



FUNDAMENTOS DA COMPUTAÇÃO

PROF. JOSENALDE OLIVEIRA

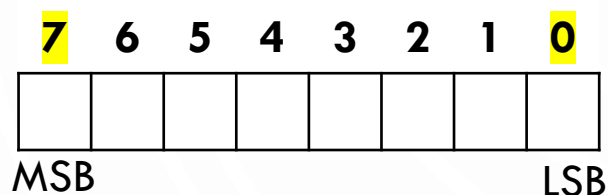
josenalde.oliveira@ufrn.br

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

MEDINDO DADOS EM BYTES: UNIDADES

- **bit (b)** – unidade fundamental: 0 ou 1

- **Byte (B)** – 8 bits



More Significant Bit

Less Significant Bit

- Os múltiplos do byte são escritos na base 2 (teóricos)

- kiloByte (kB): $1 \text{ kB} = 2^{10} \text{ B} = 1024 \text{ bytes}$
- megaByte (MB): $1 \text{ MB} = 2^{10} \text{ kB} = 1024 \text{ kbytes} = 2^{20} \text{ B} = 1.048.576 \text{ bytes}$
- gigaByte (GB): $1 \text{ GB} = 2^{10} \text{ MB} = 1024 \text{ MB} = 2^{20} \text{ kB} = 2^{30} \text{ bytes}$
- teraByte (TB): $1 \text{ TB} = 2^{10} \text{ GB}$

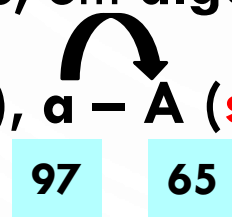
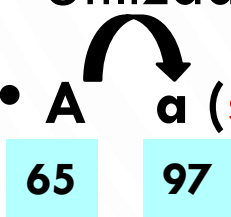
Na prática o Sistema Operacional, pode usar valores aproximados com base nos padrões de formatação e representação interna do sistema de arquivos. Por simplicidade, pode-se utilizar a base 10, sendo $1 \text{ KB} = 1000 \text{ B}$ e assim por diante.

18/06/2020 11:00	Aplicativo	477.450 KB
16/06/2020 21:25	Arquivo JPEG	110 KB
16/06/2020 21:18	Arquivo JPG	2.870 KB
16/06/2020 21:16	Arquivo JPEG	110 KB
16/06/2020 21:10	Arquivo JPEG	50 KB
16/06/2020 21:00	Arquivo JPEG	119 KB
15/06/2020 14:11	Documento do Mi...	40 KB
12/06/2020 16:14	Documento do Mi...	14 KB

Tipo de arquivo:	Arquivo JPEG (.jpeg)
Abre com:	Fotos
Local:	C:\Users\Josenalde\Downloads
Tamanho:	109 KB (112.181 bytes)
Tamanho em disco:	112 KB (114.688 bytes)

CÓDIGO ASCII: AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE

- Código de **8 bits** (primeira versão com 7 bits + 1 extra) : pode representar 256 símbolos diferentes (DEC de 0 a 255), com caracteres imprimíveis e não imprimíveis (`\a`, `\b`, `\r`, `\n`, `\t`, `\v`, `\f`) – chamados caracteres de controle
- A tabela apresenta a codificação BIN, DEC e HEX
- Utilizada, por exemplo, em **algoritmos de conversão** $a \leftrightarrow A \leftrightarrow a$

- $A \leftrightarrow a$ (**somar 32 DEC**), $a \leftrightarrow A$ (**subtrair 32 DEC**)


























65	01000001	41	A
66	01000010	42	B
67	01000011	43	C
68	01000100	44	D
69	01000101	45	E
70	01000110	46	F


97	01100001	61	a
98	01100010	62	b
99	01100011	63	c
100	01100100	64	d
101	01100101	65	e
102	01100110	66	f

UTF-8 (8-BIT UNICODE TRANSFORMATION FORMAT)

- Permite estender de 2 a 4 bytes para representar qualquer caracter UNICODE (qualquer símbolo): *unicode.org*



