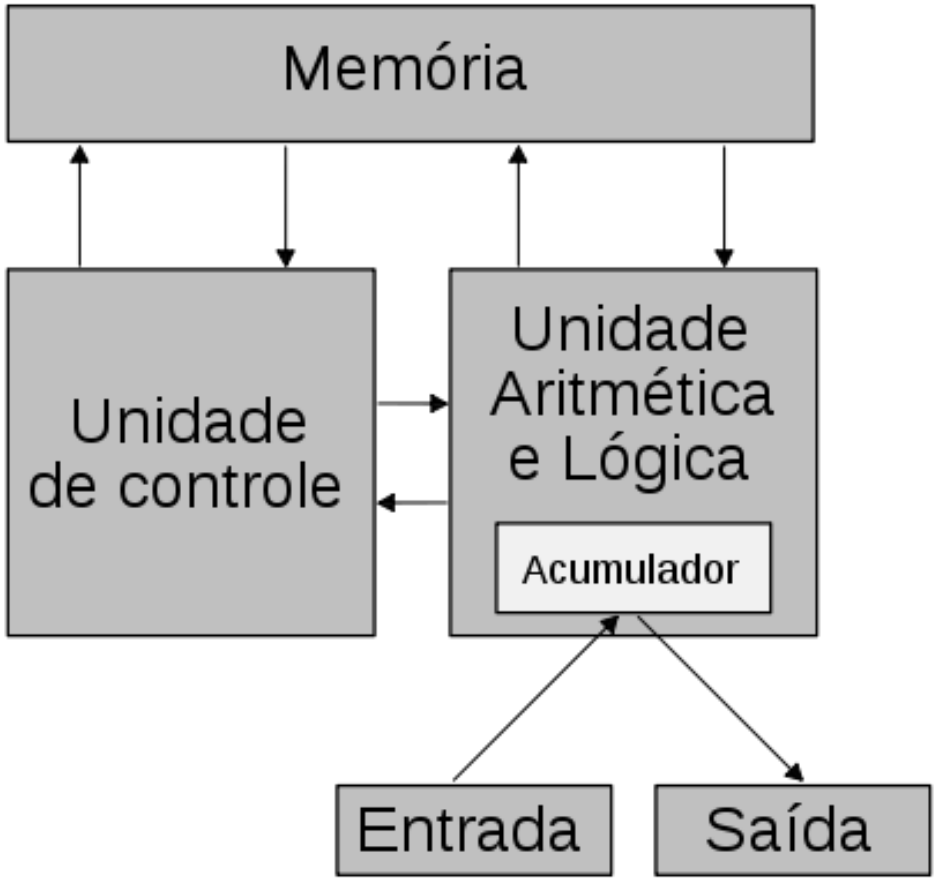
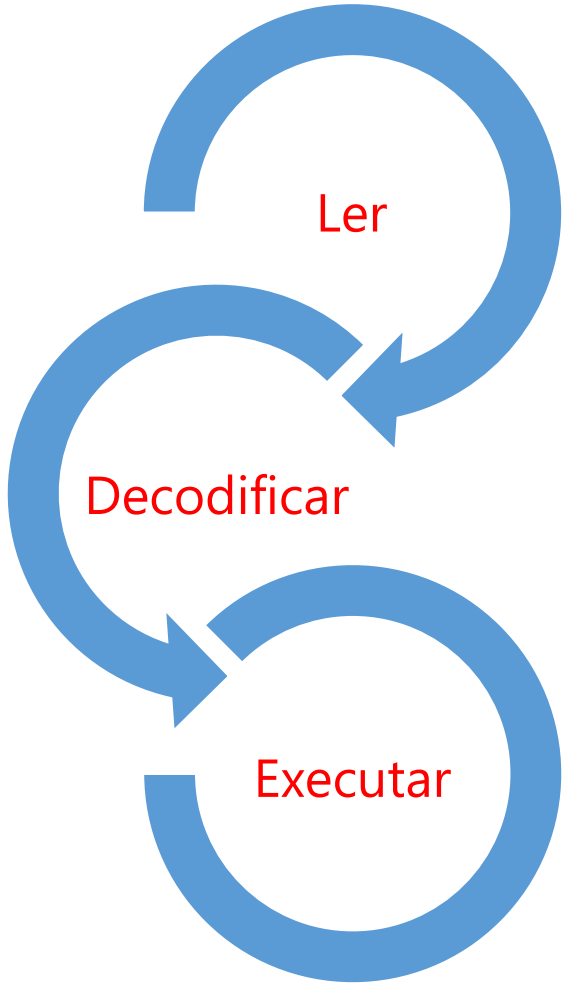
Capitulo 0

