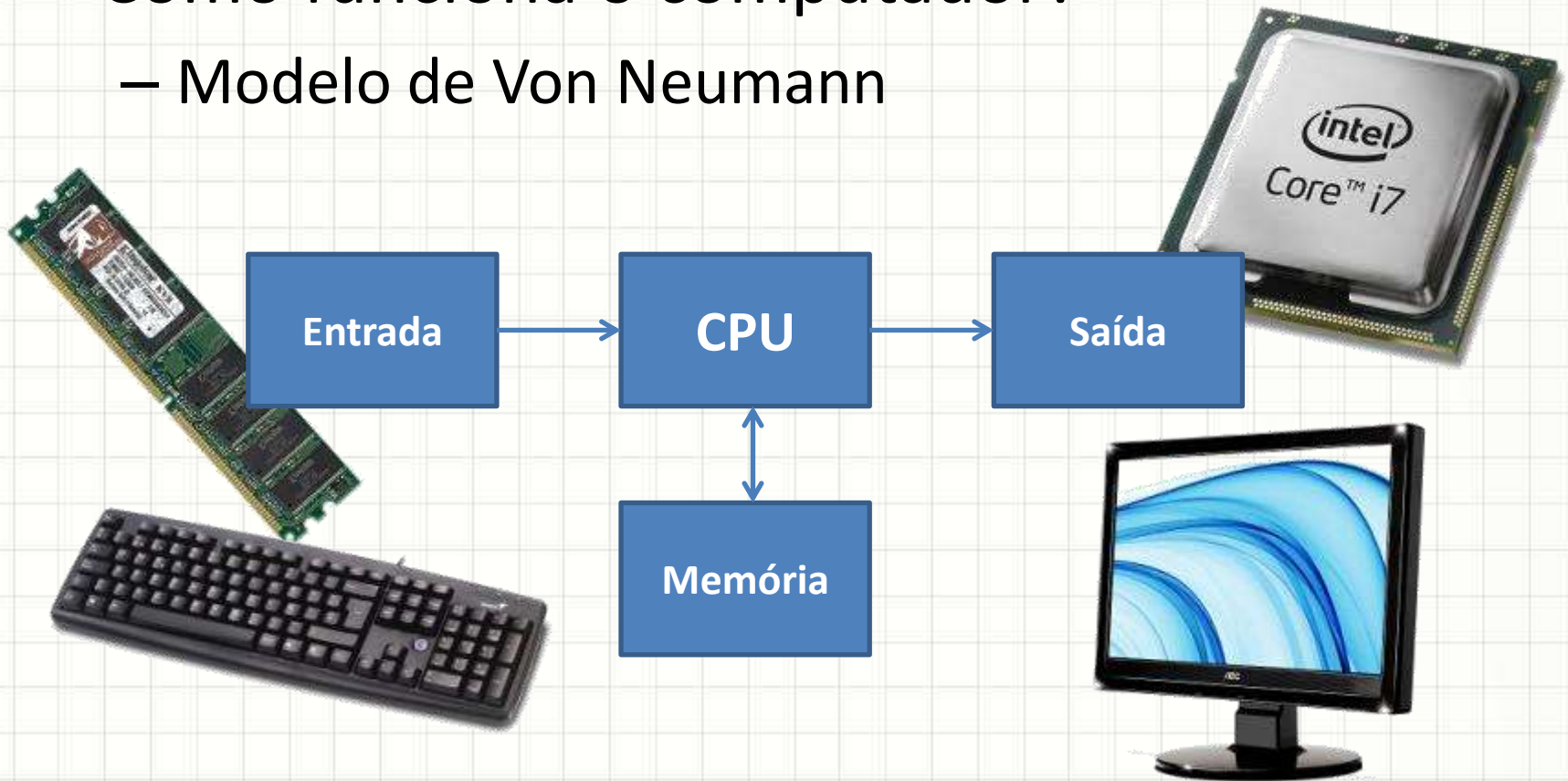
prof@caetano.eng.br



O COMPUTADOR

Entendendo o Computador

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

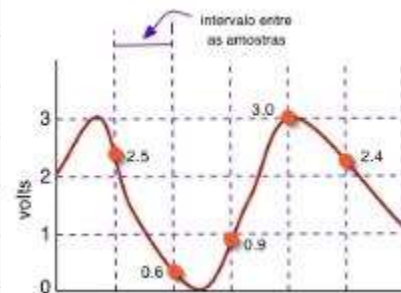


Entendendo o Computador

- **CPU**: Coordena todo o funcionamento do computador e realiza cálculos numéricos
- **Unidade de Entrada**: Recebe estímulos externos e os transforma em dados (números)
- **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

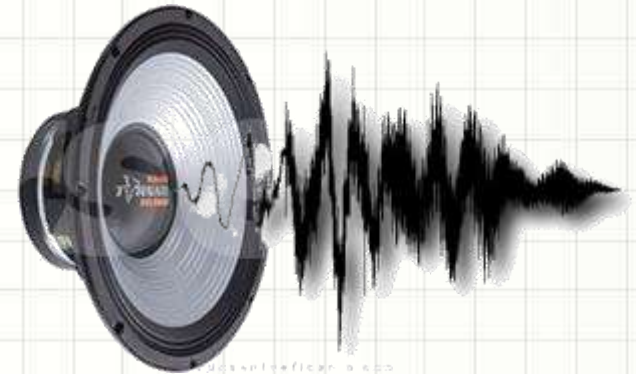
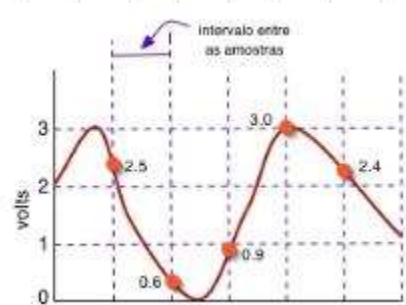


