

Table of Contents

Arquitetura de Computadores	2
Capitulos	3
Evolução dos Computadores 	4
Sistema Numérico	9
Componentes básicos de um PC	15
Tabela Verdade	17
 Portas Lógicas	22
Tipos de Memória	27
Organização do Processador 	31
Barramentos	36
Representação de Dados	40
Dispositivos de E/S	42

Arquitetura de Computadores

Sumario:

[\[\[learn-computing/arquiteture-computer/Capitulos/00 - Evolution of Computers.md\]\]](#)
[\[\[learn-computing/arquiteture-computer/Capitulos/01 - Numerical System.md\]\]](#)
[\[\[learn-computing/arquiteture-computer/Capitulos/02 - Components of a Basic PC.md\]\]](#) [\[\[learn-computing/arquiteture-computer/Capitulos/03 - Truth Table.md\]\]](#)
[\[\[learn-computing/arquiteture-computer/Capitulos/04 - Logic Ports.md\]\]](#) [\[\[learn-computing/arquiteture-computer/Capitulos/05 - Types of Memory.md\]\]](#) [\[\[learn-computing/arquiteture-computer/Capitulos/06 - Organization of the Processor.md\]\]](#)
[\[\[learn-computing/arquiteture-computer/Capitulos/07 - Barrament.md\]\]](#) [\[\[learn-computing/arquiteture-computer/Capitulos/08 - Data Representation.md\]\]](#) [\[\[learn-computing/arquiteture-computer/Capitulos/09 - IO Devices.md\]\]](#)


Capitulos

[[learn-computing/arquitecture-computer/Capitulos/00 - Evolution of Computers.md]]
[[learn-computing/arquitecture-computer/Capitulos/01 - Numerical System.md]]
[[learn-computing/arquitecture-computer/Capitulos/02 - Components of a Basic PC.md]] [[learn-computing/arquitecture-computer/Capitulos/03 - Truth Table.md]]
[[learn-computing/arquitecture-computer/Capitulos/04 - Logic Ports.md]] [[learn-computing/arquitecture-computer/Capitulos/05 - Types of Memory.md]] [[learn-computing/arquitecture-computer/Capitulos/06 - Organization of the Processor.md]]
[[learn-computing/arquitecture-computer/Capitulos/07 - Barrament.md]] [[learn-computing/arquitecture-computer/Capitulos/08 - Data Representation.md]] [[learn-computing/arquitecture-computer/Capitulos/09 - IO Devices.md]]

Evolução dos Computadores

Elementos Computacionais


Temos que primeiro definir o que é um computador. Ele é uma máquina que recebe determinadas entradas e, com base em seus algoritmos, produz determinadas saídas.

 Algoritmos são **sequências de instruções lógicas e finitas** 

Os elementos são:


Softwares

- Parte lógica do computador, são os algoritmos

 Aquilo que você xinga

Hardware

- Parte física da máquina, aquela em que é usada para transmitir, guardar e ser o esqueleto do computador.

 Aquilo que você chuta

Elementos de Hardware

Eles são classificados em 3 tipos básicos e essenciais:

```
Processador -> hardware do Computador <- Memória
                /\
                |
                Entrada/Saída
```


Os computadores se dividem em dois tipos: 🚀

Eles são divididos com base na sua forma de funcionamento: analógico e digital, onde:

Analógicos:

É o tipo de computador em que ele trabalha de maneira que não usa números e sim outras formas, como unicamente dois valores de uma corrente de energia.

Digital:

Aquele que usa um sistema de dígitos, ou seja, usam números (sendo de uma base binária: 0 e 1). 

```
[ Analógico ] => Medem  
[ Digital   ] => Calcula
```

Gerações de Computadores 🕒

Os computadores podem ser divididos em gerações por causa da tecnologia que eles usam.

Geração Zero

Geração base, foi até a Segunda Guerra (1932 - 1945), os computadores eram:

- Essencialmente **mecânicos e alguns tinham engrenagens eletromecânicas**.
- Exemplos: **Máquina de Pascal**

![[Pasted image 20240908191302.png]]



Assista esse vídeo sobre como funcionava a máquina:

https://www.youtube.com/watch?app=desktop&v=CJ7o-ir4R_E

Primeira Geração

Geração que apareceu impulsionada pela segunda grande guerra (1945 - 1955)

- Essencialmente usaram as **válvulas**, usando-as ao invés dos **relés** (um componente mecânico que era mais lento que as válvulas).
- Exemplos: ENIAC

![[Pasted image 20240908191120.png]]



💡 **Veja mais sobre o Eniac:** https://youtu.be/6X2B8Z_DCo0?si=1rCeBGrN48Yal49_

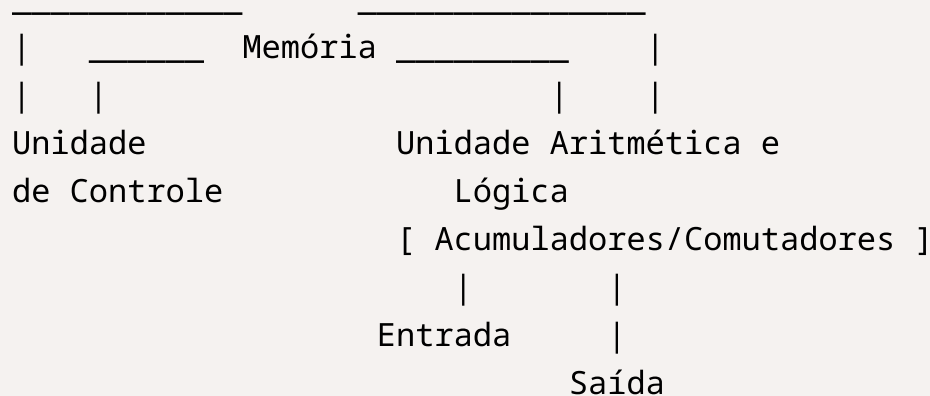
- Algumas **desvantagens** de seu uso são:
 - Que eram enormes, ou seja, ocupavam muito espaço.
 - Gastavam muita energia.
 - Pouca confiabilidade.

Esses computadores para entrada/saída e armazenagem de dados usavam os **cartões perfurados**.

Modelo de Von Neumann

Foi um matemático que contribuiu para a criação da forma como se faz a arquitetura de computadores. O modelo que ele inventou foi usado como base para os modelos posteriores e que são usados hoje.