Componentes básicos

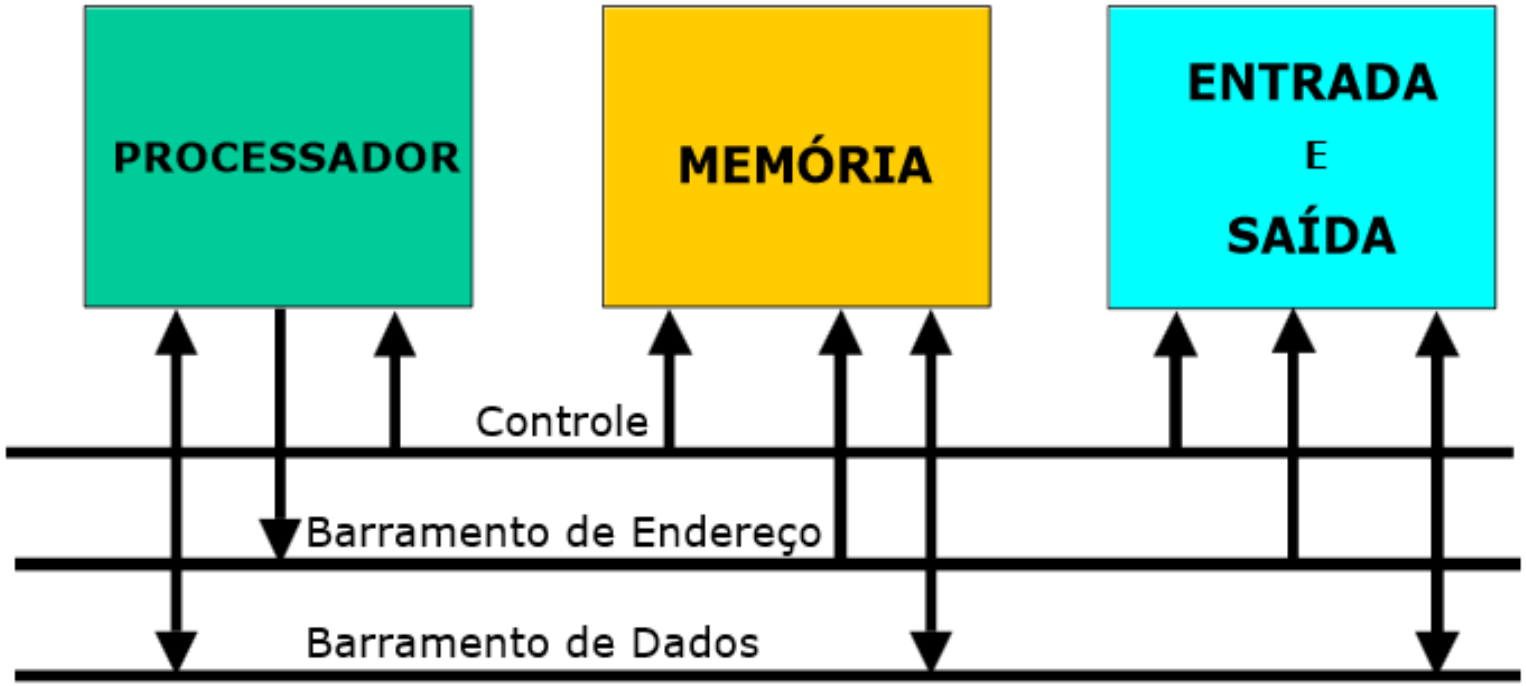
Máquina de Von Neumann



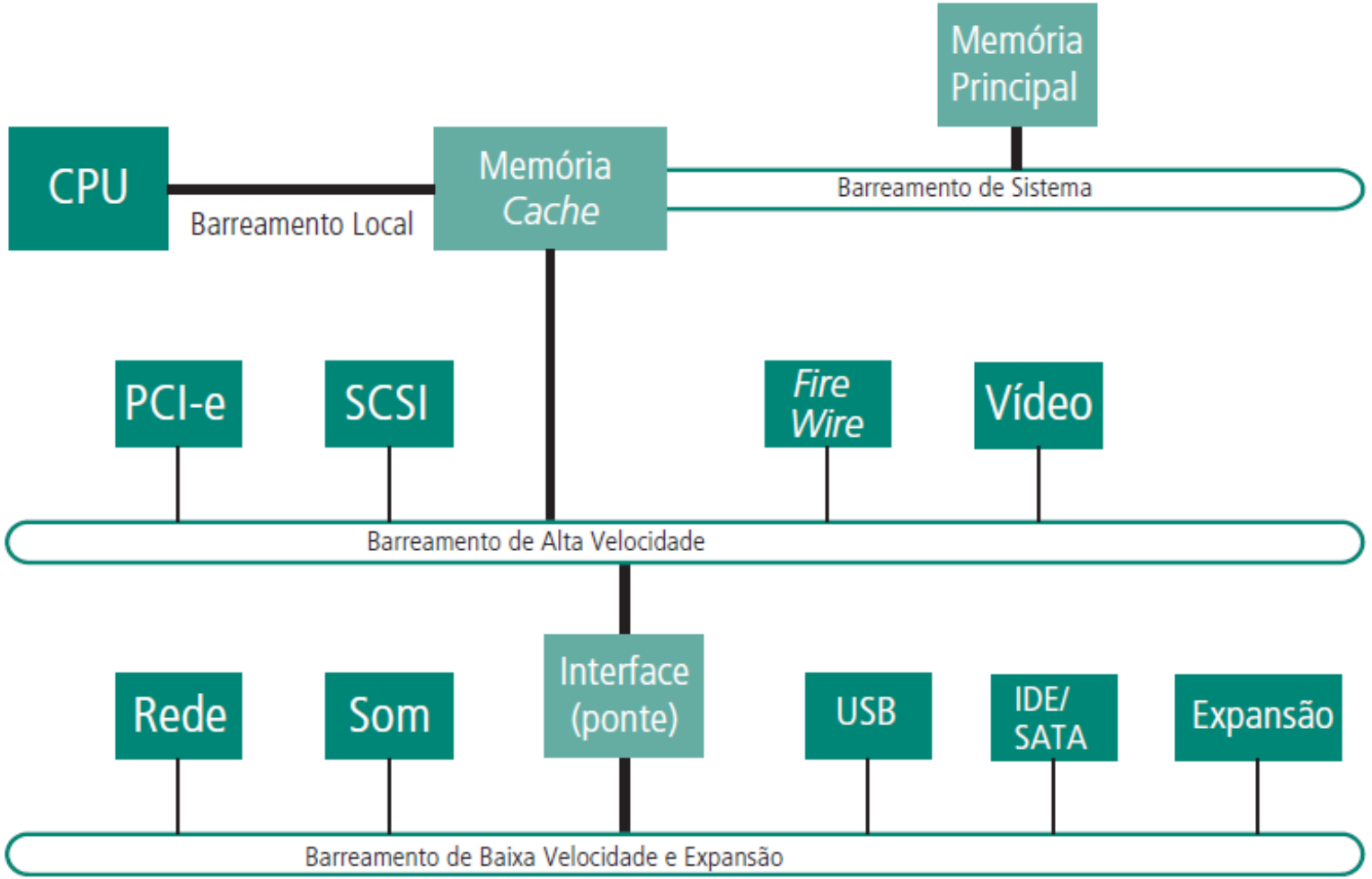
Ciclo de von Neumann



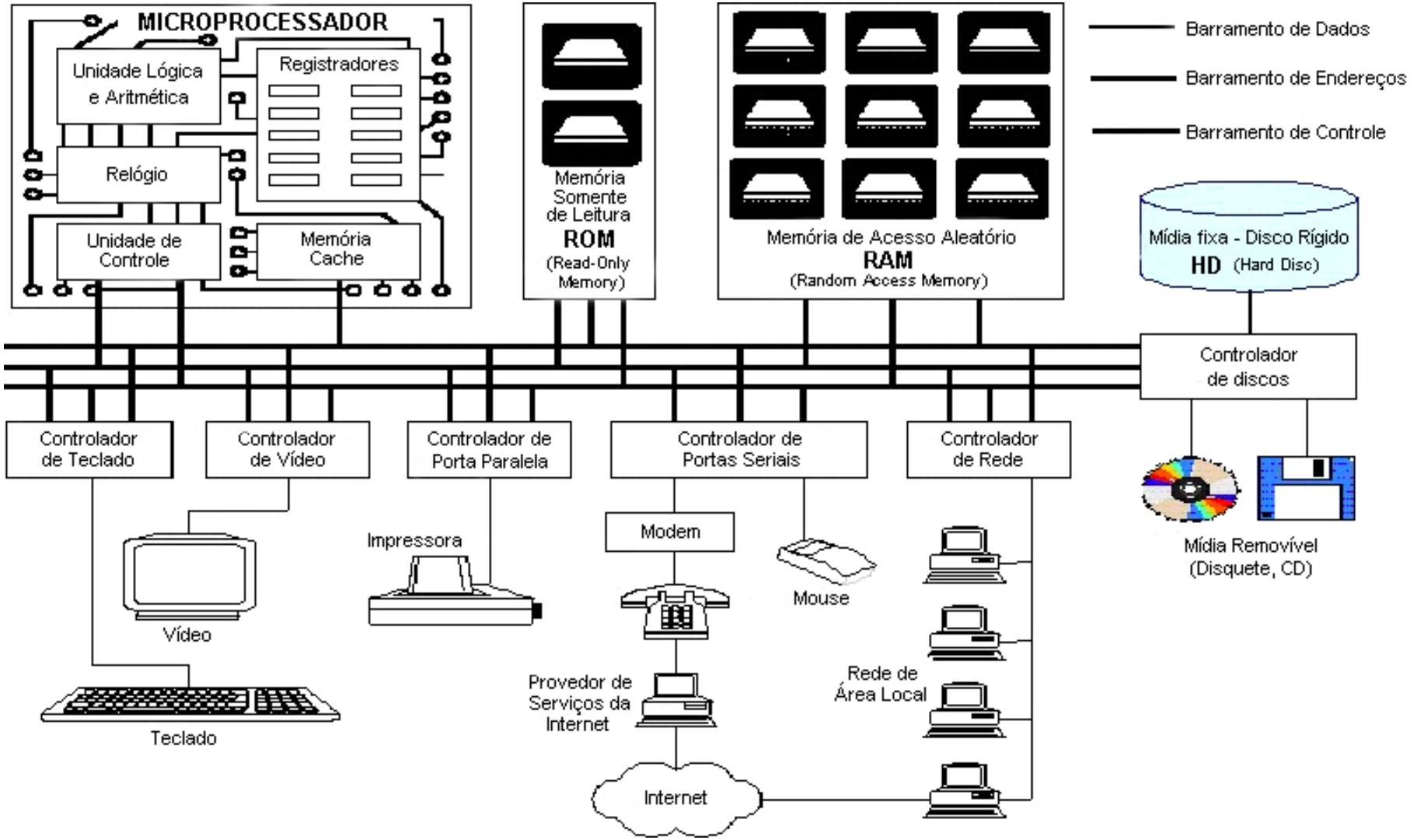
Modelo Barramento



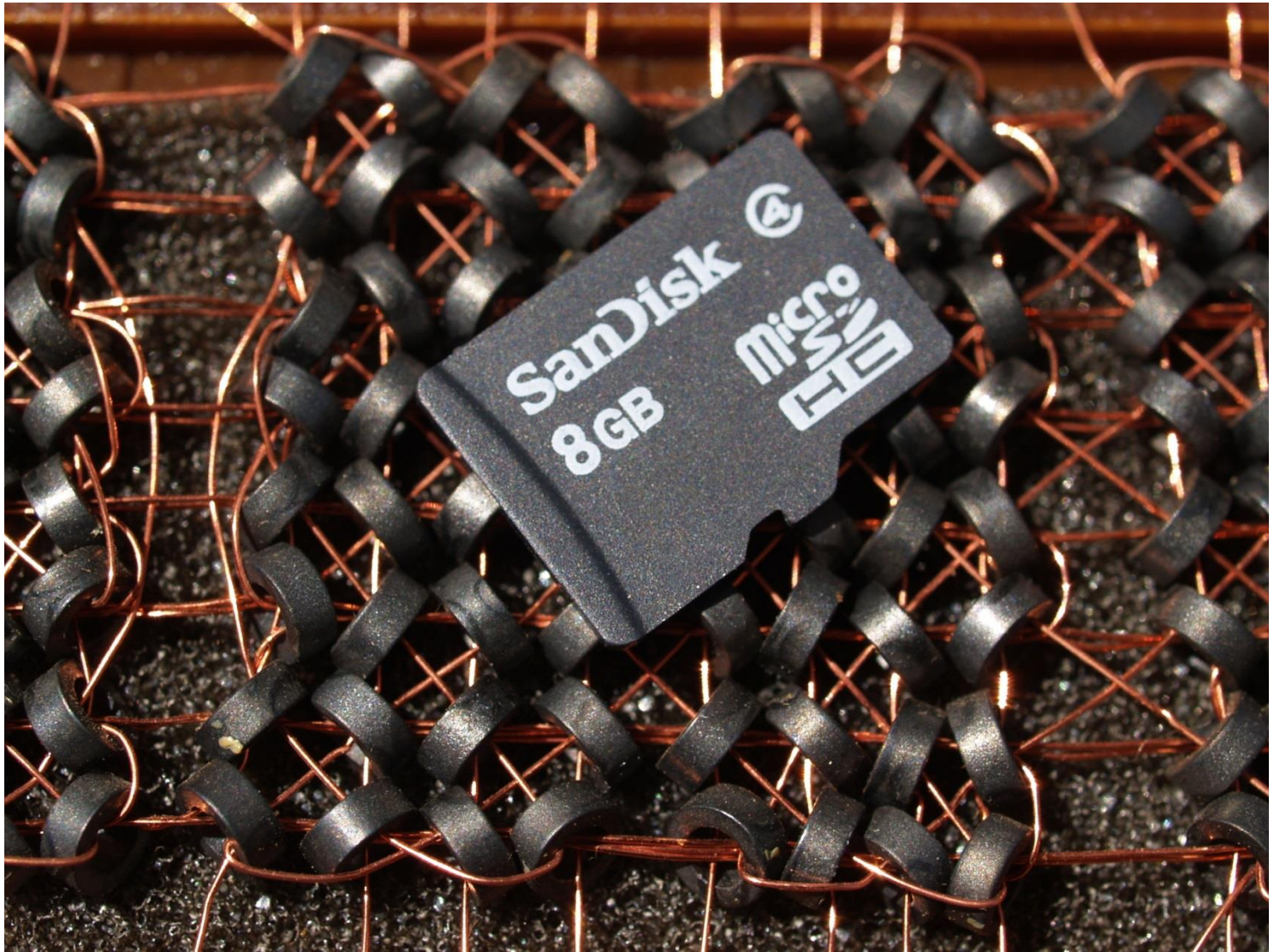
Modelo múltiplos barramentos



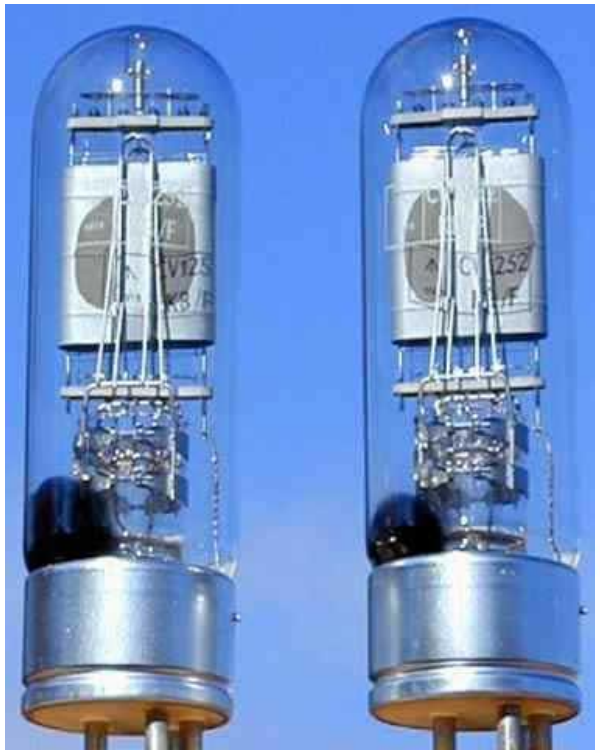
Modelo Barramento

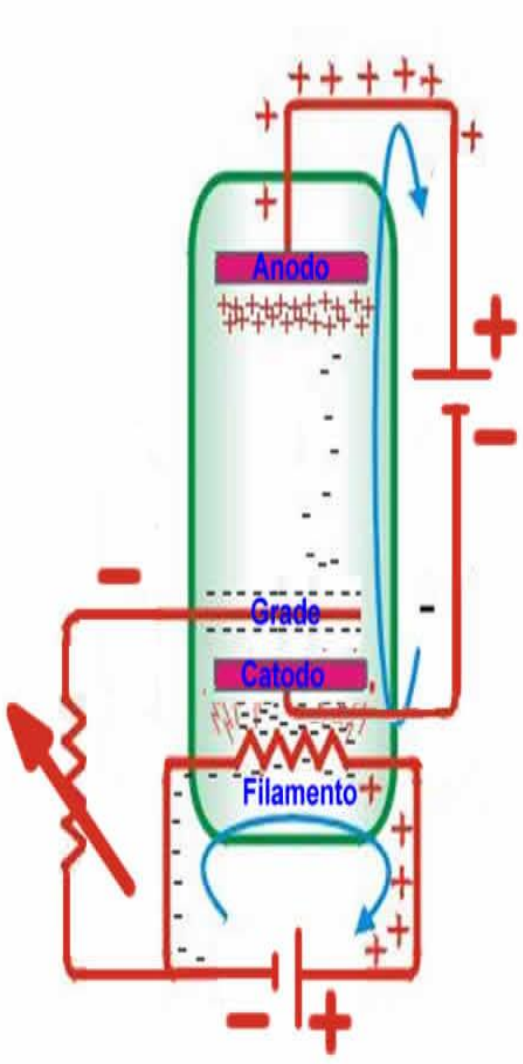


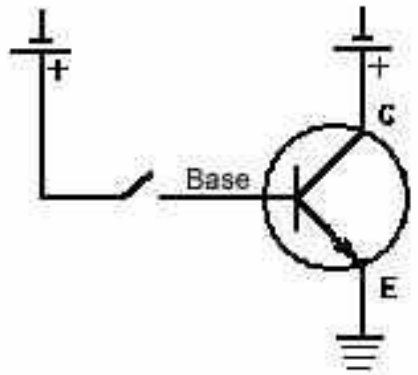
Bits e Bytes-Memoria



Componente Eletrônico







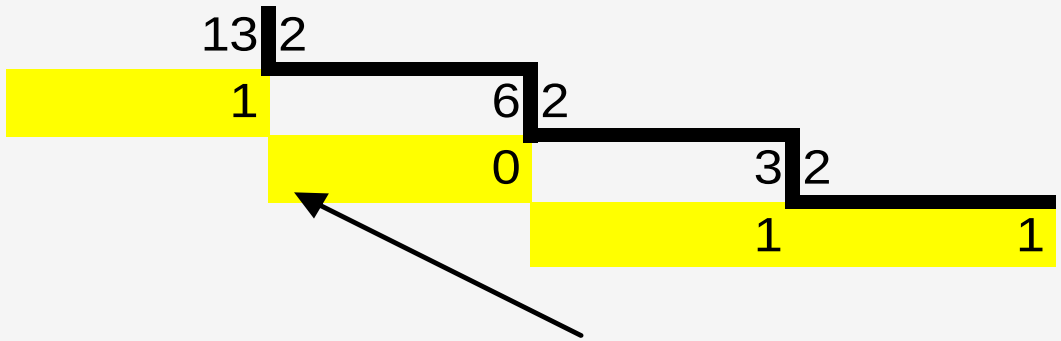
Sistemas de numeração

- Decimal

Quando falamos do número 123, imaginamos certo número de itens que esse número representa e esquecemos o seu significado matemático. Na realidade 123 representa:

$(1 \times 10^2) + (2 \times 10^1) + (3 \times 10^0)$, ou seja: $100 + 20 + 3 = 123$

Converter o numero 13(decimal) para binário



8	4	2	1	decimal
1	1	0	1	binário

8x1+4x1+2x0+1=13

Binário – Número 202

