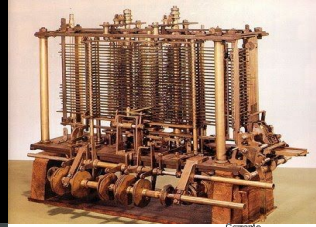
Programação Aplicada à Engenharia Civil

Professora Priscila Marques Kai
Aula 3

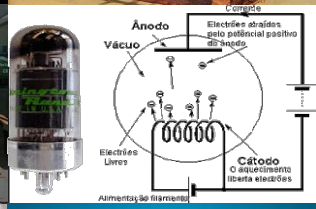
Conceitos de Computação e Computadores

Agora iremos discutir conceitos básicos de computação e computadores, incluindo a estrutura básica de um computador, organização de dados e execução de instruções.

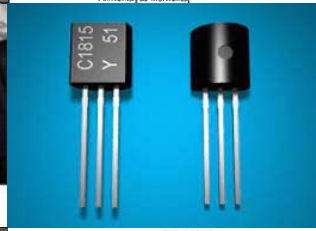
Geração zero de computadores - Computadores mecânicos



Primeira geração de computadores - Computadores à Válvula

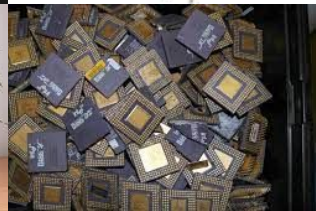


Segunda geração de computadores - Computadores Transistorizados

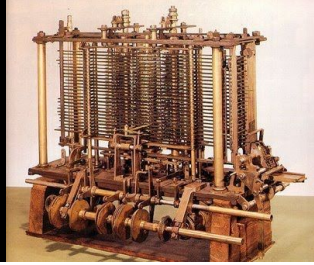


Terceiro geração de computadores - Computadores com Circuitos Integrados

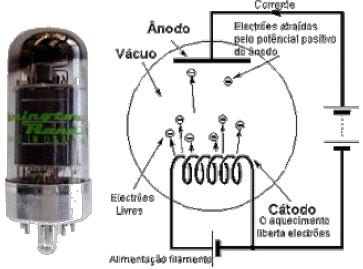
Quarta geração de computadores - Computadores com Chips VLSI



Geração zero de computadores - Computadores mecânicos



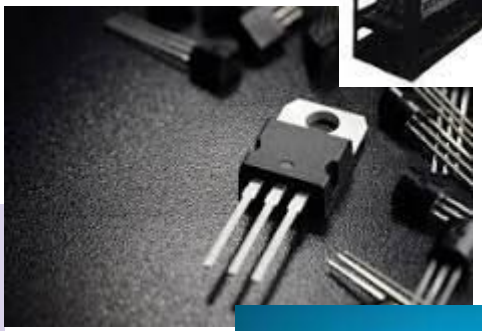
- Surgimento de máquinas que possuíam a capacidade de fazer multiplicações, por somas sucessivas, sendo os cálculos realizados por meio de engrenagens.
- Desenvolvimento de um sistema de lógica simbólica (lógica booleana)

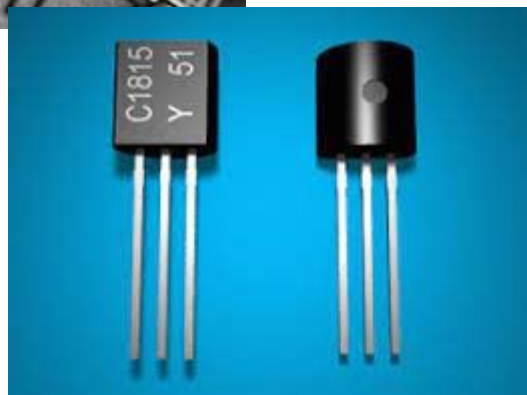


Primeira geração de computadores - Computadores à Válvula

- Os primeiros computadores eletrônicos surgiram na década de 1930, sendo esses computadores máquinas eletromecânicas baseadas em dispositivos por corrente elétrica funcionando como uma chave *liga-desliga*.

Segunda geração de computadores - Computadores Transistorizados



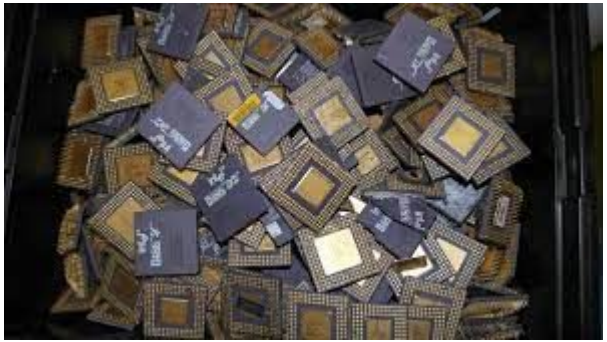


- Com a invenção do transistor em 1947, foi possível então a construção de máquinas mais confiáveis e com maior velocidade, além de serem menores. No entanto, dez anos foram necessários para seu uso em um computador

Terceiro geração de computadores - Computadores com Circuitos Integrados

- Após o surgimento do circuito integrado em 1958, por possibilitar comportar dezenas de transistores, possibilitou a execução de funções lógicas simples e complexas, necessitando de menor espaço e menor custo.