Modelo:





Veja mais em:

- <https://youtu.be/YCe0mX3r9y4?si=1MMYctW2MN8ZgqVx>
- https://youtu.be/tZ5W2LpdcEw?si=3jCp89cc3G_sq62c
- <https://youtu.be/MI3-kVYLNr8?si=gRpWp494K1TnuYDu>

Segunda Geração

Também conhecida como **Geração dos Transistores**, logo, deixaram de usar as válvulas e começou o processo de miniaturização dos componentes eletrônicos.

Assim, temos que os transistores trouxeram:

- Menor tamanho aos computadores.
- Consumo de energia caiu.
- Menos aquecimento.
- Tornou-se mais confiável que as válvulas.

![[Pasted image 20240908193336.png]]



Confira em: Manchester's Experimental Transistor Computer, the First Computer to Use Mainly Transistors as Switches

(<https://www.historyofinformation.com/detail.php?entryid=4074>)



Veja mais em: From Transistors To Tetris Part 1 : Computer Architecture

(<https://youtu.be/6caLyckwo7U?si=g8uCA5xnOTVmaQWD>)

Terceira Geração

Conhecida pelo advento dos **Circuitos Integrados** (CI), assim esses circuitos contêm vários e vários transistores que estão miniaturizados em um único componente, então

acabou se tornando mais acessível para os públicos além dos militares, empresas e universidades.

![[Pasted image 20240908194231.png]]



Fonte:

<https://computerscience.chemeketa.edu/cs160Reader/HistoryOfComputers/Ger>



Veja mais:

- https://youtu.be/ti9VVBHljWU?si=zy_5nhGTJ_9yYNdw
- <https://youtu.be/IGHbRfawoCo?si=jylmioc60MxukN3c>
- <https://youtu.be/m0nza32BRI8?si=4aSyB6tCLYT1LVC->

Quarta Geração (Atual)

A miniaturização dos chips para uma larga escala e integração dos mesmos. Nasce então o **LSI** (Large Scale Integration), visando aumentar o poder de processamento dos processadores.

Que por sua vez, nessa geração, tomou o nome de **microprocessadores**, criado inicialmente pela INTEL. Eles vieram como uma solução poderosa que continha todos os componentes que um computador precisava de um processador: **unidade central de processamento, controladores de memória e de E/S**.

![[Pasted image 20240908194448.png]]



Fonte: <https://chauman4.weebly.com/fourth-generation-computers.html>



Nota: Um ponto importante para levantar é que dependendo do material ou do escritor podemos ter mais de 4 gerações, o que não está errado, mas iremos nos abster nessas quatro que são as principais.

Sistema Numérico

Sistema de Numeração e Conversões Entre Bases

1. Introdução

Os sistemas de numeração são ferramentas essenciais para representar quantidades e realizar cálculos matemáticos. No dia a dia, utilizamos predominantemente o sistema decimal, com base em 10 símbolos (0 a 9). No entanto, diversos outros sistemas de numeração foram desenvolvidos ao longo da história, cada um com suas características e aplicações específicas.

Neste artigo, exploraremos os principais sistemas de numeração utilizados: binário, octal e hexadecimal, além de abordar as técnicas de conversão entre eles.

2. Sistema Decimal

O sistema decimal é o mais utilizado no mundo, com base em 10 símbolos (0 a 9). Cada posição à direita do ponto decimal representa uma potência de 10. Por exemplo, no número 123,456:

- 1 na posição das centenas de milhar (10^5)
- 2 na posição das dezenas de milhar (10^4)
- 3 na posição das unidades de milhar (10^3)
- 4 na posição das centenas (10^2)
- 5 na posição das dezenas (10^1)
- 6 na posição das unidades (10^0)

3. Sistema Binário

O sistema binário utiliza apenas dois símbolos: 0 e 1. É amplamente utilizado na computação digital, pois representa diretamente os estados de um circuito eletrônico (ligado/desligado). Cada posição à direita do ponto binário representa uma potência de 2. Por exemplo, no número 101101:

- 1 na posição das unidades (2^0)
- 0 na posição das dezenas (2^1)
- 1 na posição das centenas (2^2)
- 1 na posição das unidades de milhar (2^3)
- 0 na posição das dezenas de milhar (2^4)
- 1 na posição das centenas de milhar (2^5)

4. Sistema Octal

O sistema octal utiliza 8 símbolos: 0 a 7. É menos comum que o sistema binário, mas ainda encontra aplicações em algumas áreas, como em permissões de arquivos em sistemas Unix. Cada posição à direita do ponto octal representa uma potência de 8. Por exemplo, no número 12345:

- 1 na posição das unidades (8^0)
- 3 na posição das dezenas (8^1)
- 4 na posição das centenas (8^2)
- 5 na posição das unidades de milhar (8^3)
- 2 na posição das dezenas de milhar (8^4)

5. Sistema Hexadecimal

O sistema hexadecimal utiliza 16 símbolos: 0 a 9 e A a F. É comumente utilizado em programação e eletrônica, pois permite representar grandes quantidades de dados de forma mais concisa. Cada posição à direita do ponto hexadecimal representa uma potência de 16. Por exemplo, no número 1F9A:

- A na posição das unidades (16^0)
- 9 na posição das dezenas (16^1)
- F na posição das centenas (16^2)

- 1 na posição das unidades de milhar (16^3)

6. Conversões Entre Bases

A conversão entre sistemas de numeração é essencial para trabalhar com diferentes representações de dados. Diversas técnicas podem ser utilizadas, como:

- **Conversão manual:** envolve cálculos passo a passo, utilizando as definições de cada sistema.
- **Calculadoras online:** ferramentas online podem realizar conversões entre bases de forma rápida e precisa.
- **Funções de conversão em linguagens de programação:** linguagens como Python possuem bibliotecas que facilitam a conversão entre bases.

7. Tabela de Conversão Resumida

Decimal	Binário	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7

Continuando a seção sobre conversões entre sistemas de numeração, apresentamos algumas técnicas para converter manualmente entre decimal, binário, octal e hexadecimal:

A. Decimal para Binário:

1. Divida o número decimal por 2.
2. Anote o resto da divisão (0 ou 1) como o bit menos significativo do número binário.
3. Divida o quociente da divisão anterior por 2 e repita os passos 1 e 2.
4. Continue dividindo por 2 e anotando os restos até o quociente se tornar 0.
5. Leia os restos da divisão na ordem inversa, do último para o primeiro. Essa é a representação binária do número decimal.