$(1 \times 2^7) + (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) + (0 \times 2^0)$, ou seja: $128 + 64 + 0 + 0 + 8 + 0 + 2 + 0 = 202$

Hexadecimal

O sistema hexadecimal representa os números em base 16. É usado na informática, pois os computadores costumam utilizar o *byte* como unidade básica da memória e com um *byte* podemos representar 256 valores possíveis, o que abrange todo alfabeto (maiúsculas e minúsculas), os números e vários caracteres especiais.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Nesse sistema, o **A** vale **10**, o **B** vale **11**, etc, até o **F**, que vale **15**.

Cada algarismo é multiplicado por uma potência de 16. Os expoentes de 16 são numerados da direita para a esquerda começando com 0 (zero).

Exemplo: $3E0 = 3 \times 16^2 + 14 \times 16^1 + 0 \times 16^0 = 992$

Exercício

- Converta o numero 64 para binário
- Some – em binário- o valor 4
- Responda qual o valor obtido em binário

Tipos de codificação

Codificação	Descrição
BCD – *Binary Coded Decimal*	É um grupo de 6 *bits*/caractere, o que permite a codificação de 64 caracteres (26). Praticamente não mais utilizado.
EBCDIC – *Extended Binary Coded Decimal Interchange Code*	Exclusivo da IBM, com agrupamentos de 8 *bits*, permitindo a codificação de 256 símbolos diferentes.
ASCII – *American Standard Code for Information Interchange*	Usado pelos demais fabricantes, há uma versão com 8 *bits*, desenvolvida para aplicações com os microcomputadores PC e compatíveis.
UNICODE	Um código de 16 *bits*, que pode representar 65.536 símbolos diferentes. Isso praticamente resolve os problemas de suficiência de códigos aos símbolos, já que podem atender de forma universal a todas as linguagens (grego, hebraico, chinês, japonês, francês, inglês, espanhol, símbolos da União Européia, etc.). Há um consórcio de empresas que escreve as revisões, que pode ser obtida no *site* da *Web* (www.unicode.org).