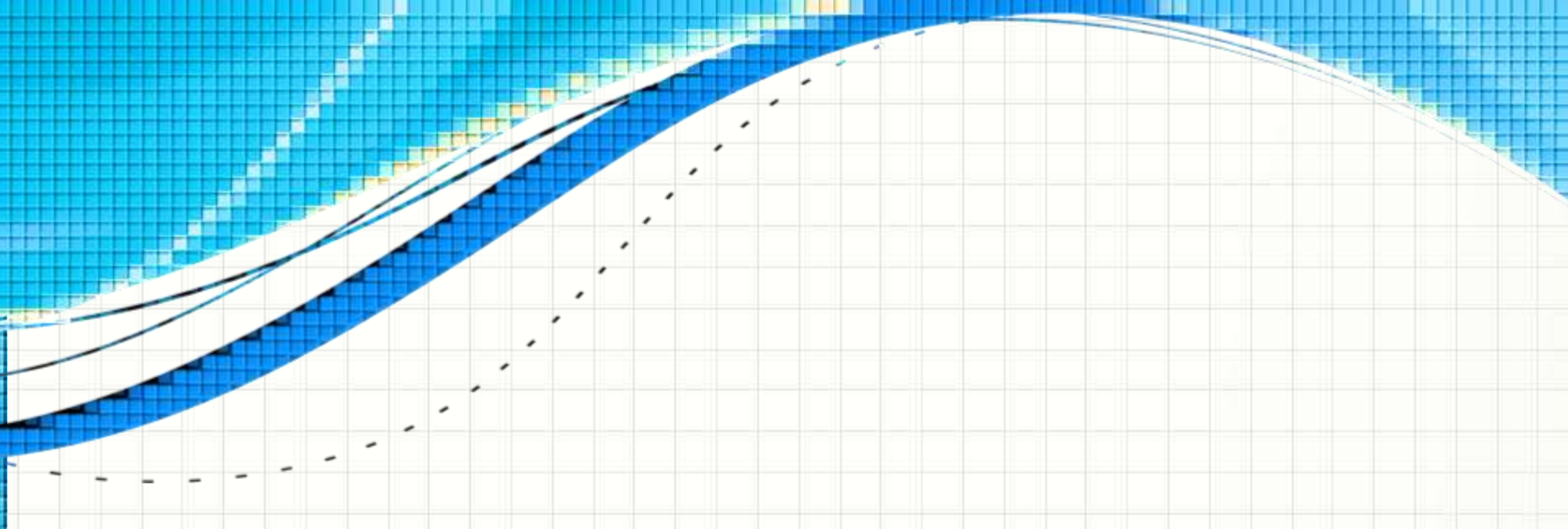
10011001
10011110
10101100
10111001
11001010
11001111
11010011
10111101