Exemplo: Converter 13 (decimal) para binário.

1. $13 / 2 = 6$ (resto 1)
2. $6 / 2 = 3$ (resto 0)
3. $3 / 2 = 1$ (resto 1)
4. $1 / 2 = 0$ (resto 1)

Lendo os restos na ordem inversa: 1101 (binário).

B. Binário para Decimal:

1. Cada bit na representação binária tem um peso equivalente a uma potência de 2 (começando em 2^0 para o bit menos significativo).
2. Multiplique cada bit pelo seu peso correspondente.
3. Some os resultados das multiplicações.

Exemplo: Converter 1011 (binário) para decimal.

1. $1 * 2^3 = 8$

2. $0 * 2^2 = 0$

3. $1 * 2^1 = 2$

4. $1 * 2^0 = 1$

Soma: $8 + 0 + 2 + 1 = 11$ (decimal).

C. Decimal para Octal:

1. Divida o número decimal por 8.
2. Anote o resto da divisão como o dígito menos significativo do número octal.
3. Divida o quociente da divisão anterior por 8 e repita os passos 1 e 2.
4. Continue dividindo por 8 e anotando os restos até o quociente se tornar 0.
5. Leia os restos da divisão na ordem inversa, do último para o primeiro. Essa é a representação octal do número decimal.

D. Octal para Decimal:

1. Cada dígito na representação octal tem um peso equivalente a uma potência de 8 (começando em 8^0 para o dígito menos significativo).
2. Multiplique cada dígito pelo seu peso correspondente.
3. Some os resultados das multiplicações.

E. Decimal para Hexadecimal:

1. Divida o número decimal por 16.
2. Anote o resto da divisão como o dígito menos significativo do número hexadecimal (0 a 9, A a F).
 - Se o resto for maior que 9, use a letra correspondente (A para 10, B para 11, etc.).
3. Divida o quociente da divisão anterior por 16 e repita os passos 1 e 2.
4. Continue dividindo por 16 e anotando os restos até o quociente se tornar 0.

5. Leia os restos da divisão na ordem inversa, do último para o primeiro. Essa é a representação hexadecimal do número decimal.

F. Hexadecimal para Decimal:

1. Cada dígito na representação hexadecimal tem um peso equivalente a uma potência de 16 (começando em 16^0 para o dígito menos significativo).
2. Multiplique cada dígito pelo seu peso correspondente (A = 10, B = 11, etc.).
3. Some os resultados das multiplicações.

Estas são apenas algumas técnicas básicas de conversão. Existem outras abordagens e tabelas de conversão que podem ser utilizadas para facilitar o processo.

Componentes básicos de um PC

Nesse caso iremos abordar por um lado do hardware, ou seja, parte física.

Componentes:

- São eles:

CPU | Memória | Dispositivos de Entrada e Saida | Barramentos

- Esses componentes são conectados por **Barramentos** (são 'estradas' ou cabos que ligam os componentes)

```
-----  
CPU   | Memória | E/S |  
-----  
|     |         |    |  
|     |         |    |  
-----  
|  Barramentos  |  
-----
```

CPU

CPU é a **Unidade de Processamento Central** ele executa, processa os dados e atividades a serem feitas

Memória

Memória é a responsável de **guardar as informações** do computador sendo as informações voláteis ou não voláteis

⚠ Memória **voláteis** é aquela que precisa armazenar informações até o momento em que o PC está ligado. Agora memórias **não voláteis** guardam as informações mesmo desligando o computador.

- Existem dois tipos de classificação de memórias em relação a sua ordem. Sendo elas:
 - **Ordem Primaria** = são memórias responsáveis por **guardar os programas** a serem executados e outras informações importantes, focadas em serem acessadas
 - Exemplo: RAM, Cache
 - **Ordem Secundaria** = são memórias que **armazenam os dados permanentemente**, focadas em apenas armazenar
 - Exemplo: HD, SSD

E/S

São dispositivos responsáveis pela interação **humano** e **computador**. servindo para entrar e sair com dados.

- **Exemplos**
 - Entrada: WEBCAM, Teclado, Mouse
 - Saída: Monitor, Caixa de Som

Barramento

são fios ou caminhos condutores que passam pelo computador e interligam os componentes.

Tabela Verdade

O que seria Tabela Verdade?

Tabela Verdade a forma como podemos obter valores de **expressões booleanas** de todas as possíveis combinações.

⚠ Expressões Booleanas são aquelas que possuem apenas dois valores lógicos: verdadeiro (true) ou falso (false).

Ajudando a construção de circuitos e na verificação de preposições lógicas.

Lógica Proposicional

É a linguagem que usamos para manipular **proposições**.

Proposição

São afirmações que podem ser vvaloradas em dois únicos valores: **verdadeiro** ou **falso**. Pode-se tipar as proposição de dois tipos: **simples** e **compostas**; Elas possuem uma regra de reepresentação, que é: devem usar letras minúsculas

⚠ Normalmente se usa: **p, q, r**, etc

- Exemplo: p = Sou o Douglas; q = Você não é o Douglas

Simples

É quando existe uma única proposição.

- Exemplo: Está chovendo (verdadeiro ou Falso)

Compostas

É quando se tem mais de uma proposição, ligadas por **conectivos lógicos**.

- Exemplo: Está chovendo **ou** está nublado (com o uso do **ou** se liga duas proposições e assim se estabelece determinada regra do conectivo que está sendo usado)

Conectivos Lógicos

São partículas usadas para **combinar** e **modificar** proposições e assim criar **expressões booleanas**