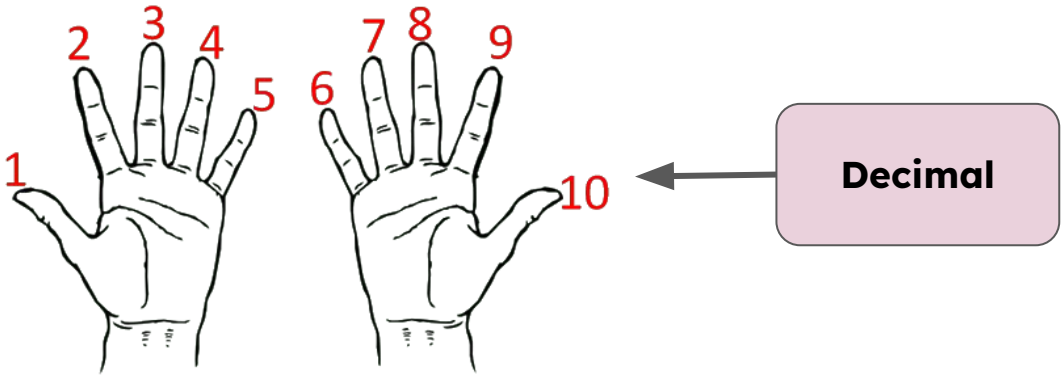
Quarta geração de computadores - Computadores com Chips VLSI



- Nesta geração temos o uso dos microprocessadores, consistindo de um dispositivo eletrônico encapsulado em um chip e uma unidade de controle, unidade lógica-aritmética e memória interna.
- Primeiro microprocessador: Intel 4004 (1971)

Computação: ato ou efeito de computar, verbo que exprime o ato de fazer contagem, de contar ou calcular.

Para representar quantidades relacionadas à **computação** foi necessária a criação de um sistema de numeração.



No caso de um computador, temos que os circuitos eletrônicos operam com sinais de dois níveis distintos ou **`binários`**.

Motivações:

- ➔ Simplicidade
- ➔ Menor custo de implementação
- ➔ Possibilidade de implementar hardware para qualquer tipo de função lógica

Os circuitos digitais binários também usam resultados da **lógica booleana**.

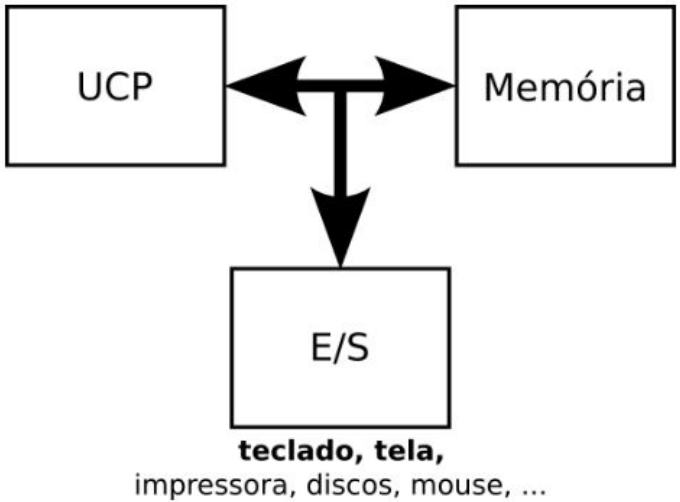
- Hardwares são implementados para que executem qualquer tipo de função lógica, permitindo a construção de diversos tipos de circuitos presentes em um computador, como memórias, microprocessadores, dentre outros.

Os circuitos digitais binários também usam resultados da **lógica booleana**.

Todos os dados armazenados e processados em um computador são traduzidos em sinais elétricos como um conjunto de **0s** e **1s**.