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

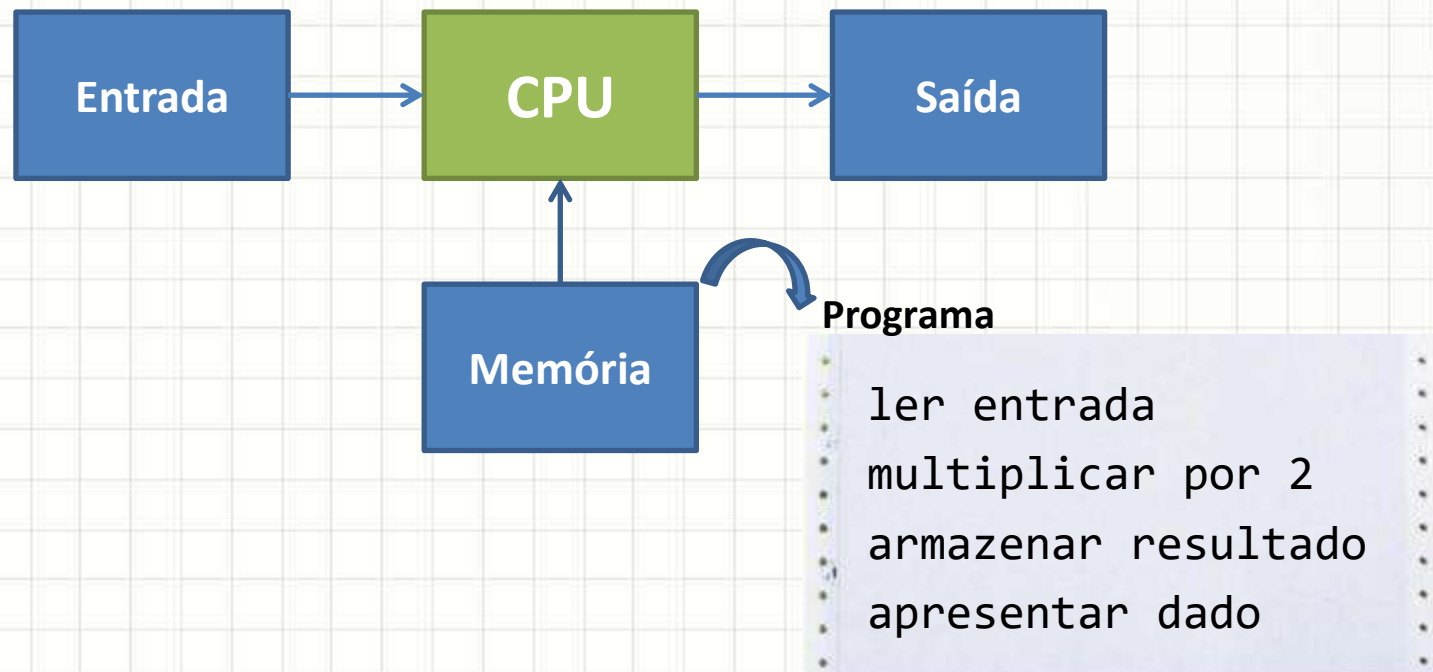




FUNCIONAMENTO DA CPU

Funcionamento da CPU

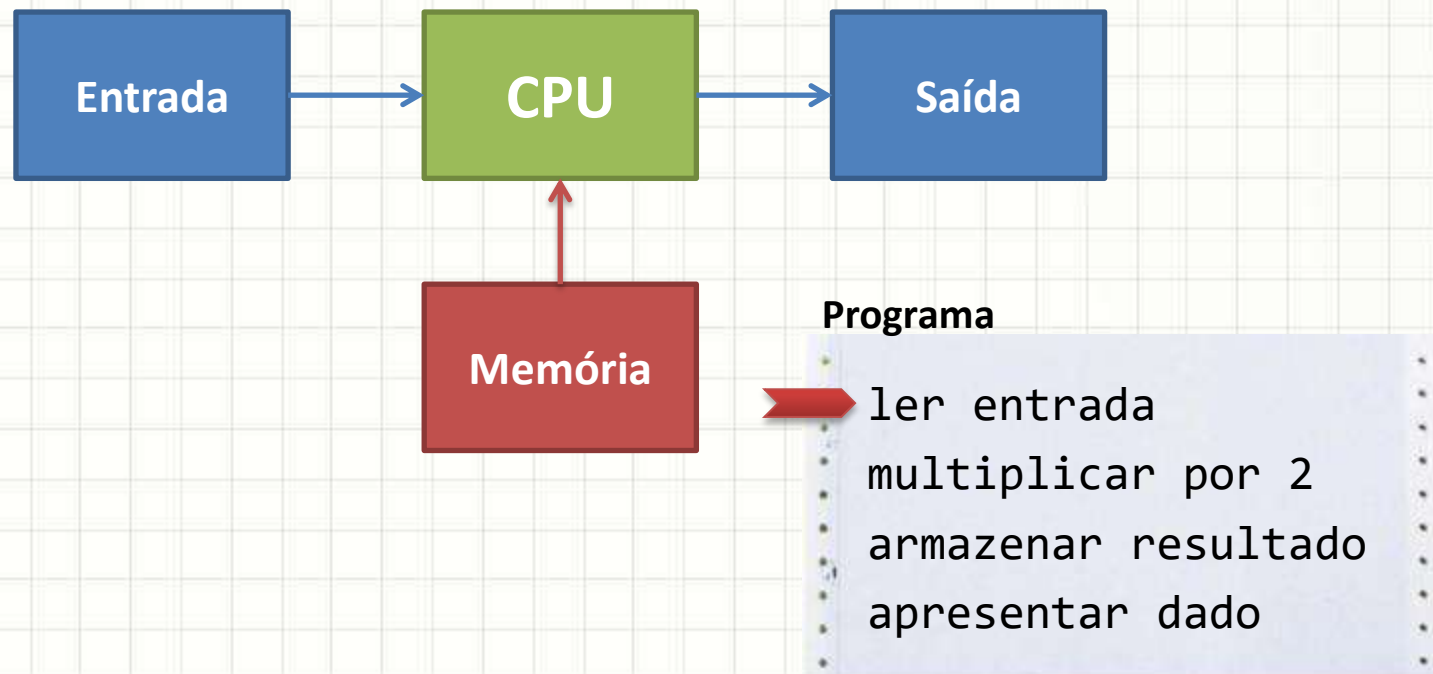
- Como a CPU coordena essas partes para produzir resultado útil?



Funcionamento da CPU

- **1. Busca Instrução**

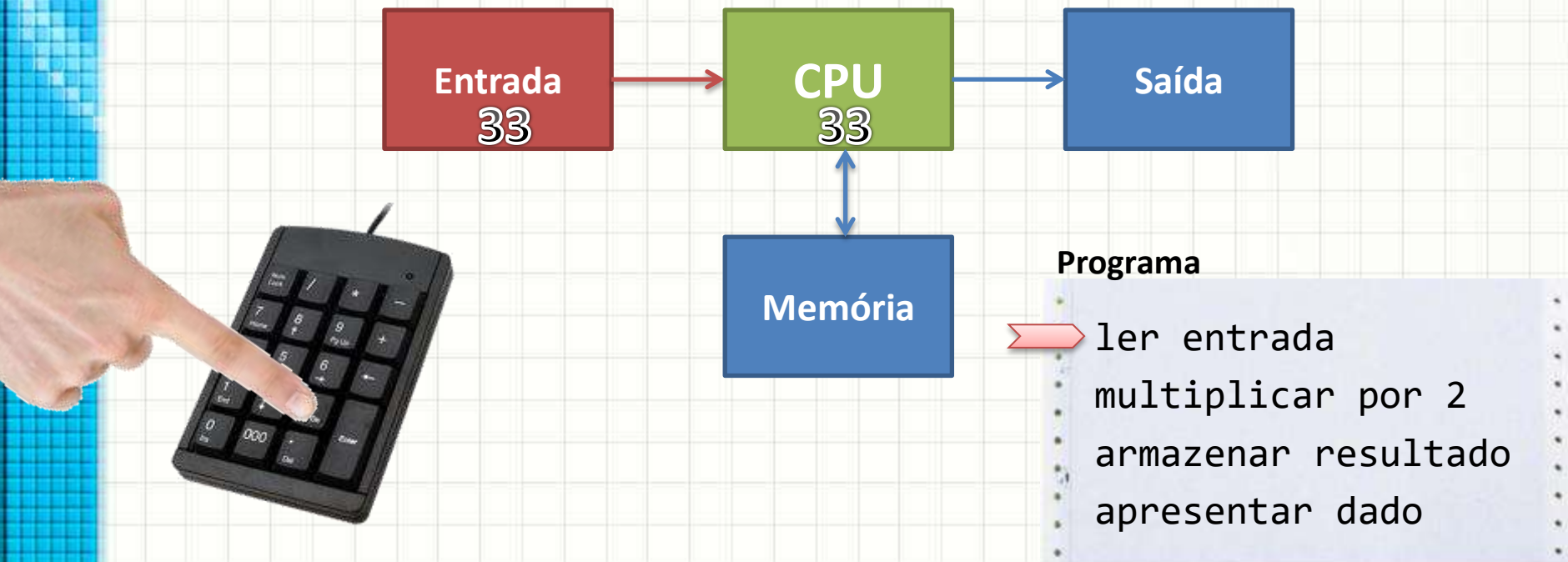
- CPU lê a memória em busca do que deve fazer
- Instrução lida: **ler entrada**