Alguns Conectivos

- **Negação** (\neg ou \sim): ele nega a sentença, logo o que é verdadeiro fica falso e o que é falso se torna verdadeiro.
 - Exemplo: p = Gustavo é antinteração; $\sim p$ = Gustavo não é antinteração

p	q	$\sim p$	$\sim q$
V	F	F	V
F	V	V	F
V	V	F	F
F	F	V	V

- **Conjuntivo** (\wedge): esse conectivo diz que para ser verdadeiro as duas devem ser verdadeiras
 - Exemplo: p = verdadeiro; q = verdadeiro; $p \wedge q$ = Verdadeiro

p	q	$p \wedge q$
v	f	f
f	v	f
v	v	v
f	f	f

- **Disjunção (v):** é o conectivo em que apenas uma precisa ser verdadeira para que a expressão resulte em verdadeiro.
 - Exemplo: p = verdadeiro; q = falso; $p \vee q$ = verdadeiro

p	q	$p \vee q$
v	f	v
f	v	v
v	v	v
f	f	f

- **Condicional (se...então) (\rightarrow):** é o conectivo em que para ser verdadeiro, a condição p deve ser verdadeira para que a proposição q seja verdadeira.
 - Exemplo: p = verdadeiro; q = verdadeiro; $p \rightarrow q$ = verdadeiro

p	q	$p \rightarrow q$
v	f	f
f	v	v
v	v	v
f	f	v

- **Bicondicional (se somente se) (\leftrightarrow):** é o conectivo em que a condição estabelecida equivale para as duas proposições, ou seja, será verdadeiro se as duas forem atendidas.
 - Exemplo: p = Vou gastar; q = tiver dinheiro; $p \leftrightarrow q$ = verdadeiro = Vou gasta, se somente se, tiver dinheiro

p	q	$p \leftrightarrow q$
v	f	f
f	v	f
v	v	v
f	f	v

Tabela de Conectivos

Conectivo	Valor
\sim	Será verdadeiro se for falso e falso se for verdadeiro
\wedge	Será verdadeiro se ambas forem verdadeiras
\vee	Se uma ou as duas forem verdadeiras então resulta em verdadeiro
\rightarrow	Se a primeira for verdadeira e a segunda for falsa então resulta em falso, senão é verdadeiro
\leftrightarrow	Só será verdadeiro se as duas tiverem o mesmo valor, senão é falso



Portas Lógicas

Os computadores são constituídos de elementos eletrônicos como: **capacitores**, **resistores** e **transistores**.

- **Capacitores:** são os componentes responsáveis por armazenar e liberar carga elétrica, sendo que realizam a filtragem de ruído e a estabilização de tensões.
- **Resistores:** limitam a passagem de corrente elétrica, são usados para controlar a voltagem da corrente elétrica.
- **Transistores:** amplificam sinais e controlam o fluxo da corrente dentro do circuito. Eles permitem ou não a passagem de binários para realizarem operações através das portas lógicas, e tudo isso forma os circuitos lógicos.

Portas Lógicas são a base para a construção de um processador. Elas são embutidas em um CI (Circuito Integrado) com o objetivo de realizar tarefas específicas. Podem ser encontradas tanto em **ULSI (Ultra Larga Escala Integrada)** quanto em circuitos mais simples.

Álgebra de Comutação

Assim como a álgebra básica da escola, criou-se a necessidade de fazer operações com os dígitos binários. Surge então a **Álgebra de Comutação**.

Para tal, era necessário primeiro definir-se as representações gráficas e então se adotou 0 (falso) e 1 (verdadeiro).

Porta AND

Esta porta aceita dois operandos: A e B, sendo binários 0 e 1. A operação AND simula a multiplicação binária, possuindo a finalidade de garantir que o mesmo bit de entrada seja o mesmo da saída (transferência de bit, ou seja, é usado para transferir dados da memória para a CPU).

- Porta Lógica **AND**:

Entrada	Saída	
A	B	$Y = AB$
1	0	0
0	1	0
1	1	1
0	0	0

Porta OR

Esta porta aceita dois operandos: A e B, sendo binários 0 e 1. Ela simula a soma binária, ou seja, só resultará em verdadeiro (1) se um dos operandos for igual a 1.

- Porta Lógica OR:

Entrada	Saída	
A	B	$Y = A+B$
1	0	1
0	1	1
1	1	1
0	0	0

Porta XOR (exclusive or)

Esta porta aceita dois operandos: A e B, sendo binários 0 e 1. Ela serve como uma verificação de igualdade, em que se os operandos tiverem os seus valores binários diferentes, a operação resultará em verdadeiro (1). Caso contrário, resultará em falso (0).

- Porta Lógica **XOR**:

Entrada	Saída	
A	B	$Y = A \text{ XOR } B$
1	1	0
0	1	1
1	0	1
0	0	0

Porta NOT

Esta porta aceita um operando: A, sendo binário 0 ou 1. Ela faz uma inversão de valores, ou seja, se o valor do operando for 1, ele se torna 0, e se for 0, ele se torna 1.

- Porta Lógica **NOT**:

Entrada	Saída
A	NOT A
1	0
0	1

Portas Derivadas



A execução dessas portas se dá por uma de cada vez, ou seja, irá executar primeiro uma e depois a outra.

Porta NAND

Esta porta aceita dois operandos: A e B, sendo binários 0 e 1. Ela faz a operação AND e em seguida realiza a execução do NOT.

- Porta Lógica **NAND**:

Entrada	Saída	
A	B	$Y = A \text{ NAND } B$
1	0	1
0	1	1
1	1	0
0	0	1

Porta NOR

Esta porta aceita dois operandos: A e B, sendo binários 0 e 1. Ela faz primeiro o OR e em seguida opera o NOT.

- Porta Lógica **NOR**:

Entrada		Saída
A	B	$Y = A + B$
1	0	0
0	1	0
1	1	0
0	0	1

Tipos de Memória

Primeiro devemos entender como é o ciclo das memórias, num computador:

```
-> ligar -> PC -> Corre na CPU -> Confere a ROM  
-> Vai na memória secundaria para buscar o SO  
-> usa a memória principal para carregar os programas
```

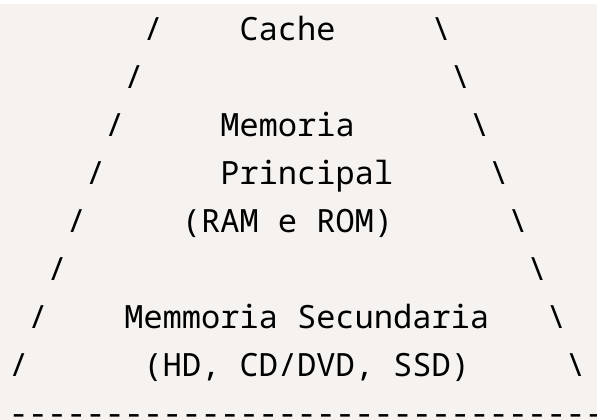
A **Memória** é um dispositivo que vai servir para armazenar dados ou instruções para que a CPU use. Possuindo dois tipos de memórias, **permanente** e **ão permanente**. As memórias são interligas e formam o **subistema de memórias**.

Pirâmide de Memórias

Pode-se classificar as memórias com base em algumas propriedades.