									
U+1F603	U+0C66	U+1F31E	U+8DD1	U+0254	U+FF03	U+1F929	U+2661	U+271E	U+27E8
									
U+30ED	U+3010	U+FF8D	U+2040	U+FE3F	U+1F618	U+30EA	U+FE43	U+00D2	U+1F512
		<p>Everyone in the world should be able to use their own language on phones and computers.</p> <p>LEARN MORE ABOUT UNICODE</p>							
U+266A	U+03EB								
									
U+300B	U+0E07								



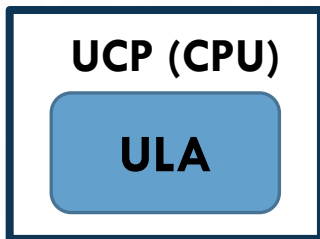
ADOPT A CHARACTER

```
char16_t u1 = u'\u0023';  
char32_t u2 = U'\U0001F603';
```

Javascript: emojis são combinações (16 + 16)
console.log('\uD83D\uDC36') 🐶

ARITMÉTICA BINÁRIA (MULT, DIV POR 2^N , N NATURAL)

- A **ULA (Unidade Lógico Aritmética)** emprega artifícios binários para realizar operações, por exemplo, multiplicações ou divisões sucessivas por 2. Exemplo, seja o número **binário $A = 0100$, que equivale a 4 DEC (inteiro)**. Se deslocarmos uma vez ($N=1$) para a direita (*Right shift*), teremos $A1 = 0010$, 2 DEC. Mais um deslocamento e teríamos $A2 = 0001$, 1 DEC.



4: 0 1 0 0 »
2: 0 0 1 0 »
1: 0 0 0 1

- Agora, seja $A = 0100$ e desloquemos para a esquerda (*Left shift*), teremos $A3 = 1000$, 8 DEC. Se tivermos mais um deslocamento, $A4 = 10000$, 16 DEC, e assim sucessivamente. Se o número for $A5 = 0011$, 3 DEC. Um *right shift* dá $A6 = 0001$, ou seja, a parte fracionária foi “perdida”, arredondada.

PARA SOMAR E SUBTRAIR

- Somam-se os bits normalmente usando as regras básicas:

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 10$

fica 0 e vai 1 para a coluna seguinte à esquerda (*carry-out* , *carry-in*)

	1	1	1	← Vai um						
	1	0	0	1	0	1		3	7	
+		1	1	1	1	0		+	3	0
	-----								-----	
	1	0	0	0	0	1	1		6	7

PARA SOMAR E SUBTRAIR

- Subtrai os bits normalmente usando as regras básicas:
 - $0 - 0 = 0$
 - $1 - 0 = 1$
 - $1 - 1 = 0$
- Para subtrair, normalmente utiliza-se a regra $A - B = A + (-B)$, reduzindo o problema a uma soma binária, com o segundo operando **NEGATIVADO** por **COMPLETO DE 2** (técnica mais comum)

COMPLEMENTO DE (PARA) 2

- É o sistema para codificação de valores positivos e negativos (inteiros) mais usado, utiliza um comprimento fixo de bits
- Significa quanto falta para 2^N , ou seja, bastaria subtrair este número de 2^N
- 2^N em binário é o bit 1 seguido de N zeros
 - $2^5 = 32 = 1\text{00000}$, $2^8 = 256 = 1\text{00000000}$
- Outro método é calcular o complemento de 1 e somar 1 ao bit LSB
- O MSB indica o sinal (0 – positivo), (1 – negativo)

EXEMPLO: -7 ; 7 em DEC (0111), invertendo, (1000) e somando 1 (1000 + 0001). Com quatro bits, se representa do -8 ao +7, porém não é possível representar o 8 POSITIVO

```
1 0 0 0
0 0 0 1
-----
1 0 0 1
```

Decimal	Binário s/ sinal	Binário (Compl. 2)
-8	-	1000
-7	-	1001
-6	-	1010
-5	-	1011
-4	-	1100
-3	-	1101
-2	-	1110
-1	-	1111
0	000	0000
1	001	0001
2	010	0010
3	011	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111

