Módulo 02 · Fundamentos

Codificação de Dados & Modelos Computacionais

Portal de Computação — Curso GTI

Gestão da Tecnologia da Informação — Fatec Assis

Agenda do Módulo 02 — 50 minutos

Bloco	Tempo	Tópico	Prática / Simulador
1	10 min	Codificação de Texto: ASCII vs Unicode	Tabela ASCII 0–127
2	15 min	Estrutura de Máscara de Bits UTF-8	UTF-8 (1 a 4 Bytes) & Emojis
3	15 min	Máquina de Turing & Tese de Church-Turing	Simulador da Máquina de Turing
4	10 min	Arquitetura de Von Neumann & Paradigmas	Memory Wall & Ternário Balanceado

Bloco 1 · Codificação de Texto

ASCII vs Unicode

Como o Computador Armazena Texto?

O computador não armazena letras — armazena **números binários**. É necessária uma convenção para mapear cada caractere a um número.

Tabela ASCII (1963)

- Define caracteres de **0 a 127** (7 bits).
- Cobre o alfabeto inglês, números $0-9$, pontuações e caracteres de controle (`NUL`, `BS`, `LF`, `CR`).

```
'A' = 65  = 0x41 = 01000001
'a' = 97  = 0x61 = 01100001
Diferença entre 'A' e 'a' = 32 (exatamente o bit 5 alterado!)
```

⚠️ **Limite do ASCII:** Não possui acentos (`ç`, `ã`, `é`), alfabetos não-latinos ou emojis.

Bloco 2 · O Padrão Unicode & UTF-8

Codificação Universal de Tamanho Variável

Estrutura da Máscara de Bits em UTF-8

```
1 BYTE  (ASCII):      [ 0x x x x x x x ] → Usado para letras 'A', 'B', '1'
2 BYTES (Acentos):    [ 1 1 0 x x x x x ] [ 1 0 x x x x x x ] → 'ç', 'ã'
3 BYTES (Símbolos):   [ 1 1 1 0 x x x x ] [ 1 0 x x x x x x ] [ 1 0 x x x x x x ] → '∑', 'π'
4 BYTES (Emojis):     [ 1 1 1 1 0 x x x ] [ 1 0 x x x x x x ] [ 1 0 x x x x x x ] [ 1 0 x x x x x x ] → '😀'
```

Tamanho	Intervalo Code Point	Máscara de Bits UTF-8	Exemplo
1 Byte	`U+0000..U+007F`	`0xxxxxxx`	`'A'` (`0x41`)
2 Bytes	`U+0080..U+07FF`	`110xxxxx 10xxxxxx`	`'ç'` (`U+00E7` / `C3 A7`)
3 Bytes	`U+0800..U+FFFF`	`1110xxxx 10xxxxxx 10xxxxxx`	`'∑'` (`U+2211`)
4 Bytes	`U+10000..U+10FFFF`	`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	`'😀'` (`U+1F600`)

Bloco 3 · Modelos Computacionais

A Máquina de Turing (1936)

O Modelo Abstrato da Computabilidade

```
stateDiagram-v2
    [*] --> CARREGAR
    CARREGAR --> CARREGAR: lê 1 → escreve 0, move ←
    CARREGAR --> PARAR: lê 0 → escreve 1
    PARAR --> [*]
```

⚙ **Demonstração:** Teste os programas de *Incremento Binário* e *Inversor* no **Simulador da Máquina de Turing** do portal!

Bloco 4 · Arquitetura de Von Neumann

E Novos Paradigmas

Arquitetura de Von Neumann (1945)

Modelo adotado por todos os computadores modernos: **Dados e Programas na mesma memória**.

- **CPU:** Unidade de Controle (UC) + Unidade Lógica/Aritmética (ALU) + Registradores.
- **Memória Principal:** Armazena o código executável e os dados em uso.
- **Ciclo de Instrução:** *Fetch* (Busca) $\rightarrow$ *Decode* (Decodifica) $\rightarrow$ *Execute* (Executa).

> ⚠️ **Gargalo de Von Neumann (*Memory Wall*):** A velocidade do processador supera a velocidade de transferência do barramento com a memória RAM.

Computação Ternária (Base 3)

- **Ternário Balanceado:** Utiliza 3 estados por posição (**Trits**): `-1`, `0` e `+1`.
- **Vantagem Histórica (Setun 1958):** Operações com números negativos sem necessidade de bit de sinal.
- **Aplicação Atual: Quantização de Redes Neurais** (pesos $-1, 0, +1$) para aceleração de LLMs em GPUs e SoCs!

Resumo do Módulo 02

1. **ASCII vs Unicode:** ASCII usa 7 bits; Unicode usa Code Points com codificação UTF-8 de 1 a 4 bytes.
2. **Máquina de Turing:** O modelo universal que define o que é computável.
3. **Von Neumann:** Unificação da memória para dados e programa; origem do gargalo de memória.
4. **Ternário:** Base 3 reutilizada na IA moderna para eficiência energética.

> 💡 **Próximo Módulo:** Módulo 03 — Algoritmos, Big-O, Ponteiros & Memória.