ASCII-*American Standard Code for Information Interchange*

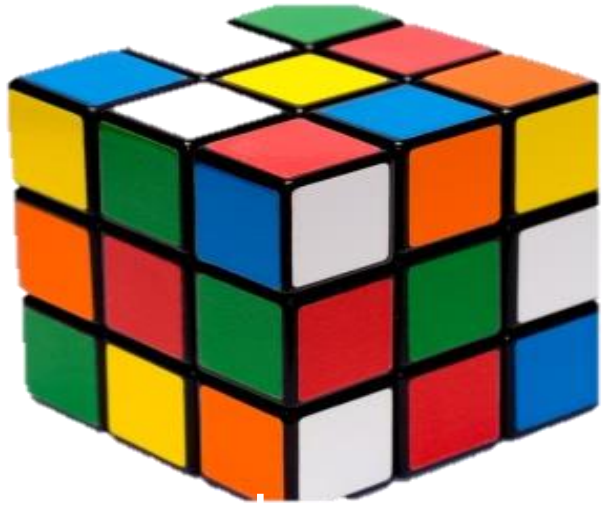
Sinais gráficos (imprimíveis) [editar | editar código-fonte]

Bin	Oct	Dec	Hex	Sinal
0010 0000	040	32	20	(espaço)
0010 0001	041	33	21	!
0010 0010	042	34	22	"
0010 0011	043	35	23	#
0010 0100	044	36	24	$
0010 0101	045	37	25	%
0010 0110	046	38	26	&
0010 0111	047	39	27	'
0010 1000	050	40	28	(
0010 1001	051	41	29	)

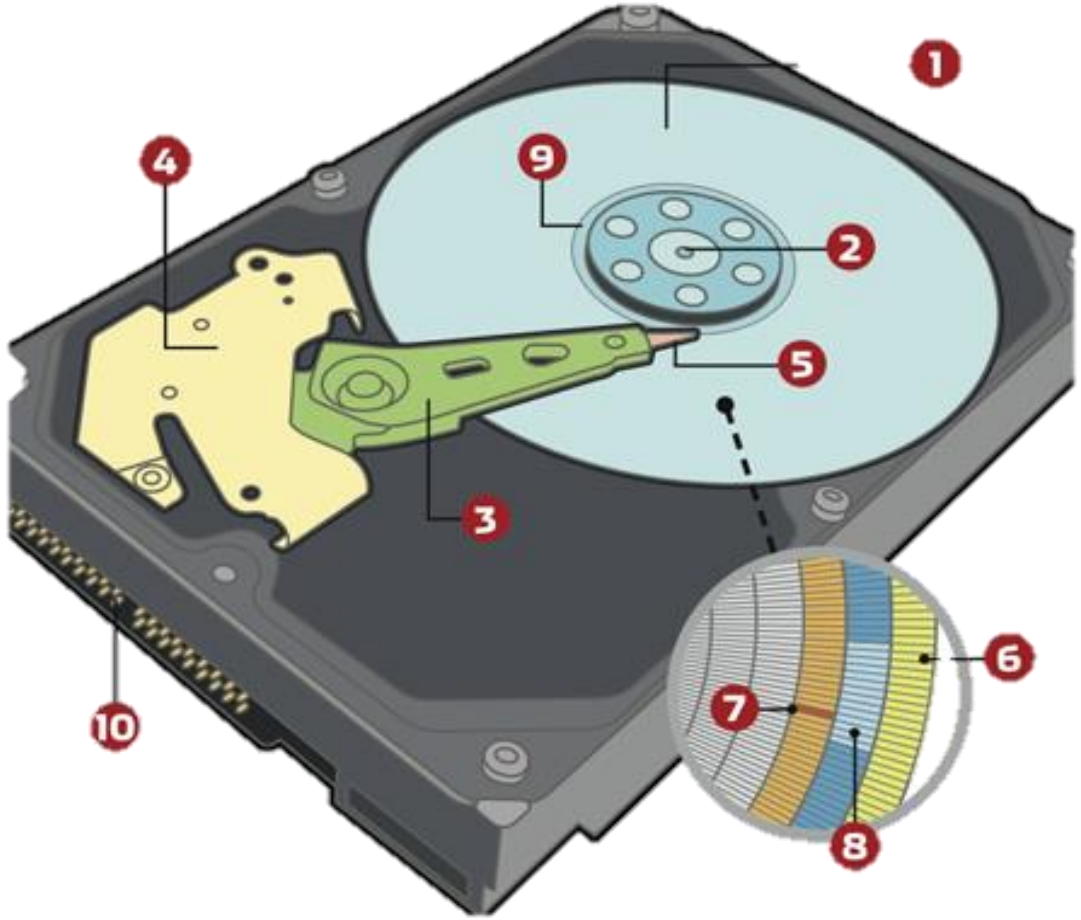
Bin	Oct	Dec	Hex	Sinal
0100 0000	100	64	40	@
0100 0001	101	65	41	A
0100 0010	102	66	42	B
0100 0011	103	67	43	C
0100 0100	104	68	44	D
0100 0101	105	69	45	E
0100 0110	106	70	46	F
0100 0111	107	71	47	G
0100 1000	110	72	48	H
0100 1001	111	73	49	I

Bin	Oct	Dec	Hex	Sinal
0110 0000	140	96	60	`
0110 0001	141	97	61	a
0110 0010	142	98	62	b
0110 0011	143	99	63	c
0110 0100	144	100	64	d
0110 0101	145	101	65	e
0110 0110	146	102	66	f
0110 0111	147	103	67	g
0110 1000	150	104	68	h
0110 1001	151	105	69	i