- Quanto mais alto:
 - Mais caro
 - Mais rapido
 - Baixa capacidade de armazenamento
- Quanto mais baixo:
 - Mais barato
 - Menos rapido
 - Alta capacidade de armazenamento





Registradores

O processador busca informações e instruções na memória e armazena em seu interior o que deve ser feito. Para tal esse local onde são armazenados é denominado como **registradores**.

Cache

O Cache serve para suprir a necessidade do processador de ter uma memória auxiliar. Servindo para acelerar a comunicação entre o **processador** e a **memória principal**.

Cache L1

Sendo a primeira a ser buscada, a memória cache que está dentro do processador, dividida em dois campos:

- **L1d**: feito para guardar dados
- **L1i**: feito para guardar instruções

Cache L2

Ao não encontrar o que se procurava na L1 é feita a busca em L2. E essa memória utiliza o sistema: Static Random Access Memory

Ao não se encontrar no Cache L2, ela é responsável por chamar a memória principal, ou seja, a DRAM (Dynamic Random Access Memory)

Memoria Principal

Responsável por alocar informações das aplicações executadas no momento em que o PC ainda está ligado. Por isso ela precisa ficar refrescando a memória, assim ela funciona de modo dinâmico. A memória principal é denominada memória RAM.

Padrões de RAM

- DDR: este padrão tem como característica a duplicação de lotes de bits por 1 ciclo de clock, ou seja, 64 bits por 1 ciclo de clock (ida e volta do CPU a Memória)
- Ela possui várias versões que mudam em voltagem, latência, etc...

Formato

- DIMM: usadas em desktops, são mais paradas, tudo em uma linha
- SO-DIMM: usadas em laptops, são mais acopladas e em várias linhas

ROM

É a memória que é apenas para leitura (Read Only Memory = ROM), escrita normalmente pelo fabricante e apenas lida depois. Os programas instalados nela são chamados de **firmwares**.

Firmwares já instalados:

- BIOS: sistema Básico de Entrada e Saída;
- POST: autoteste e verificação no momento da inicialização;
- SETUP: muda as configurações já pre-definidas;

Memoria Secundaria

É a memória que vai ser aquela que vai armazenar os dados em massa, ou seja, o **SO** e outros programas, assim como, outros dados e por isso ela não pode ser **vólátil**. Essas memórias **não** são **acessadas** pelo **processador**.



Volátil é igual a desligou o PC ela deixou de existir, igual a gigolos, acabou o dinheiro não se vê mais ninguém

SO é o Sistema Operacional, sendo a interface entre hardware e humano e software. ele é aquele amigo cúpido

Organização do Processador

A CPU é responsável pelo processamento e execução de programas que estão armazenados nela. Ela é dividida em três partes:

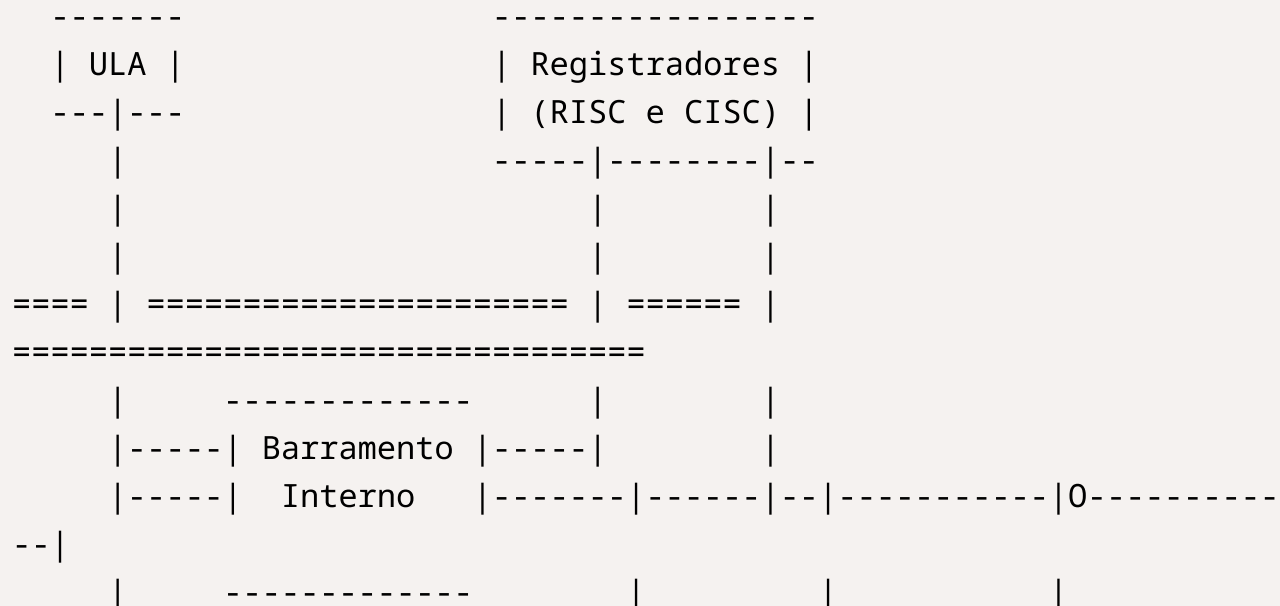
1. **Unidade Central (UC);**
2. **Registradores;**
3. **Unidade Lógica Aritmética (ULA ou UAL);**