Arquitetura de um computador

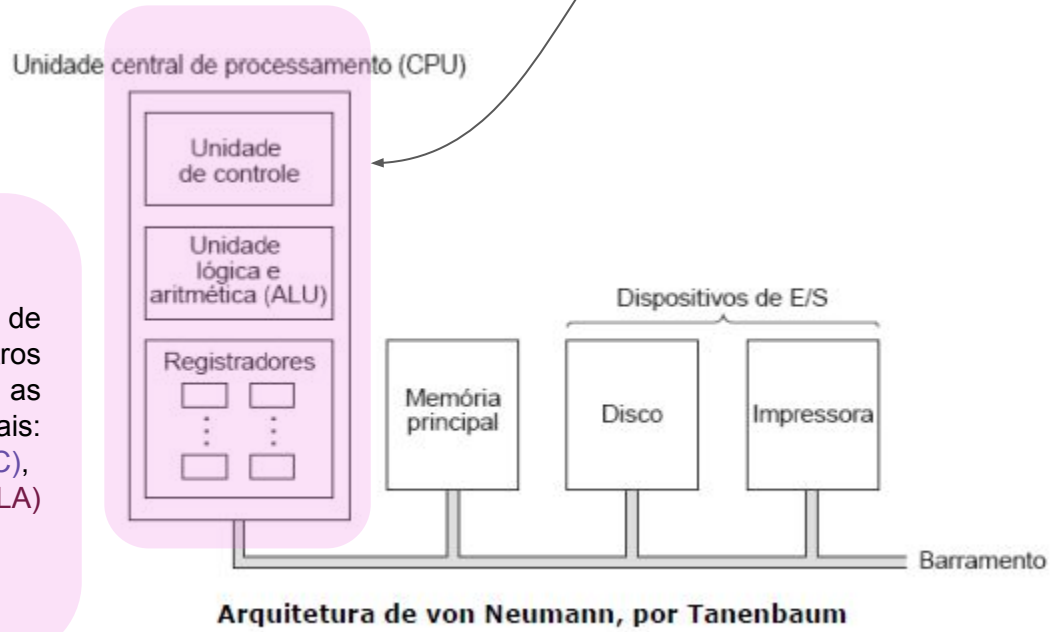
Internamente, os computadores modernos podem ser identificados por três componentes essenciais: a Unidade Central de Processamento (CPU), a Memória (MEM) e os dispositivos de Entrada e Saída (E/S).



Arquitetura de um computador

Busca instruções na memória principal

Modelo de arquitetura de Von Neumann



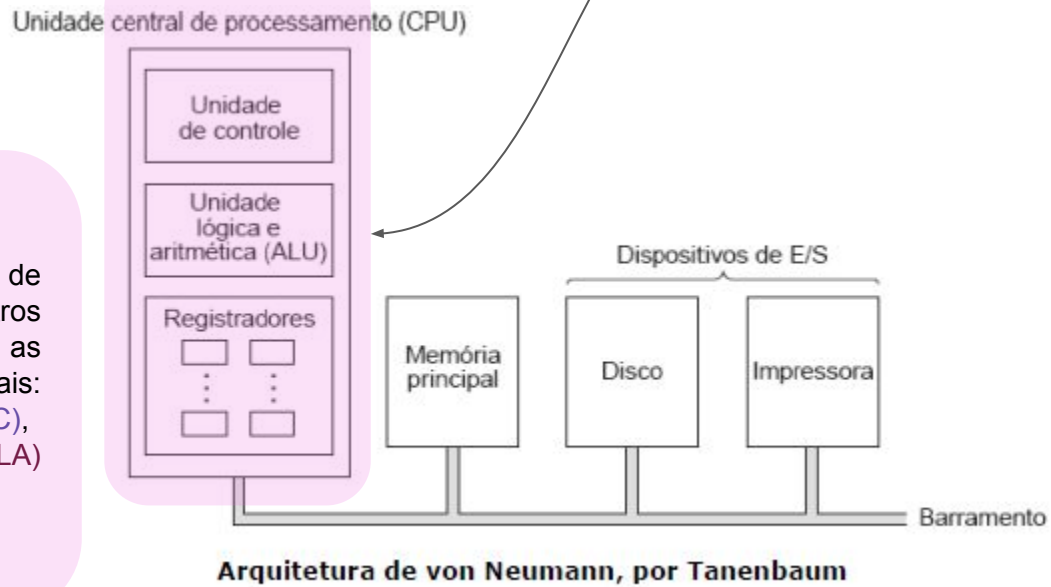
Arquitetura de von Neumann, por Tanenbaum

Responsável pela execução de programas e controle dos outros componentes. Possui as seguintes unidades funcionais: Unidade de controle (UC), unidade lógico-aritmética (ULA) e conjunto de registradores.

Arquitetura de um computador

Executa operações lógicas e aritméticas além de instruções advindas da memória principal.

Modelo de arquitetura de Von Neumann



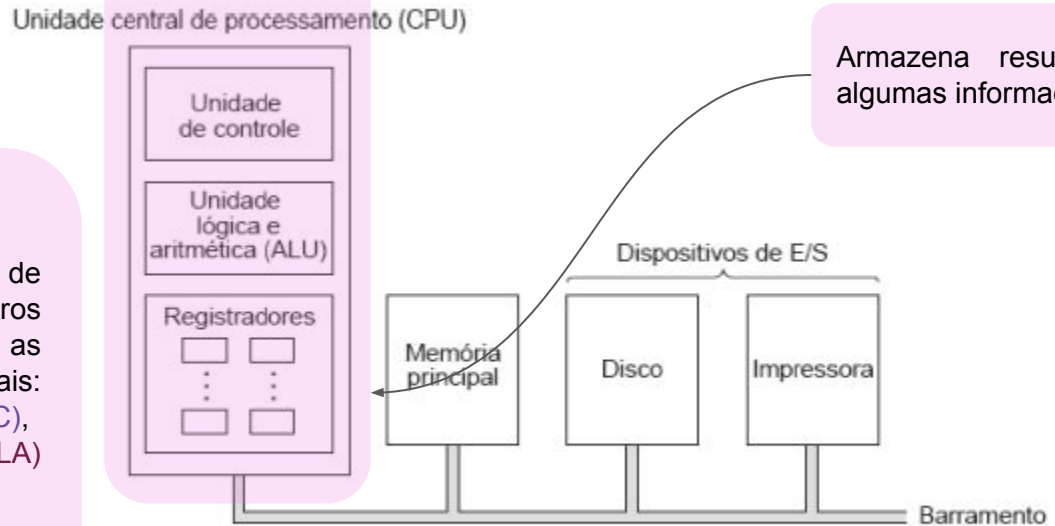
Arquitetura de von Neumann, por Tanenbaum