COMPLEMENTO DE (PARA) 2

Programador

Memória

0 -
-8 Não há nada salvo na memória

HEX FFFF FFFF FFFF FFF8

DEC -8

OCT 1 777 777 777 777 777 770

BIN 1111 1111 1111 1111 1111 1111 1111 1111
1111 1111 1111 1111 1111 1000

QWORD MS

0 -
-8

HEX F8

DEC -8

OCT 370

BIN 1111 1000

BYTE

TRANSMISSÃO E PARIDADE

Bit de paridade: segurança em transmissões assíncronas de baixa velocidade

Detetar ERRO em um bit

PAR: se o número de 1's for ímpar, adiciona 1, se for par, adiciona 0

IMPAR: se o número de 1's for par, adiciona 1, se for ímpar, adiciona 0

Dica: usar função lógica **OU-EXCLUSIVO** para paridade PAR e para paridade IMPAR, usar COINCIDÊNCIA, ou seja, NÃO (OU-EXCLUSIVO)

PARIDADE PAR

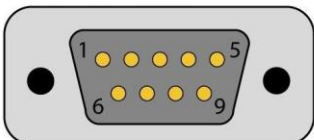
HANDSHAKING PARA ESTABELECEER DIÁLOGO INICIAL
INTENÇÃO DE TRANSMISSÃO – INFORMAR PARIDADE

CODIFICAR PAR (OU EXCLUSIVO)

DECODIFICAR PAR (OU EXCLUSIVO)