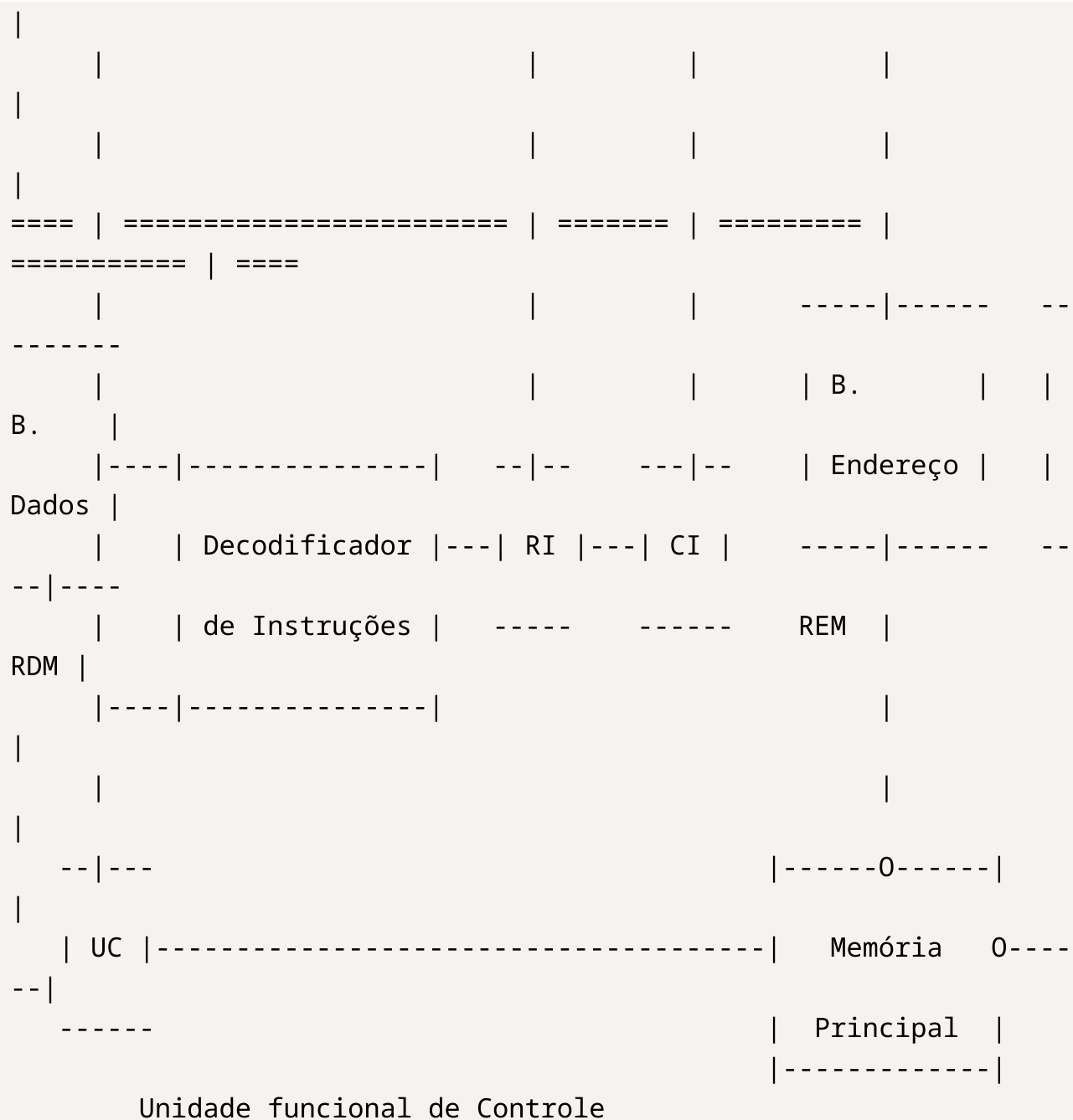
Podendo ser dividida em duas partes funcionais:

- **Unidade Funcional de Controle;**
 - UC
- **Unidade Funcional de Processamento;**
 - ULA e Registradores

Diagrama de Funcionamento da CPU

Unidade Funcional de Processamento





- REM => Registro de Endereços Memória
- RDM => Registro de Dados Memória

Unidade Funcional de Processamento

Todo sistema operacional possui uma única função de existência, ou seja, o porquê dele existir e a função para esses sistemas são: **entrar com dados, processar dados, saída de**

dados processados, assim nasce a Unidade Funcional de Processamento.

Logo a UFP, possui algumas operações básicas:

- Operações Aritméticas
- Álgebra Booleana
- Movimentação de Dados entre a CPU e a Memória

ULA

É a parte central da CPU já que é onde as operações lógicas e aritméticas irão ser feitas. Ela não recebe as instruções diretamente, e sim as instruções são processadas pela UC. Assim que processado, é enviado para a ULA que realiza o que se pede e retorna o resultado.

Registradores

É o tipo de memória que é rápida e com pouco armazenamento. Sendo que varia em sua função e quantidade de acordo com o modelo do processador. A maioria dos processadores utilizam a arquitetura baseada em registradores de processos gerais (RISC/CISC):

- **RISC (Reduced Instructions Set Computer):**
 - Caracterizado pela **simplicidade e eficiência** nas execuções de instruções (voltado mais para dispositivos que exigem menos processamento como dispositivos móveis e laptops);
- **CISC (Complex Instructions Set Computer):**
 - Caracterizado por um conjunto de instruções mais **complexas e abrangentes** (voltado para dispositivos que exigem mais poder de processamentos como desktops e servidores);

Unidade Funcional de Controle

Executa algumas funções:

- Busca de instruções a serem executadas e armazenadas em um registrador da CPU;
- Interpretar as instruções para serem enviadas à ULA
 - Gerar sinais de controle, ao interpretar vai gerar um sinal para a ULA dizendo qual das operações devem ser executadas;

Contador de Instruções (CI)

O Contador de Instruções é aquele que vai registrar a contagem para sequenciamento das instruções, ou seja, montar aquela fila de fichas, onde cada ficha possui um número de ordem de chamada para que se possa ter o controle das ordens de instruções.

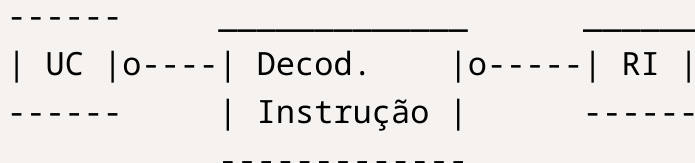
Registrador de Instruções (RI)

Este Registrador de Instruções possui a função de armazenar a instrução que deve ser executada pela CPU.

Decodificador de Instruções

O RI irá passar uma *sequência de bits* representando a instrução a ser executada para o **Decodificador de Instruções** que, por sua vez, irá interpretar essa sequência de bits e relacionar com a operação que deve ser feita. Em seguida, mandar essa instrução já interpretada para a **UC**, assim ela manda os sinais necessários para a ULA, por exemplo, do que deve ser feito.

- Diagrama de funcionamento RI e Decod. Instruc.