Responsável pela execução de programas e controle dos outros componentes. Possui as seguintes unidades funcionais: Unidade de controle (UC), unidade lógico-aritmética (ULA) e conjunto de registradores.

Arquitetura de um computador

Modelo de arquitetura de Von Neumann

Responsável pela execução de programas e controle dos outros componentes. Possui as seguintes unidades funcionais: Unidade de controle (UC), unidade lógico-aritmética (ULA) e conjunto de registradores.

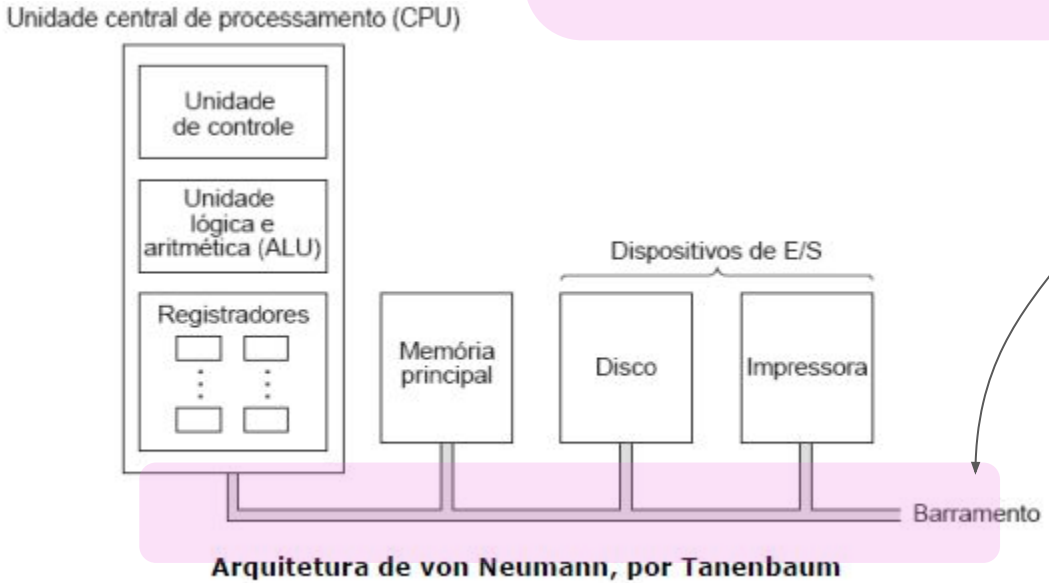


Arquitetura de von Neumann, por Tanenbaum

Arquitetura de um computador

Modelo de arquitetura de Von Neumann

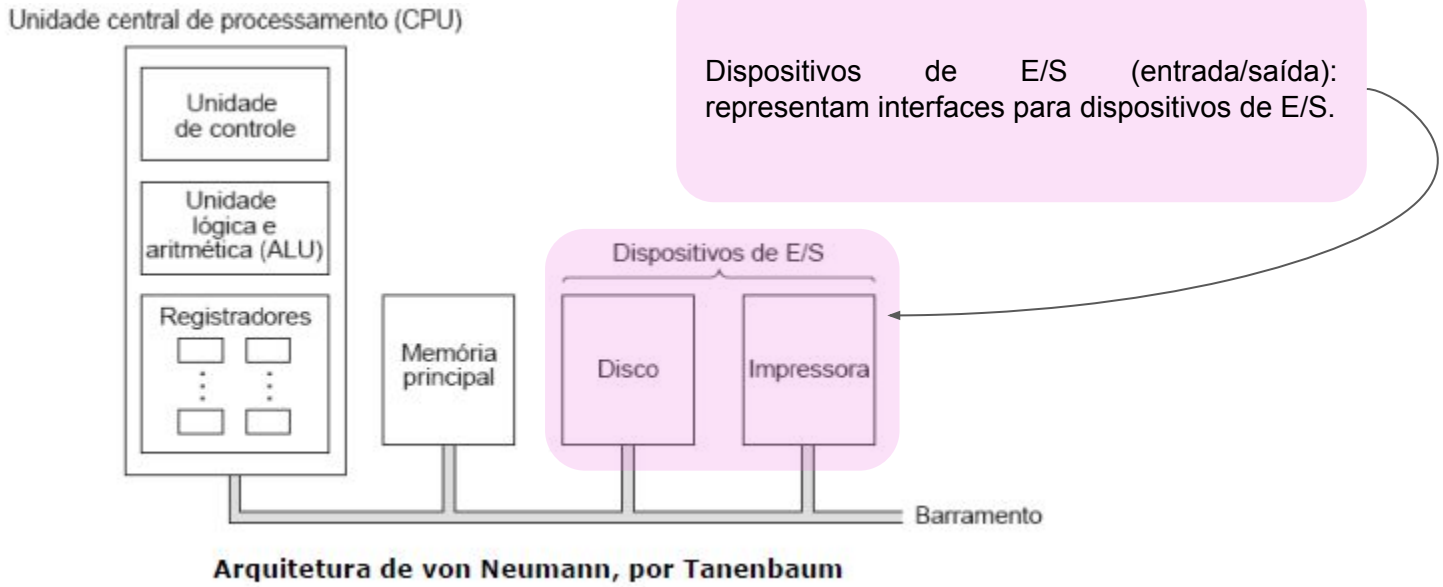
Barramento - possui o propósito de transmitir dados (endereços de áreas de memória, dados de um programa armazenados na memória)