DB9M Connector

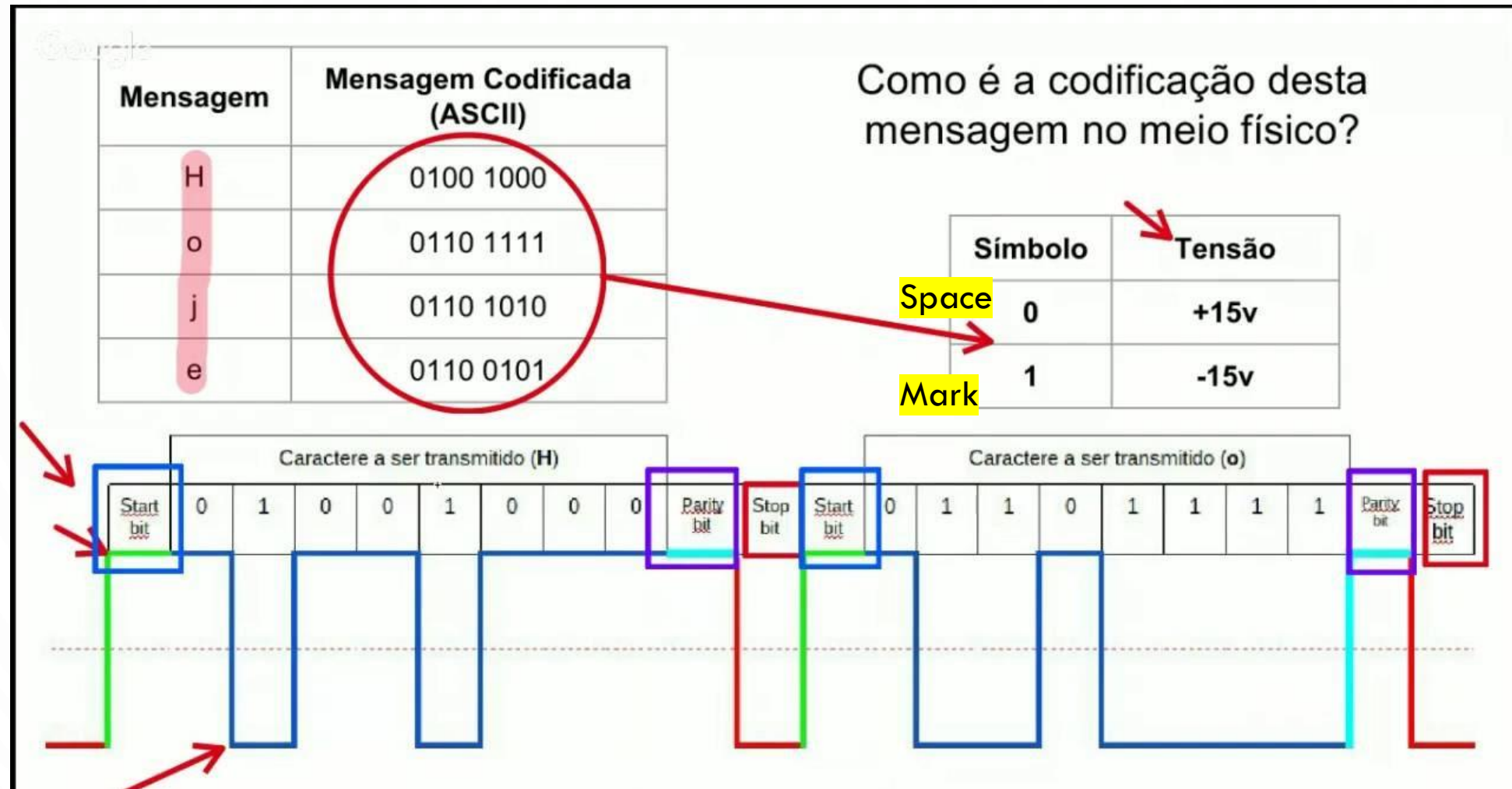


RS232 Pin Out

Pin #	Signal
1	DCD
2	RX
3	TX
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