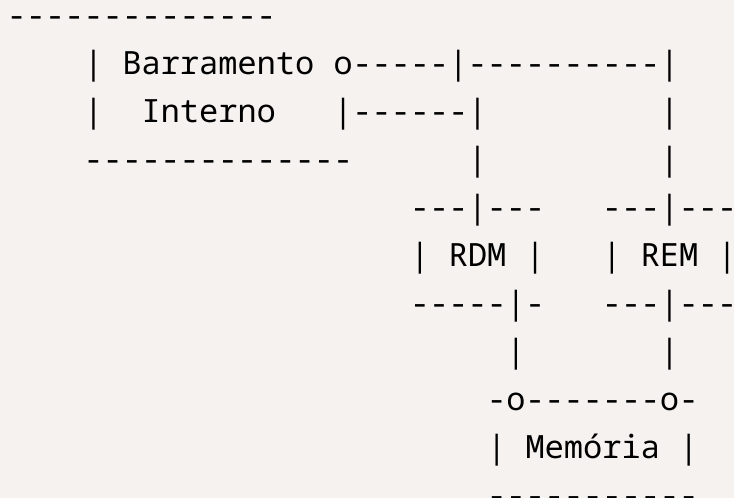


RDM e REM

- RDM (Registrador de Dados em Memória): sendo o registrador que *armazena os dados que estão sendo transmitidos* da CPU e para a Memória e vice-versa.

- REM (Registrador de Endereços de Memória): sua função é *armazenar o endereço de acesso à memória* para que seja necessária a leitura e a escrita de dados.

⚠ Ambos os registradores possuem registro temporário dos dados que são gravados neles.



o => significa o fluxo de direção dos dados

Barramentos

Barramentos são as **interconexões** dos componentes computacionais.

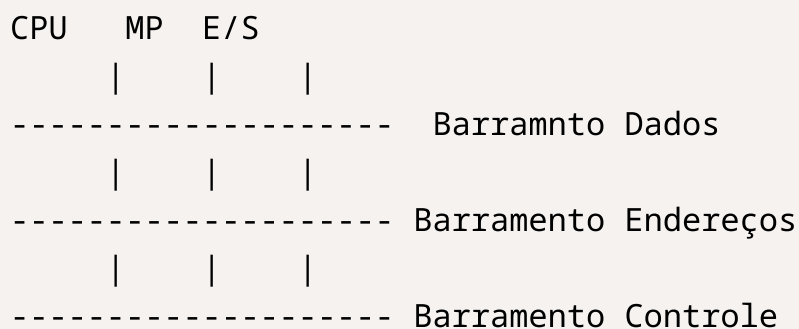
Tipos de Barramentos

São divididos em três categorias:

- Estrutura
 - Dados
 - Endereço
 - Controle
- Hierarquia
 - Local
 - Sistema
 - Expansão

Estrutura

É formado por três tipos seguintes de Barramentos:



Barramento de Dados

Esse barramento *interliga o RDM (Registrador de Dados em Memória) com a memória principal*, para a **transferência de instruções** ou **dados a serem executados**.

Sendo um barramento bidirecional: - Já que pode tanto ir da *CPU à Memória*, assim realizando uma **operação de escrita**; - E também consegue ir da *Memória à CPU*, assim realizando uma **operação de leitura**.

A **largura do barramento** está diretamente ligada ao **desempenho da máquina**, já que quanto mais envio de dados (bits) maior será a velocidade e poder de processamento, tal que, por consequência, o desempenho irá aumentar. Os primeiros PCs continham apenas 8 bits de largura do seu barramento, hoje temos algo em torno ou maior que 128 bits.

Barramento de Endereços

Feito para interligar o *REM (Registrador de Endereços de Memória)* com a *Memória Principal* que irá **fazer a transferência de bits que vão representar endereços de memória das instruções ou dados a serem executados**. Tem seu sentido **unidirecional**, já que somente a CPU aciona a memória principal para fazer operações de leitura e escrita.

Barramento de Controle

Irá *interligar a UC com os outros componentes*. Sendo **bidirecional**, pois consegue enviar sinais de controle para a Memória e receber dela sinais do tipo *wait* (espere), para que possa terminar a execução da tarefa para começar outra.

Assim, os **barramentos** compartilham os dados por vias que são físicas, por meio de fios de cobre, e conectam todos os componentes.

Hierarquia de Barramentos

A Hierarquia está ligada à **velocidade** de tráfego desses barramentos.

Barramento Local ou Interno

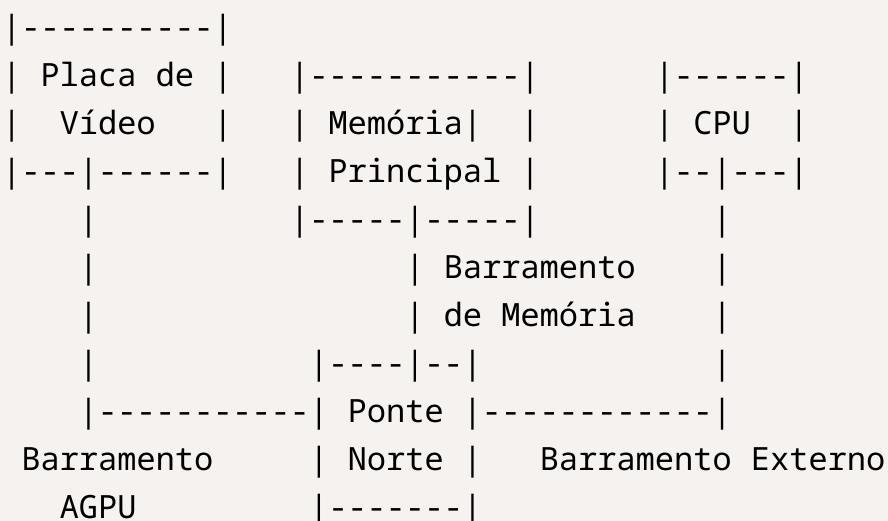
É o barramento mais rápido, já que está dentro da área da CPU e funcionando no mesmo tempo do relógio do processador, ou seja, está na mesma velocidade de clock do processador.

Barramento Sistema

Tem por finalidade fazer a conexão entre o barramento local com os outros componentes do sistema, como por exemplo: memória principal, cache L2, E/S. Ele faz uso do *chipset norte* da placa-mãe.