Arquitetura de von Neumann, por Tanenbaum

Arquitetura de um computador

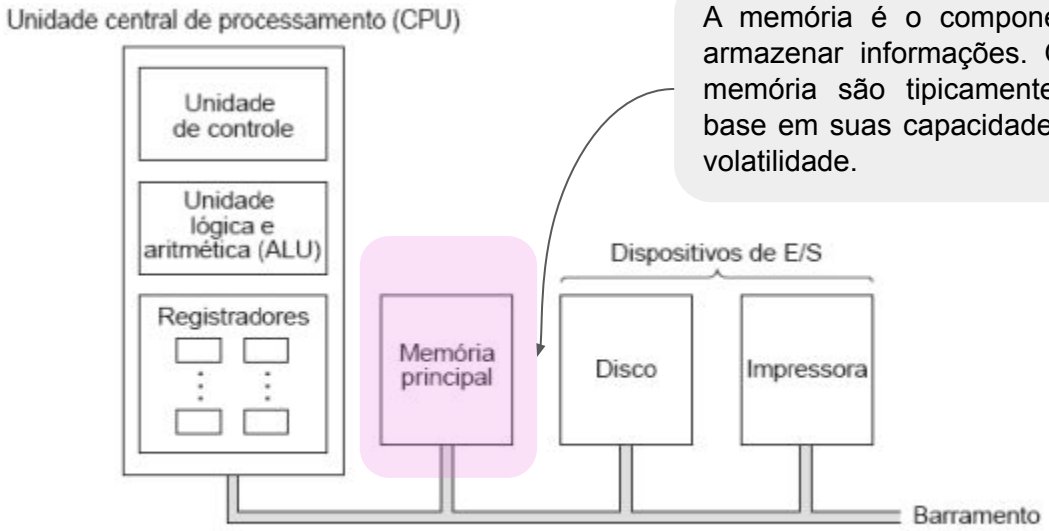
Modelo de arquitetura de Von Neumann



Arquitetura de von Neumann, por Tanenbaum

Arquitetura de um computador

Modelo de arquitetura de Von Neumann



Arquitetura de von Neumann, por Tanenbaum

A memória é o componente encarregado de armazenar informações. Os diversos tipos de memória são tipicamente categorizados com base em suas capacidades de leitura, escrita e volatilidade.

Construção de programas

Algoritmo

Um algoritmo pode ser descrito como um conjunto finito de etapas (instruções) destinadas a solucionar um problema específico.

Para desenvolvermos um algoritmo é necessário:

- Definir as instruções de maneira simples, sem ambiguidade;
- Organizar as instruções de forma ordenada
- Definir uma sequência finita de passos

Construção de programas

Algoritmo

Algoritmos possuem a capacidade de executar diversas tarefas, como:

- Ler e escrever dados;
- Avaliar expressões algébricas, relacionais e lógicas;
- Efetuar escolhas com base nos resultados das expressões analisadas;
- Repetir instruções de acordo com determinadas condições.