Funcionamento da CPU

- **2. Lê entrada**

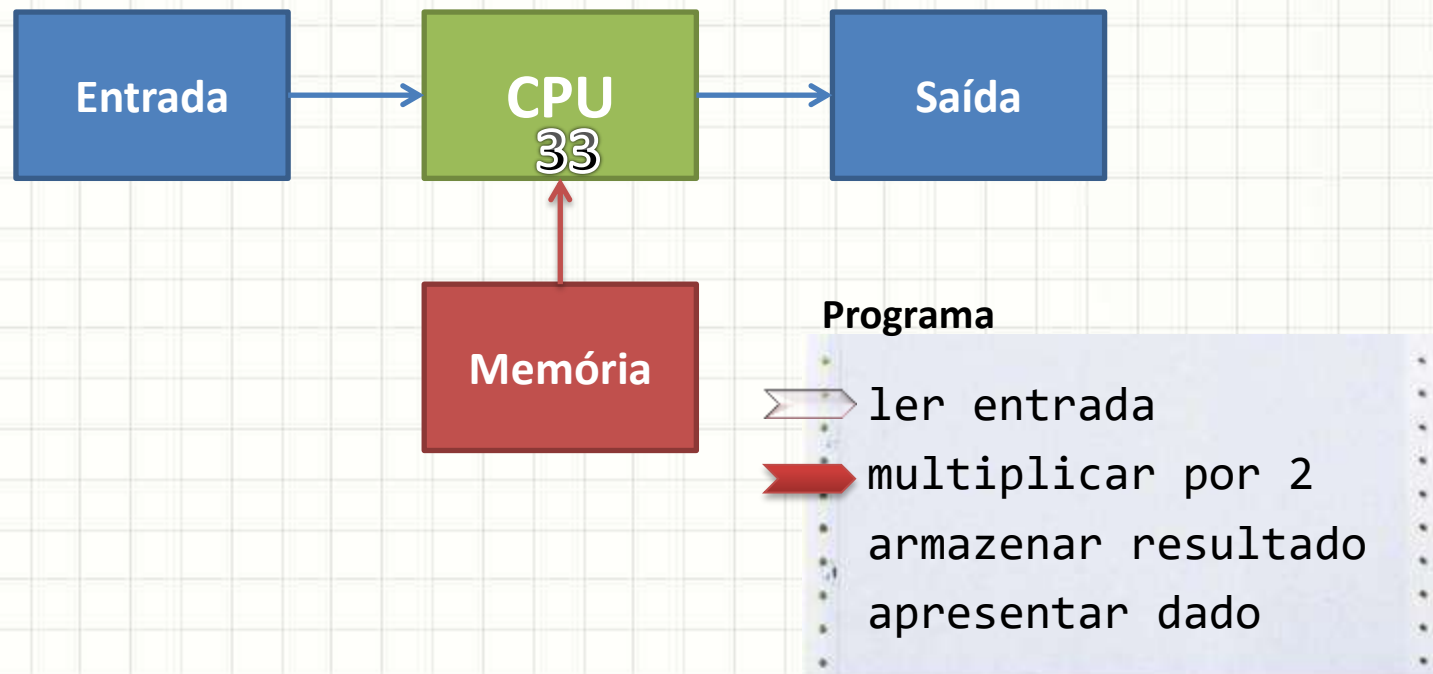
- CPU lê entrada, recebendo um dado numérico
- Dado lido: 33



Funcionamento da CPU

- **3. Busca Instrução**

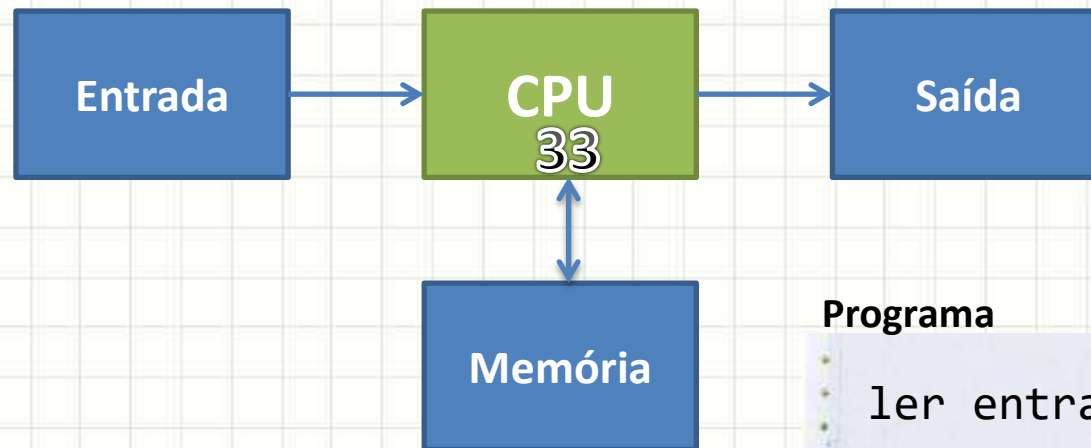
- CPU lê a memória em busca do que deve fazer
- Instrução lida: **multiplicar por 2**