UNICODE



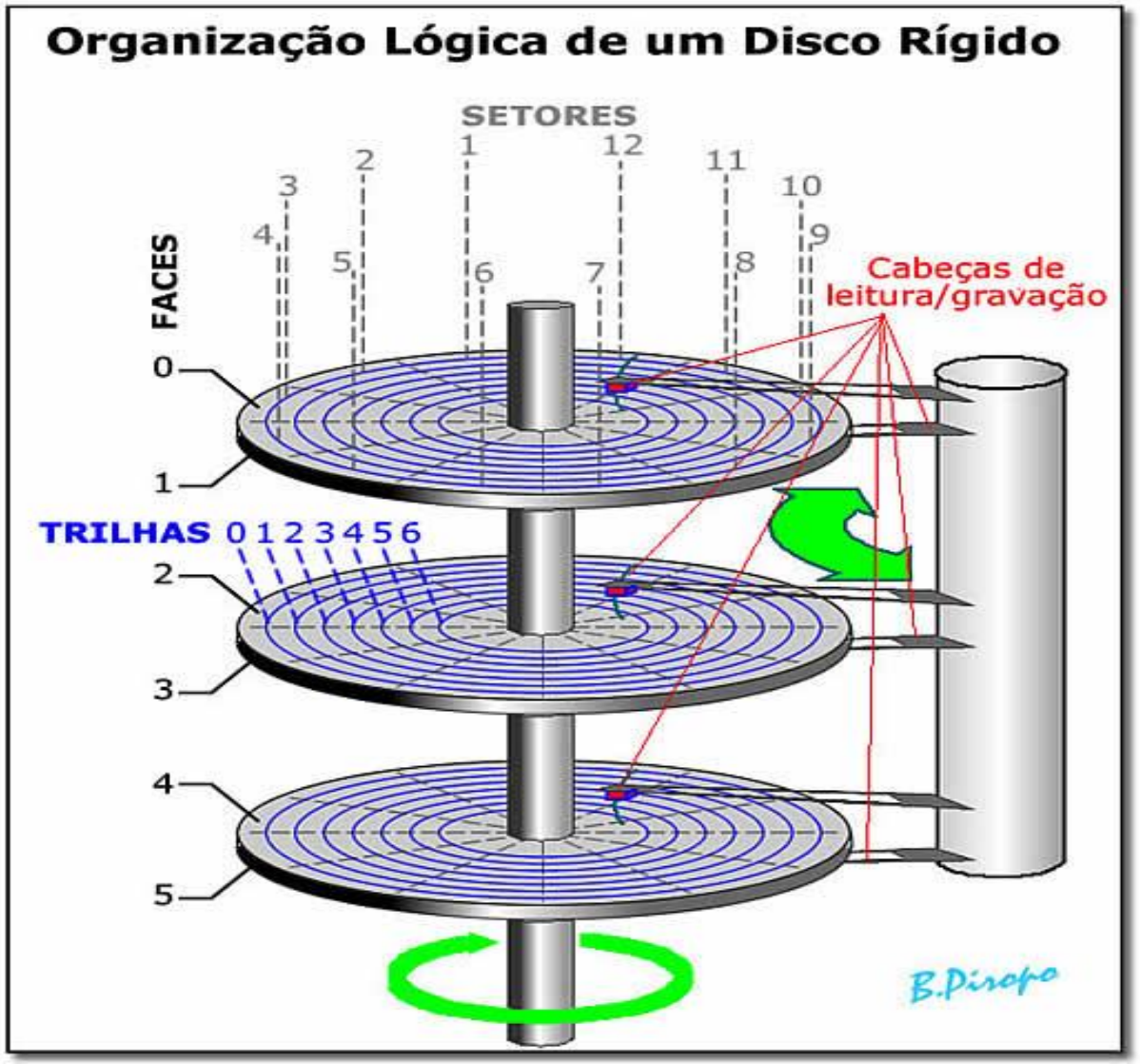
Discos



Sistema de Arquivos

- Ao trabalharmos com HDs (e outros discos) é necessário prepará-los, fazendo uma formatação física. Este processo, divide os discos em trilhas (uma espécie de caminho circular) e setores (subdivisões de cada trilha, com geralmente 512 bytes). Um conjunto de trilhas recebe o nome de cilindro.

Formatação baixo nível



- Trilhas
- Setores
- Cilindros
- Clusters

- checksum

Setores

Gap section: O “gap” separa os setores.

Sync section: A marca de sincronização indica o inicio do setor e fornece alinhamento temporização.

Address Mark section: dados para identificar o número e localização do setor status sobre o próprio setor.

Data section: dados do usuário.

ECC section: códigos de correção de erros.

Sistemas de Arquivos

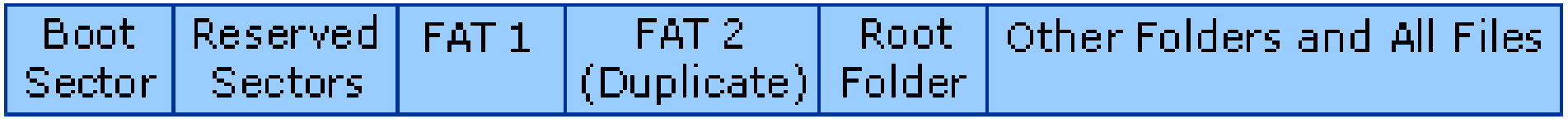
FAT

•Criada em 1977 por Bill Gates e Marc McDonald para o BASIC da Microsoft

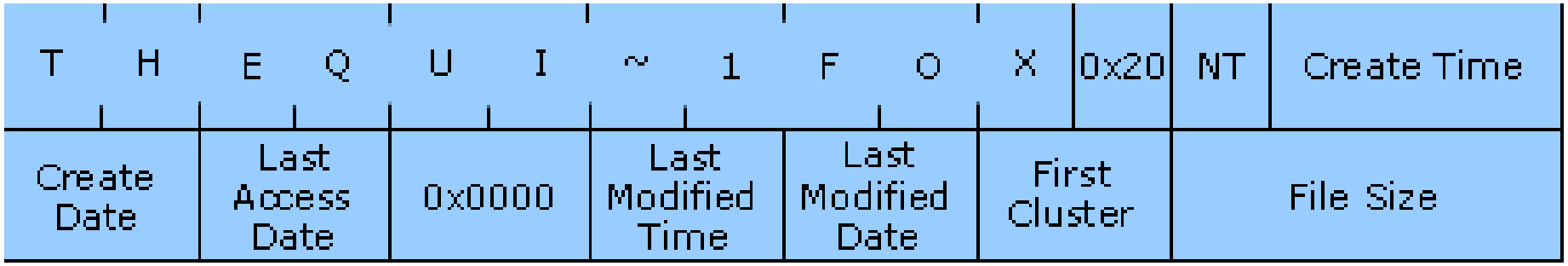
- Simples, foi utilizada como principal sistema de arquivos até a total adoção do NTFS no WinXP (2001).
- Utilizada ainda em dispositivos menores, como disquetes, cartões de memória, etc.

Estrutura

- Início do disco:



- FAT:



Vantagens e desvantagens

- Simples, rápida e eficiente para dispositivos com pequena capacidade

- Não possui segurança, pode ocorrer fragmentação dos arquivos, não possui funcionalidades adicionais e não oferece suporte a dispositivos de alta capacidade.

Variantes

- Existiram diversas variantes da FAT, cuja diferença primordial é a quantidade de bits utilizados para endereçar clusters
  - FAT12
  - FAT16
  - FAT32 (VFAT para o win95)

Windows NT FileSystem (NTFS)

- Introduzido em 1993 (NT 3.1)
- Assim como o NT, criada do zero, sem preocupações com compatibilidade.
- **Introduz METADADOS aos arquivos**
  - Cada arquivo é composto por **atributos**.