Construção de programas

Partes de um Algoritmo

Um algoritmo, quando implementado em um computador, é composto, no mínimo, por três componentes que incluem:

1. Aquisição de dados (Entrada);
2. Manipulação de dados (Processamento);
3. Exibição de dados (Saída).

Construção de programas

Tipos de Linguagem

1. Linguagem natural
2.

Construção de programas

Tipos de Linguagem

1. Linguagem natural
2. Linguagem de Máquina e Assembler

> As instruções devem ser passadas ao computador em uma linguagem que ele possa compreender, conhecida como linguagem de máquina, consistindo exclusivamente em números, representados em formato binário, representando tanto as operações quanto os dados a serem utilizados no processamento do programa.

```
00000000   7F 45 4C 46  01 01 01 00  00 00 00 00  00 00 00 00  .ELF............
00000010   02 00 03 00  01 00 00 00  D0 82 04 08  34 00 00 00  ............4...
00000020   BC 0C 00 00  00 00 00 00  34 00 20 00  07 00 28 00  ........4. ...(.
```

Construção de programas

Tipos de Linguagem

1. Linguagem natural
2. Linguagem de Máquina e Assembler

> As instruções devem ser passadas ao computador em uma linguagem que ele possa compreender, conhecida como linguagem de máquina, consistindo exclusivamente em números, representados em formato binário, representando tanto as operações quanto os dados a serem utilizados no processamento do programa.

```
00000000   7F 45 4C 46  01 01 01 00  00 00 00 00  00 00 00 00  .ELF............
00000010   02 00 03 00  01 00 00 00  D0 82 04 08  34 00 00 00  ............4...
00000020   BC 0C 00 00  00 00 00 00  34 00 20 00  07 00 28 00  ........4. ...(.
```

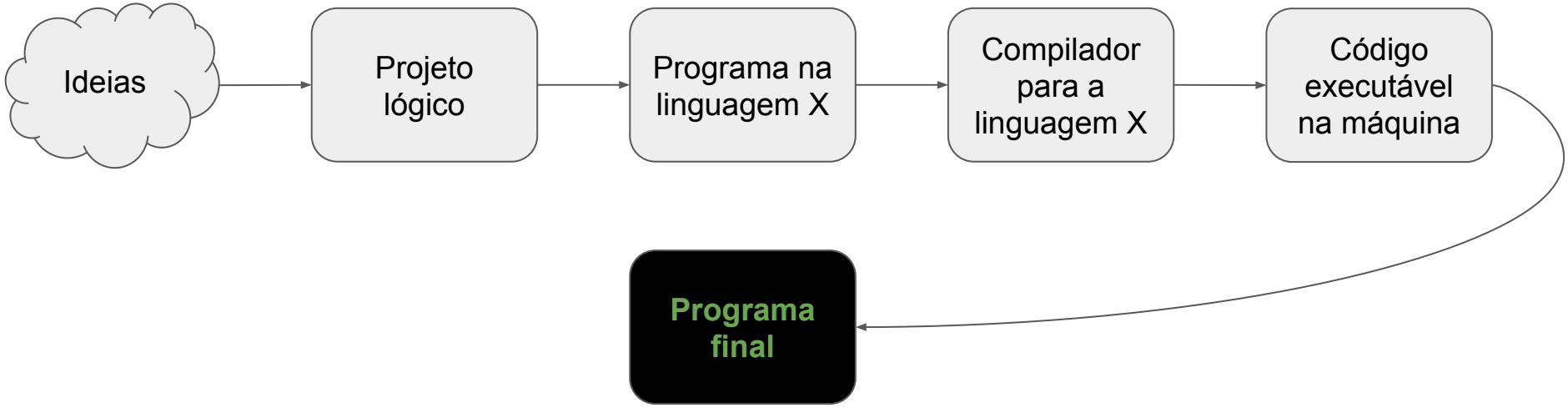
Construção de programas

Construção de programas

Linguagens de Programação

As linguagens de programação têm a finalidade de tornar a escrita de instruções que o computador executará mais semelhante à linguagem humana, como um meio-termo entre a linguagem de máquina e a linguagem natural.

Linguagens mais semelhantes à linguagem de máquina são categorizadas como linguagens de baixo nível, enquanto que linguagens de alto nível se aproximam mais da linguagem natural.



Linguagem de baixo nível

Linguagem mais próxima à linguagem de máquina mais voltada à programação de softwares de sistema.

Linguagem de alto nível

Linguagem mais compreensível ao ser humano.

Construção de programas

Pseudocódigo: meio-termo entre a linguagem natural e uma linguagem de programação.

- Utiliza um conjunto limitado de palavras-chave, com equivalências nas linguagens de programação;
- Foca mais na lógica do algoritmo;
- Pode ser traduzido para uma linguagem de programação.

Construção de programas

Pseudocódigo: meio-termo entre a linguagem natural e uma linguagem de programação.

```
ALGORITMO Multipla_numeros
var
     numero1 : INTEIRO
     numero2 : INTEIRO
     resultado : INTEIRO
INÍCIO
     LEIA(numero1)
     LEIA(numero2)
     resultado ← numero1 * numero2
     ESCREVA("Resultado da multiplicação: ", resultado)
FIM
```

Bit

“Existem 10 tipos de pessoas, as que entendem binário e as que não entendem…”

BIT (*Binary digit*) - representação de estado ligado/desligado ou `binário` em dispositivos eletrônicos. Usa-se potência de base **2**.