Funcionamento da CPU

- **4. Processamento**

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



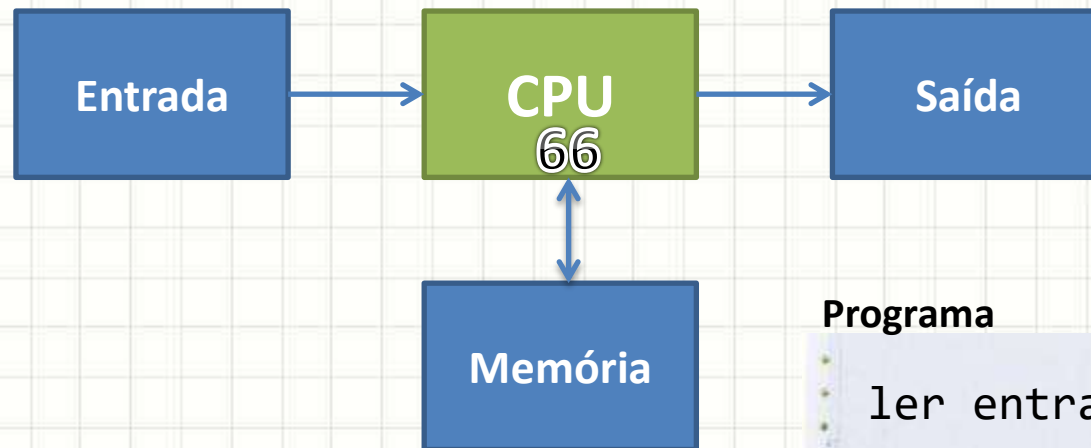
Programa

```
ler entrada  
multiplicar por 2  
armazenar resultado  
apresentar dado
```

Funcionamento da CPU

- **4. Processamento**

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



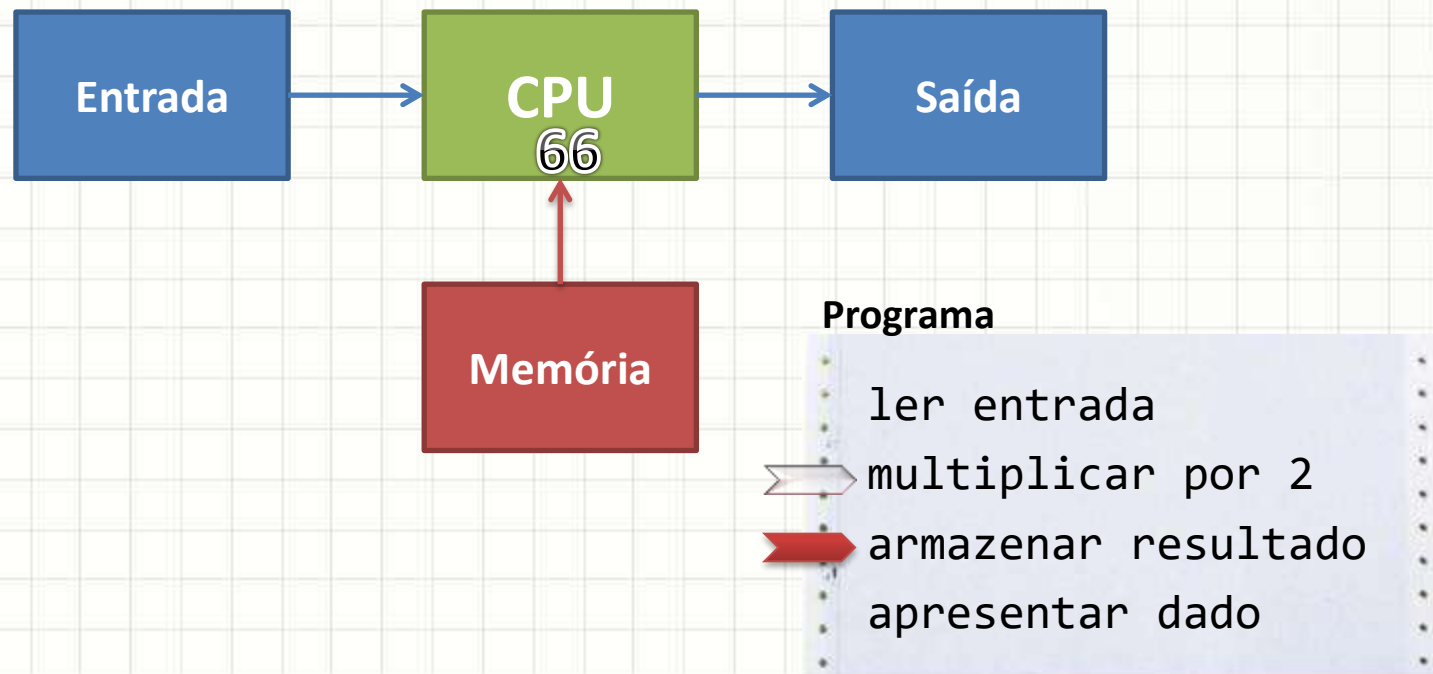
Programa

```
ler entrada  
multiplicar por 2  
armazenar resultado  
apresentar dado
```