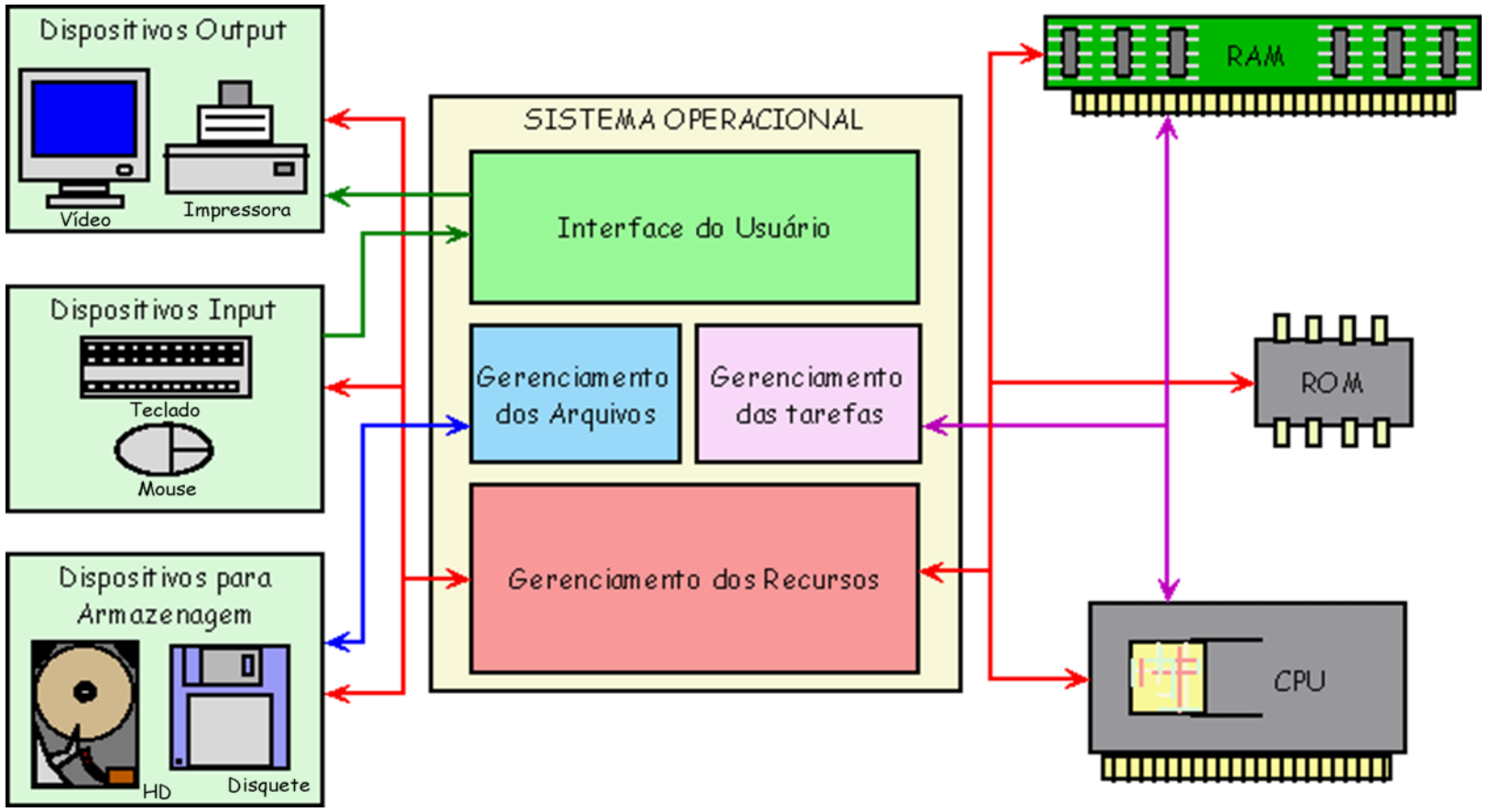
MFT

- Master File Table:
  - Uma entrada por arquivo
  - Cada entrada contém os meta dados do arquivo
  - As primeiras 16 entradas são reservadas
    - MFT, Clusters danificados, livres, etc.
    - Trata os próprios meta dados do NTFS como arquivos, que podem estar em qualquer lugar.

Armazenando

- Cada arquivo possui uma entrada na MFT (1kb)
- Os atributos que couberem, são colocados na MFT. Caso contrário, são colocados em clusters, e o ponteiro para este cluster é armazenado.
- Os clusters de um arquivo são organizados como uma árvore B+.

Sistema Operacional



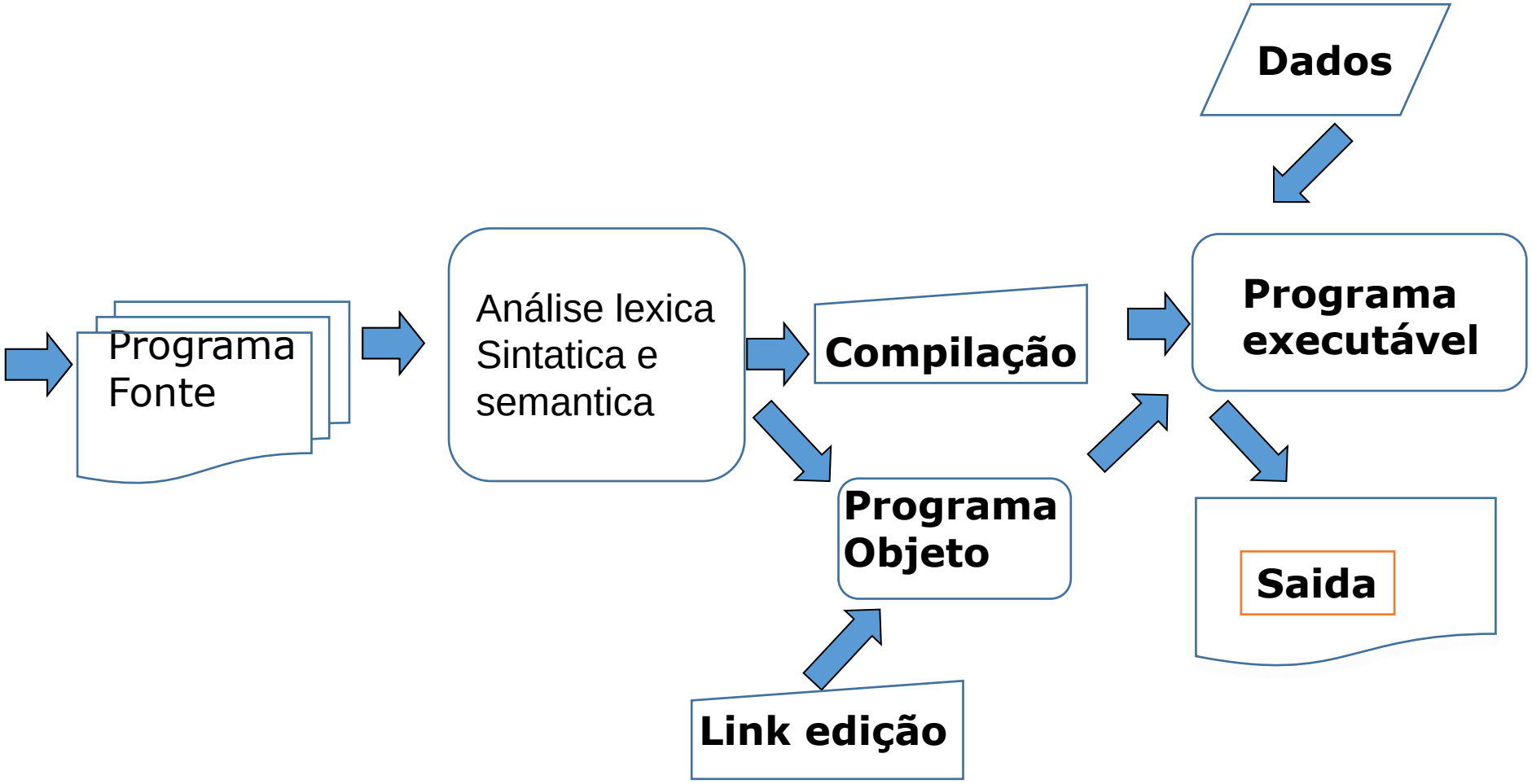
Exemplos de Sistemas Operacionais: Windows, Linux, DOS