- 1 representa bit ligado
- 0 representa bit desligado

Bit

Exemplo:

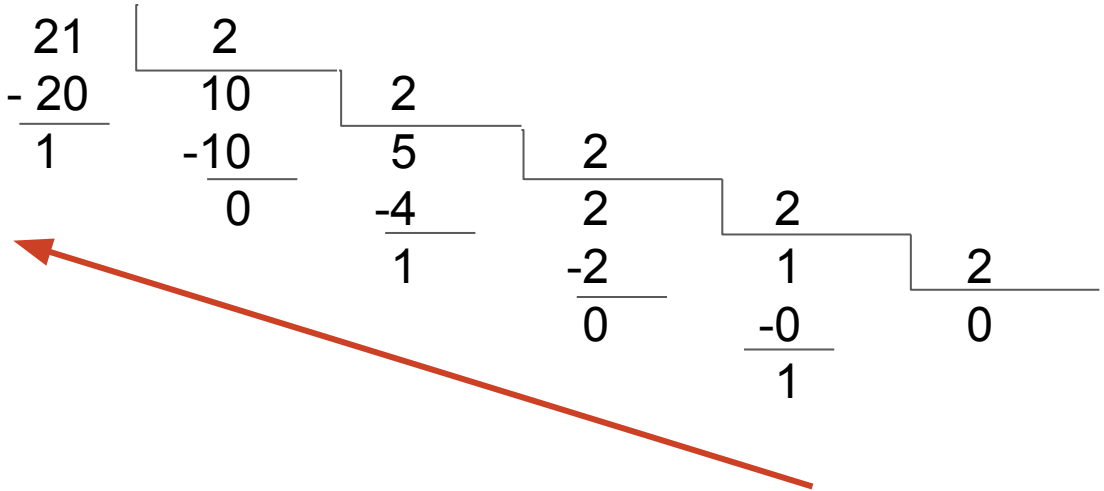
O número **10101_2** representa:

Decimal

$$10101_2 = 1x2^4 + 0x2^3 + 1x2^2 + 0x2^0 = 21_{10}$$

E como transformar decimal para binário?

Bit



$10101_2 = 21_{10}$

Bit

Na prática são utilizados múltiplos do BIT, como o **byte**.

Byte: **8 BITS**

Múltiplos do Byte:

Quilobyte (KB) - 1.024(2^{10}) bytes
Megabyte (MB) - 1.048.576(2^{20}) bytes
Gigabyte (GB) - 1.073.741.824(2^{30})bytes

Bit

Durante a execução de um programa as informações manipuladas referem-se aos **dados** e **instruções** que operam sobre esses dados.

Na memória são sempre representados por **bits**.

Caracteres: são representados como números binários no computador por meio de uma *tabela de caracteres*.

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
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	_		

Bit

A letra B ao ser digitada, é representada pelo decimal 66 (em bits: **01000010**)

Exercício: Abra um editor de texto, salve um arquivo apenas com a letra **B** e verifique o tamanho do arquivo. Após, edite o arquivo e coloque mais uma letra **B**. Salve e verifique novamente o tamanho.

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
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	_		

O papel da lógica em programação

Lógica corresponde a uma área na matemática com o objetivo de **investigar a veracidade de suas proposições**.

Considere as seguintes proposições:

1. Se estiver ensolarado, eu usarei óculos de sol.
2. Está ensolarado.

O que podemos concluir a partir delas?

O papel da lógica em programação

1. Se estiver ensolarado, eu usarei óculos de sol.
2. Está ensolarado.

Através das proposições podemos concluir que:

“Irei utilizar meu óculos de sol.”

Essa conclusão é derivada de uma *implicação lógica* na primeira proposição, no qual diz que “se estiver ensolarado” implica “eu usarei óculos de sol”, servindo de **regra** que direciona a consequência do fato.

O papel da lógica em programação

Mas agora, considere a seguinte mudança na segunda proposição:

1. Se estiver ensolarado, eu usarei óculos de sol.
2. Eu usarei óculos de sol.

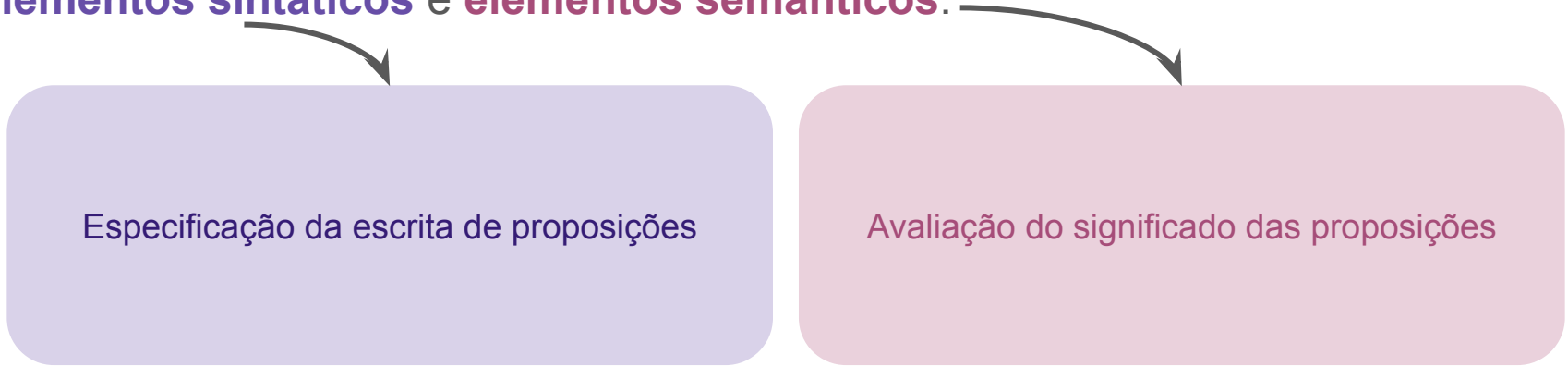
Conclui-se que:

"É provável que esteja ensolarado."

Dessa maneira, não se pode afirmar com precisão se o dia de fato está ensolarado.

O papel da lógica em programação

Assim, a lógica formada de proposições deve ser construída por uso de **elementos sintáticos** e **elementos semânticos**.



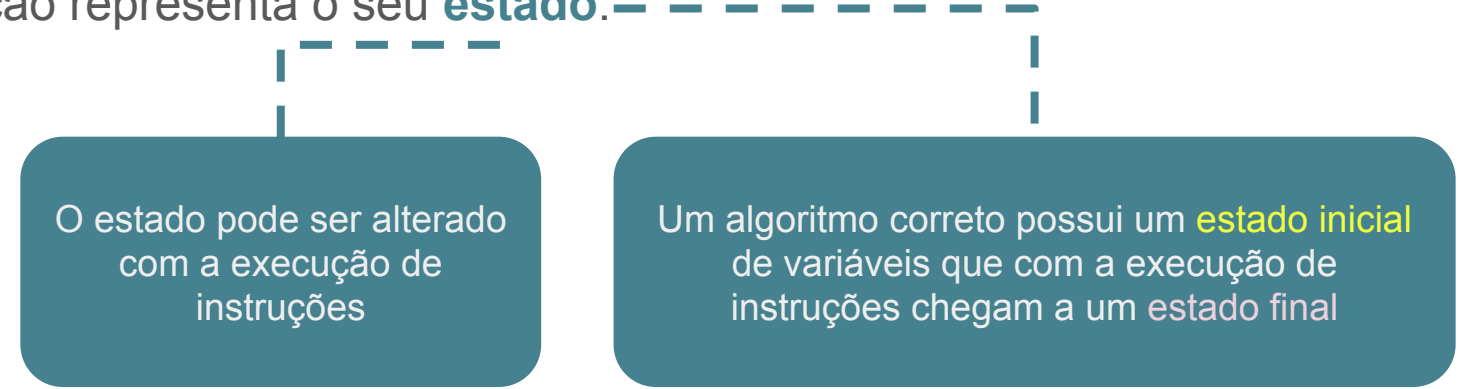
O resultado da avaliação pode pode ser somente um entre dois possíveis:

VERDADEIRO ou FALSO

O papel da lógica em programação

Com isso, o uso da **lógica na programação** está voltada na correta definição da sequência de instruções para que o programa desenvolvido funcione corretamente.

Quando um algoritmo está em execução, temos que o valor das variáveis a cada instrução representa o seu **estado**.



Pontos importantes na solução de problemas

1. Ao tentar solucionar um problema novo, entenda-o primeiro;
2. Faça a criação de um plano contendo a solução do problema:
    a. Caso o problema por muito complexo, tente dividi-lo em pedaços menores,
    b. É possível resolver o problema de maneira simplificada? Como?
3. Formalize a solução:
    a. Crie um algoritmo informal contendo as etapas que resolvam o problema,
    b. Verifique a corretude da solução,
    c. Escreva a solução por uso de fluxograma ou outra técnica.
4. Examine os resultados:
    a. Teste a solução por uso de diferentes entradas e verifique as saídas,
    b. Caso apresente problemas nos resultados, analise a sintaxe e comandos usados na solução
5. Otimize os resultados:
    a. Caso seja possível, melhore o algoritmo.

Treinando alguns comandos em Python

Algumas instruções básicas:

Input: Dados que podem ser obtidos do teclado, de um arquivo, da rede ou de algum outro dispositivo.

Saída: Obtenção de resultados, permitindo exibi-los na tela, salvá-los em um arquivo, enviá-los pela rede, etc.

Matemática: Execução de operações matemáticas básicas, como adição, subtração, divisão, exponenciação, multiplicação, entre outros.

Execução condicional: Verificação de certas condições e execute o código apropriado.

Repetição: Execução de alguma ação repetidamente, geralmente com alguma variação.

Operações Aritméticas

Python possui alguns **símbolos especiais** que representam cálculos, contendo vários operadores, como adição, subtração, multiplicação e divisão.