Funcionamento da CPU

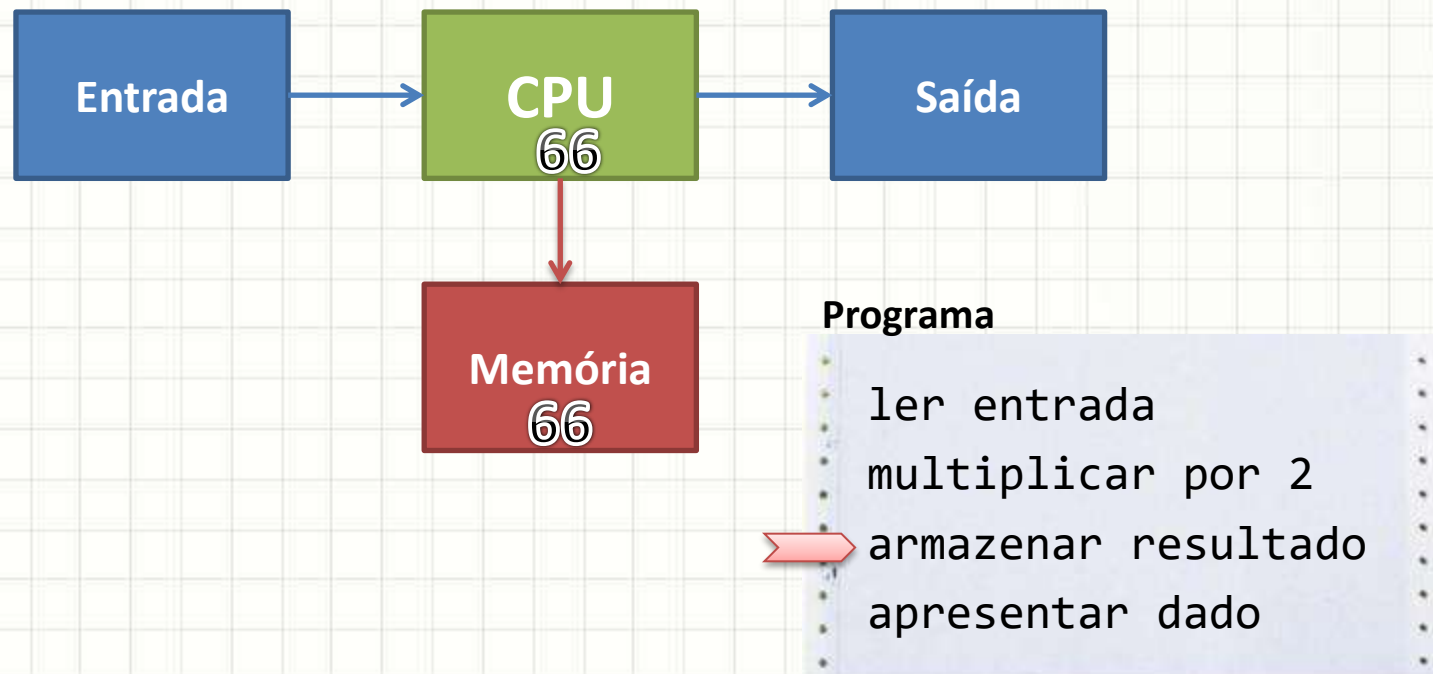
- **5. Busca Instrução**

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



Funcionamento da CPU

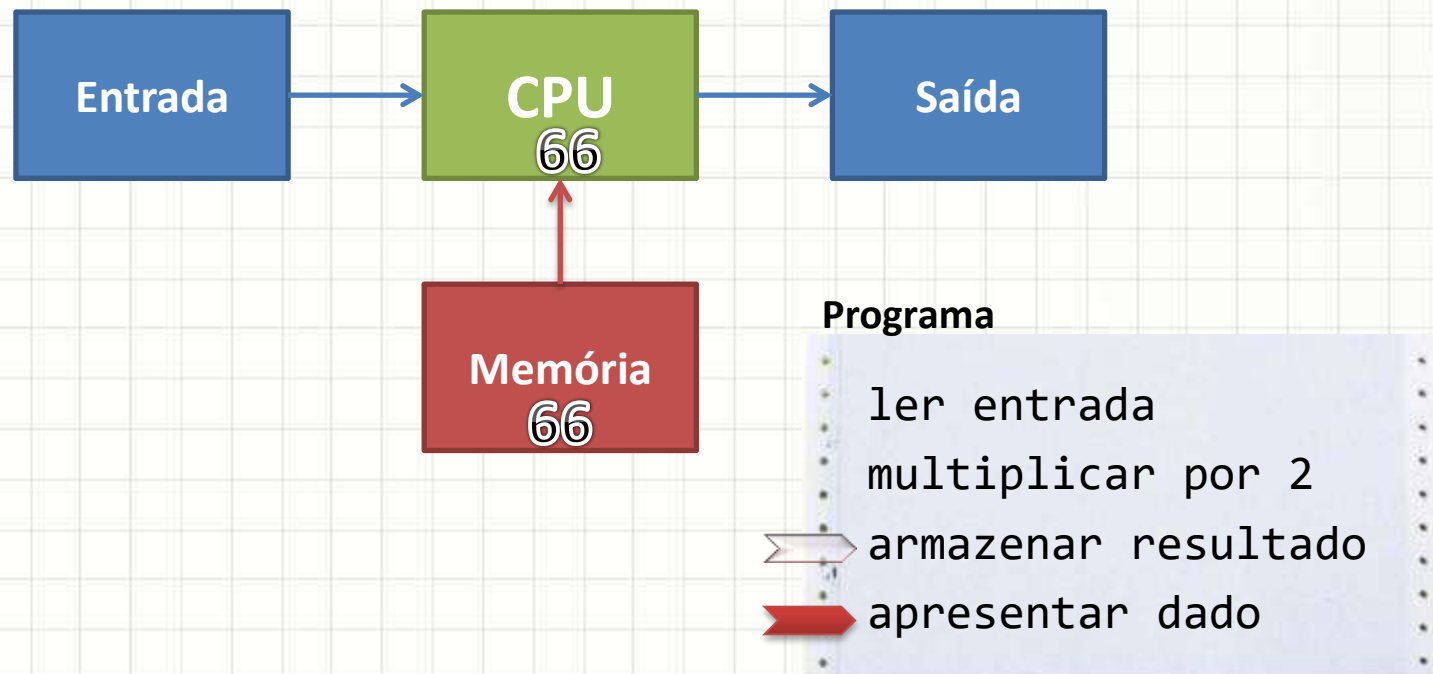
- **6. Armazenar dado**
 - Guarda o dado na memória
 - Dado armazenado: 66



Funcionamento da CPU

- **7. Busca Instrução**

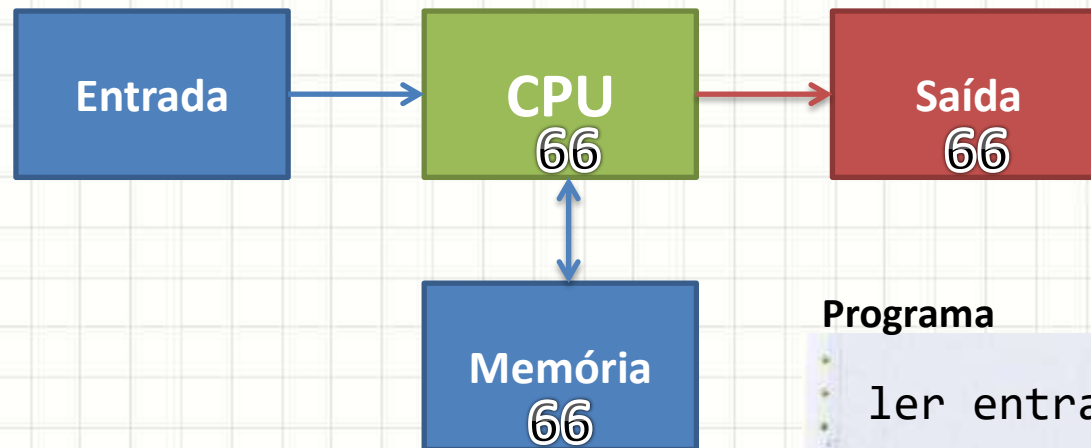
- CPU lê a memória em busca do que deve fazer
- Instrução lida: **apresentar dado**