POST power on self test

Programas Tradutores

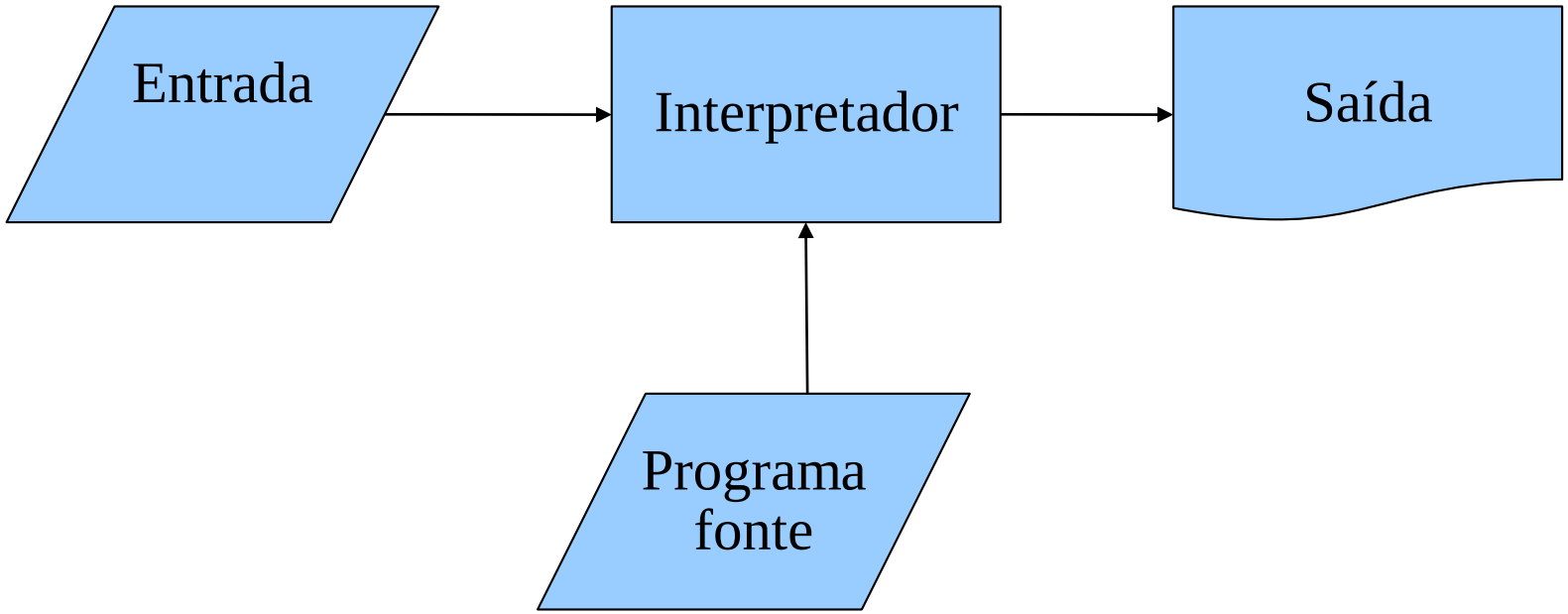
- Compiladores :verificam e decodificam TODAS as instruções do programa FONTE gerando um código executável;
- Interpretadores: verificam, decodifica e executa instrução a instrução.
- Existem outras formas usadas para traduzir do programa fonte para código executável.
- Sistemas Hibridos: combinam aspectos das duas tecnicas anteriores

Compilador


Dados
Programa
Fonte
Análise lexica
Sintatica e
semantica
Compilação
Programa
executável
Programa
Objeto
Saida
Link edição

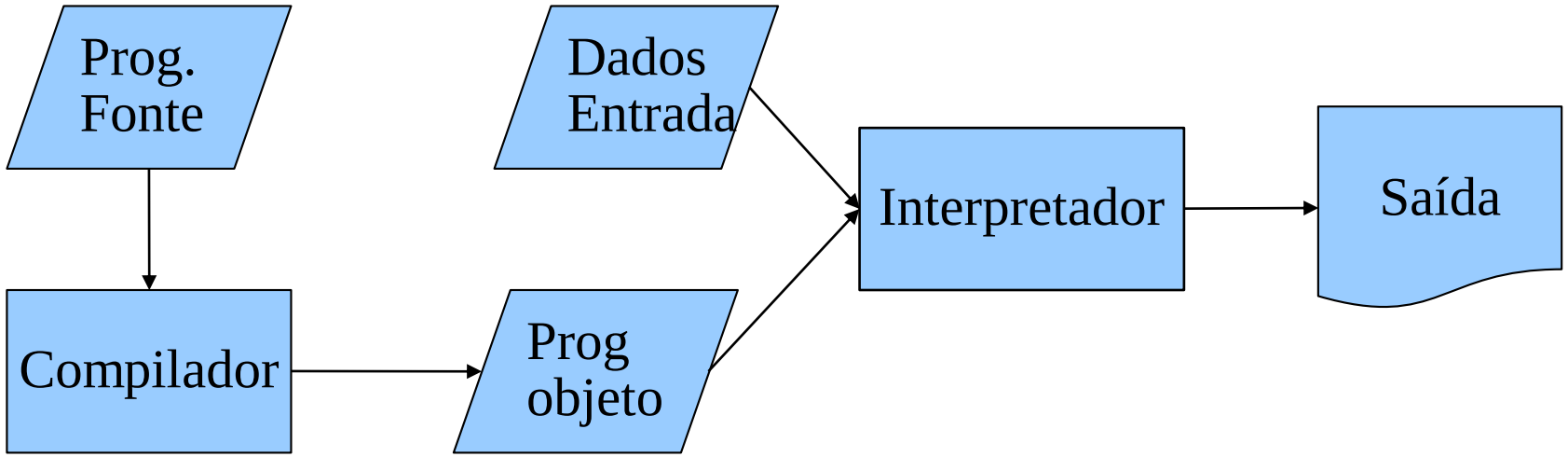
Interpretador

- Simulam uma “máquina virtual”
- Programa fonte é lido, verificado, decodificado e as instruções são executadas imediatamente



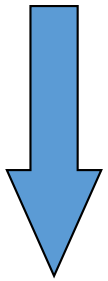
Sistemas Hibridos

- Compilador gera código para uma máquina virtual (pseudo-código)
- Máquina virtual é executada separadamente lendo pseudo-código e interpretando-o



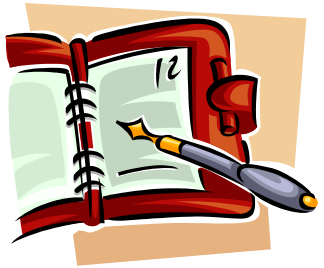
Um programa

- Armazena na memória uma sequencia de instruções



- Executa estas instruções sequencialmente da primeira até a última.
- As instruções são padronizadas

Modelo de von Neumann



Resultados

1. Lê as instruções de forma seqüencial e as executa uma a uma;
2. Possui um elenco de instruções que é capaz de entender
3. Por exemplo ler, escrever e calcular