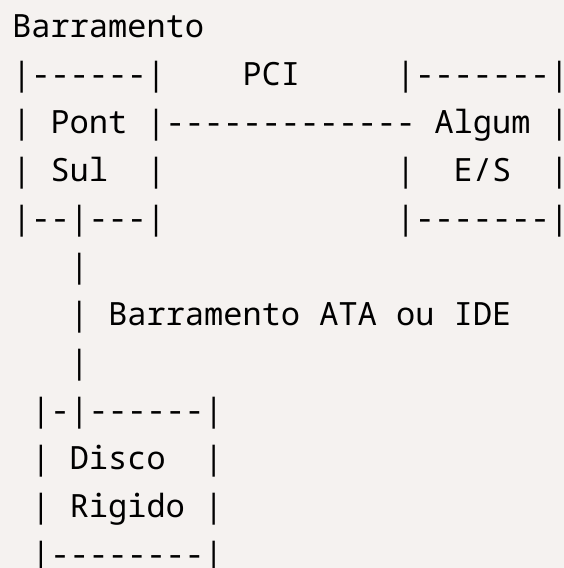
Um Circuito Integrado (**chipset**) cuida de fazer a integração desse barramento deste



modo:



Barramento de Expansão (ou E/S)

Sendo o barramento que **interliga os dispositivos de E/S com os outros componentes** do computador. Tal integração ocorre por meio do chipset (circuito integrado) chamado de: **ponte sul**.



  Em Discos Rígidos os dois tipos **ATA** e **IDE** podem ser integrados irá variar do tipo de memória escolhida.

- **ATA** 

- IDE ![[Pasted image 20240910105001.png]]

Slots

Os slots nada mais são que as entradas que os barramentos possuem (as boquinhass que têm na placa-mãe para conectar os fios). Os slots variam de acordo com o modelo da placa-mãe.

![[Pasted image 20240910105028.png]]

Representação de Dados

Um computador realiza uma execução sistemática de instruções sobre valores, esses valores são nomeados genericamente de: **dados**. Tais valores são códigos binários (0 e 1).

```
processa
  |
  |
instruções
  |
  |---valores
        |___ 0101
        |___ 11100
```

Processo de Conversão

Para conversarmos com esses dados e o computador conversar conosco, usa-se um processo conversão de linguagens: linguagem do computador para a natural. Sendo que:

- Alto nível: temos as linguagens que os programadores usam para conversar com o PC
 - Linguagem natural (não se usa binário e sim símbolos que se assemelham a linguagem humana)
 - Alto nível de abstração
- Baixo Nível:
 - Conversa diretamente com o computador, linguagem binária (0 e 1)
 - Baixo nível de abstração

Formas de Representação

Tais formas variavam de acordo com as escolhas de quem desenvolve e com a plataforma que usa-se. Podendo haver variações da forma como se define o tipo de dado a ser usado:

- Fortemente Tipada: deve ser definido os tipos de dados na hora do desenvolvimento
- Fracamente Tipada: deixa essa responsabilidade de tipagem para o compilador da linguagem

Tipos de Dados

Existem alguns tipos dados:

- integer: eles são numeros inteiros, representados de formas variadas dependendo da linguagem, podendo ter: 8bits, 16bits, 32bits e 64bits.
- real: são numeros decimais e podem ser subdivididos em dois grupos:
 - float: menor precisão.
 - double: maior precisão.
- caractere (char): usado para representar apenas um caractere, sendo normalmente usado aspas simples para criar o char
- cadeia de caracteres (string): são mais de um caractere entre aspas (sendo simples ou duplas)
- booleano: valores lógicos que podem ser: true (verdadeiro ou 1) ou false (falso ou 0)

Representação da Memória

Pode-se dizer que a memória funciona como um vetor, ela possui índices e neles tem seus valores. Só que diferente de um vetor a memória permite acesso direto e aleatório dos índices e não necessita de percorrer sequencialmente.



O sistema de gerenciamento de memória é o programa que realiza a alocação de memória e a coleta de lixo, logo a memória RAM possui alocações e liberações dinâmicas.

Dispositivos de E/S

Os dispositivos de entrada e saída são os responsáveis por fazer essa integração da máquina com o humano, para que se possa tanto inserir dados quanto receber esses dados processados. Possuindo as seguintes funções:

- Receber e enviar informações para o meio.
- Converter as informações recebidas e enviadas para possuírem um formato inteligível para o computador e para o usuário.

Exemplos: teclado, mouse, HD, SSD, etc.

Classificação dos Dispositivos

Entrada

Eles fornecem os dados, ou seja, um teclado é um dispositivo de entrada.

Saída

Eles exibem os dados, ou seja, um monitor é um dispositivo de saída.

Entrada e Saída

Eles realizam tanto a entrada quanto a saída de dados, uma impressora se enquadra nessa categoria, já que ela recebe o que se quer imprimir e gera a saída do material impresso.

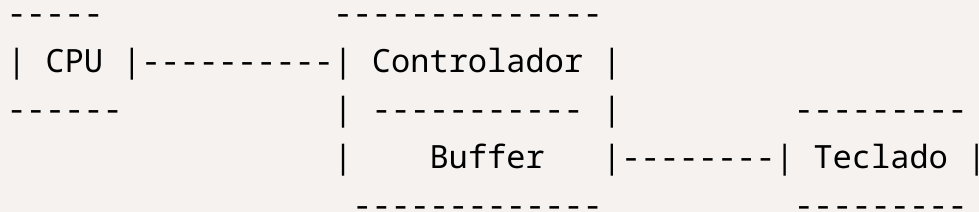
Comunicação

Os dispositivos de E/S precisam se comunicar com a CPU, e para isso existem algumas tecnologias de comunicação.

Serial

- Envio de dados bit a bit.

Gabinete



- O controlador também pode ser chamado de **driver**.
- Buffer: é a área que vai armazenar temporariamente e fazer a sincronização de velocidade.

USB

Uma das formas de fazer a comunicação de dispositivos é o Universal Serial Bus (USB), possuindo alguns tipos, como:

- Micro USB: mais lento e comum em carregadores.
- USB-A: usado de forma muito comum para conectar laptops, desktops, etc.
- USB-C: rápido, carrega e pode-se usar os dois lados para carregar, por exemplo.
- USB-B: usado em impressoras.

Bluetooth

É uma forma de conexão feita por ondas de rádio, simples, sem fio e rápida.

Wi-Fi

Conexão feita por sinais sem fio.

Gerenciamento dos Dispositivos

Existem algumas formas de gerenciamento:

- Polling: antiga forma de gerenciamento, em que a CPU fica constantemente consultando os dispositivos de E/S para verificar se estão prontos ou precisando de atenção.

- Interrupção: a CPU fica ouvindo um dispositivo e, quando este terminar sua tarefa, ela será interrompida para processar a solicitação.
- DMA (Acesso Direto à Memória): permite o acesso direto à memória principal para transferir dados, sem a intervenção da CPU em cada processo.