Funcionamento da CPU

- **8. Saída de Dados**

- CPU escreve na saída
- O número 66 aparece no monitor



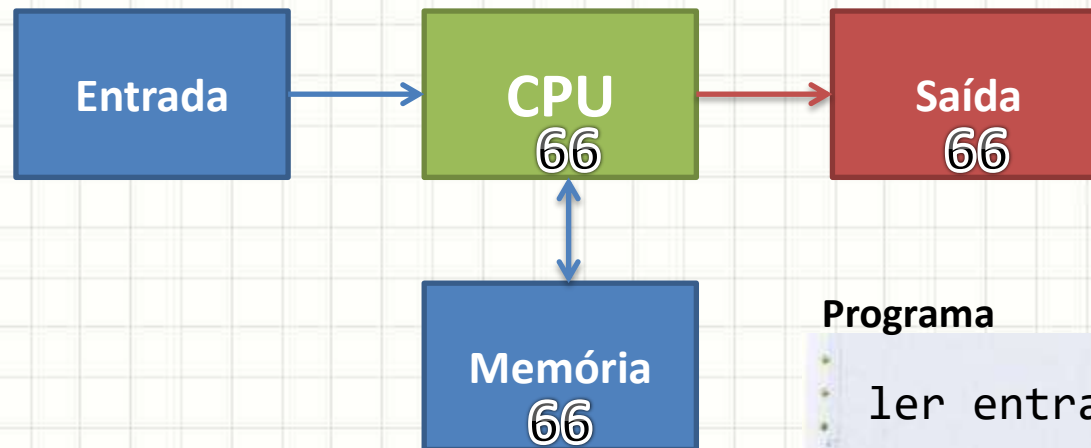
Programa

```
ler entrada
multiplicar por 2
armazenar resultado
apresentar dado
```

Funcionamento da CPU

- 8. Saída de Dados

E assim sucessivamente...



Programa

```
ler entrada
multiplicar por 2
armazenar resultado
apresentar dado
```


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





A MEMÓRIA E SEU FUNCIONAMENTO

As Memórias do Computador

- Armazenamento Temporário
 - Memória Principal / Memória Cache
 - RAM: Random Access Memory (escrita e leitura)
- Armazenamento “Permanente”
 - ROM: Read Only Memory (Cartuchos, só escrita)
 - Discos Magnéticos (Disquetes, HDDs)
 - Disco Ópticos (CDs, DVDs...)
 - Flash RAM (SDs, SSDs etc)
 - Legados (fitas magnéticas, fitas perfuradas...)

A Memória Principal

- A memória é como um arquivo de fichas
- 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

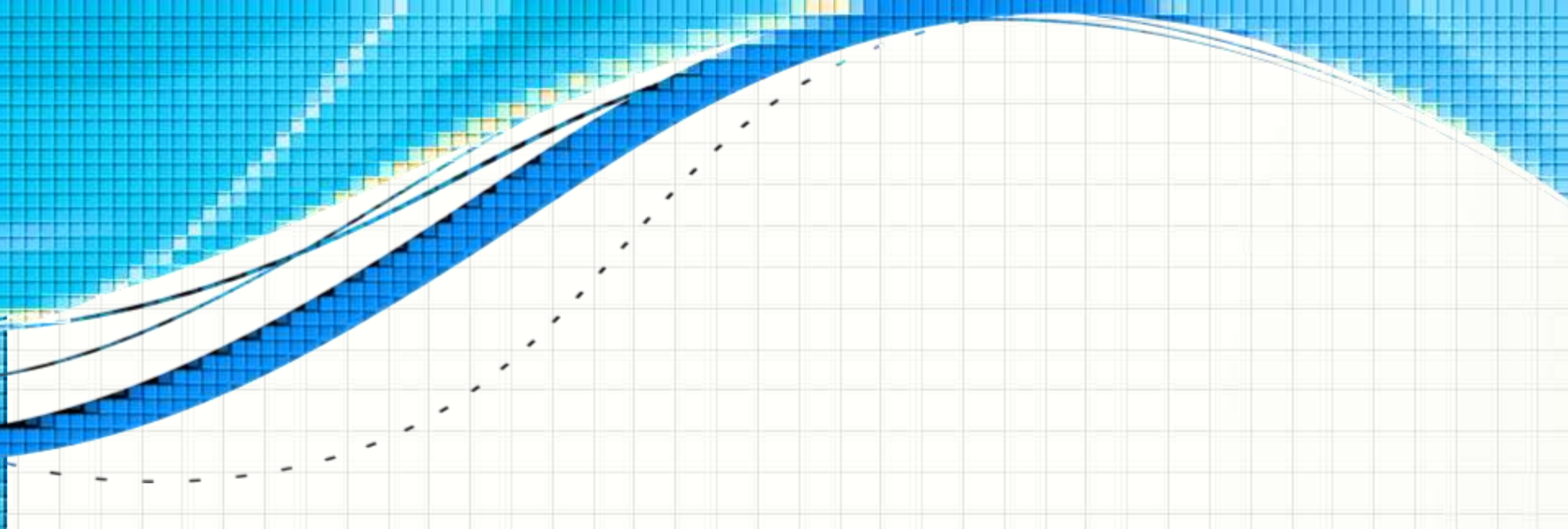
- Qual é o valor na posição de memória cujo endereço é **7**?

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**!



**COMO OS NÚMEROS
SÃO ARMAZENADOS:
O “ALFABETO” 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?



Humanos x Processadores

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



Humanos x Processadores

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



Humanos x Processadores

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



Humanos x Processadores

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



Humanos x Processadores

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



Humanos x Processadores

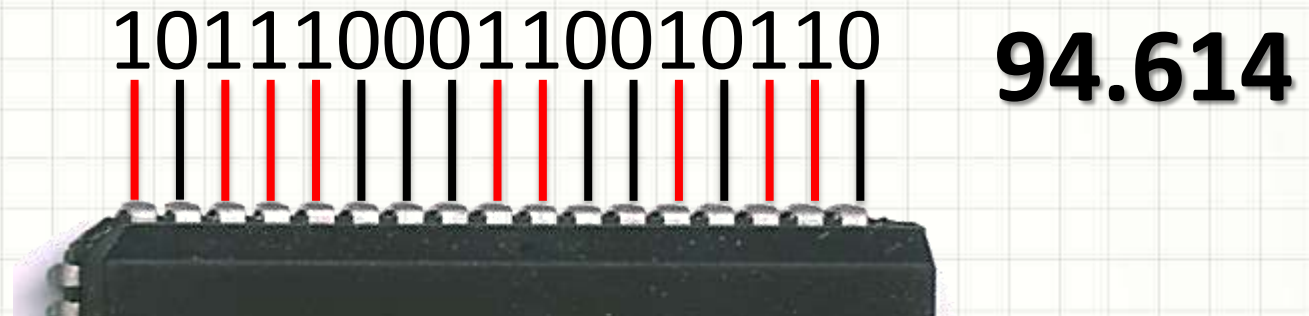
- Como indicar n^{os} decimais para o processador?



94.614

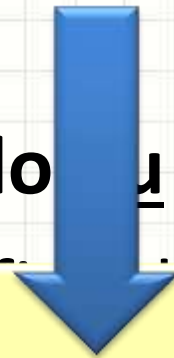
Como saber isso?

- 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:



Como saber isso?

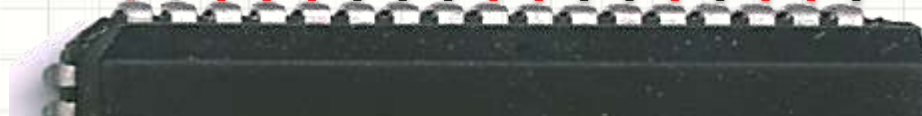
- Cada fio representa um “dígito” numérico, chamado **bit**
- Esse fio pode estar **desligado** ou **ligado**



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 múltiplos e submúltiplos

- Memória do computador: muitos dados!
 - Kibibyte (**KiB**): 1024 bytes
 - Mebibyte (**MiB**): 1024 KiB (~ 1mi de bytes)
 - Gibibyte (**GiB**): 1024 MiB (~ 1bi de bytes)
 - Tebibyte (**TiB**): 1024 GiB (~ 1tri de bytes)
 - Pebibyte (**PiB**): 1024 TiB (~ 1tetra de bytes)

Os múltiplos e submúltiplos

- Para discos (HardDisk): notação convencional
 - Quilobyte (**KB**): 1000 bytes
 - Megabyte (**MB**): 1000 KB (~ 1mi de bytes)
 - Gigabyte (**GB**): 1000 MB (~ 1bi de bytes)
 - Terabyte (**TB**): 1000 GB (~ 1tri de bytes)
 - Petabyte (**PB**): 1000 TB (~ 1tetra de bytes)



CONVERTENDO DE BINÁRIO PARA QUANTIDADE DECIMAL

Conversão de Binário para Decimal


- Regra prática: construa essa tabela

Multiplicador	32	16	8	4	2	1

Conversão de Binário para Decimal

- Regra prática: construa essa tabela


Multiplicador	32	16	8	4	2	1
101011b						


Número a Converter

Conversão de Binário para Decimal

- Regra prática: construa essa tabela

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



Repasse para as colunas (da direita para a esquerda)

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!

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

Exercícios

- Converter os valores abaixo para decimal
- 10b
- 100b
- 1000b
- 1101b
- 11110011001b



**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.
 - Divisão for exata: acrescentar 0 ao binário
 - Divisão “quebrada”: acrescentar 1 ao binário
 - “Jogue fora” o que vier depois da vírgula
- Com o resultado da divisão...
 - Repita 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

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

Fracionario!

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 $D \rightarrow B$

- Regra prática: converter 13 para binário

1101b

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

Fim!

Conversão D→B

- Regra prática: converter 13 para binário

1101b

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

13 = 1101b

Exercícios

- Converter os valores abaixo para binário
- 23
- 42
- 124

Qual o Problema com os Binários?

- **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!**

Qual o Problema com os Binários?

- Se ele guardar apenas 16 bits, por exemplo...
0,000110011001100110011001100110011001100110011...
- Que em decimal é... 0,099976
 $0,1 \neq 0,09976$
- Problemas!
 - Valores fracionários (reais) ocupam mais espaço
 - Valores fracionários não são exatos: erros!



COMO GUARDAR OUTROS DADOS NA MEMÓRIA?

Outros Dados na Memória

- Só números binários... 0101001010111b
- Mas o significado depende da interpretação!
 - Números inteiros, fracionários etc.
 - Letras
 - Sons
 - Imagens
 - ...
- Interpretação depende de uma convenção

Exemplos de Representação

- Números com sinal (simplificado)

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


Exemplos de Representação

- Números Reais (simplificado)
- Mantissa * $2^{\text{Exponente}}$
 - $0100b * 2^{-11b} =$
 - $4 * 2^{-3} =$
 - $4 / 8 =$
 - 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

Exemplos de Representação

- 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]

Exercício

- Como fica seu nome em ASCII?

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]



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

SAVA3!

- Surgimento e evolução do computador
- Formalizando a lógica
 - Como sistematizar soluções para problemas



PERGUNTAS?