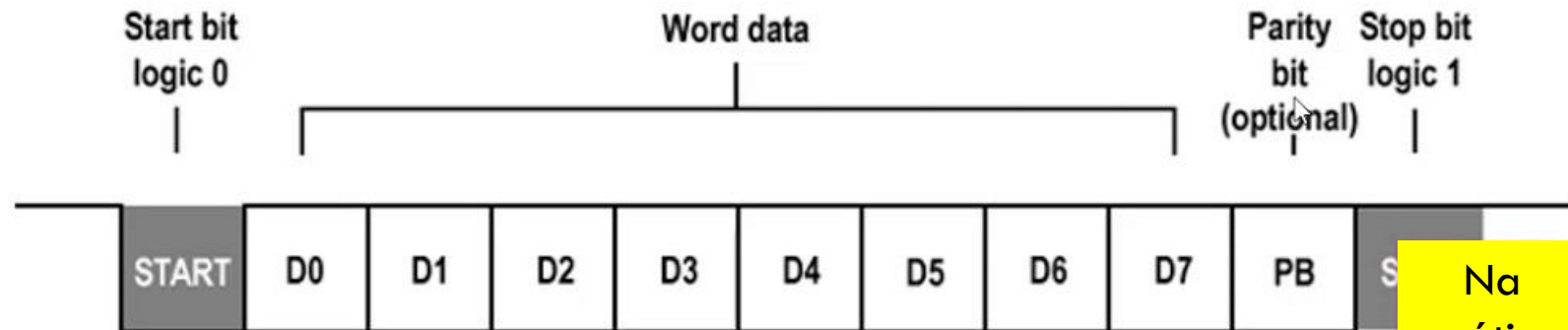


PARIDADE

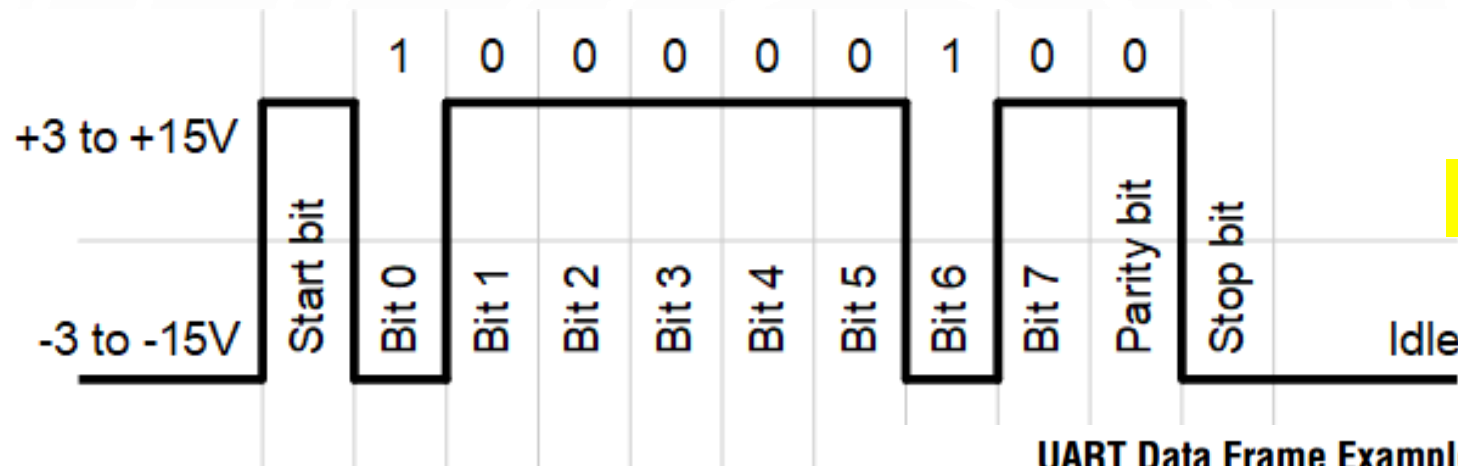


RS-232 Frame Format

- Each RS – 232 frame is consists of
 - 1 Start bit,
 - 8 Data bits, (LSB sent first and MSB sent last)
 - Parity,
 - 1 Stop bit.



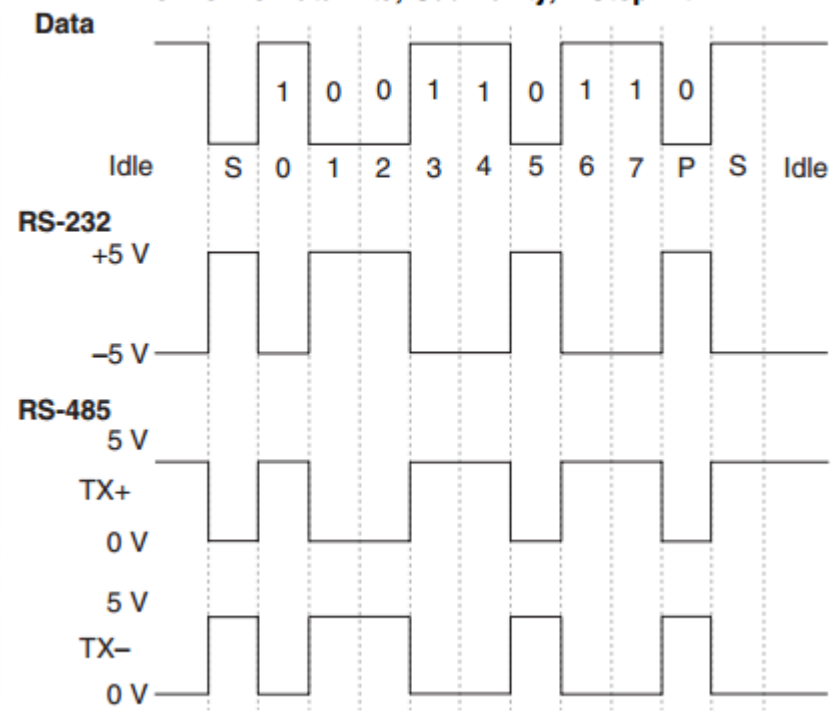
Na
prática



Sem transmissão (idle) envia 1

UART Data Frame Example

0xD9—8 Data Bits, Odd Parity, 1 Stop Bit



Voltages are for illustration only. Actual voltage levels may vary.

PARIDADE PAR

Dado: 00011001 – como o número de 1s é ímpar,
Acrescenta o bit de paridade 1 à esquerda do MSB para
tornar o conjunto PAR

Logo: 100011001



Para obter o bit de paridade, realiza XOR
entre bits do dado, dois a dois

Ex: $(((((0 \oplus 0) \oplus 0) \oplus 1) \oplus 1) \oplus 0) \oplus 0) \oplus 1 = 1$

No receptor, é feito o XOR de todo o conjunto,
incluindo o bit de paridade. Se o resultado for 0,
o dado chegou com sucesso. Se for 1, há um erro em
algum bit e solicita ao TX retransmissão. Este algoritmo
não detecta qual bit está errado!

PARIDADE ÍMPAR

CODIFICAR ÍMPAR (COINCIDÊNCIA = NOT(XOR))

TX



MEIO



RX

DECODIFICAR ÍMPAR (NOT(XOR))

Dado: 00011001 – como o número de 1s é ímpar,
Acrescenta o bit de paridade 0 à esquerda do MSB para
tornar o conjunto ÍMPAR
Logo: 000011001

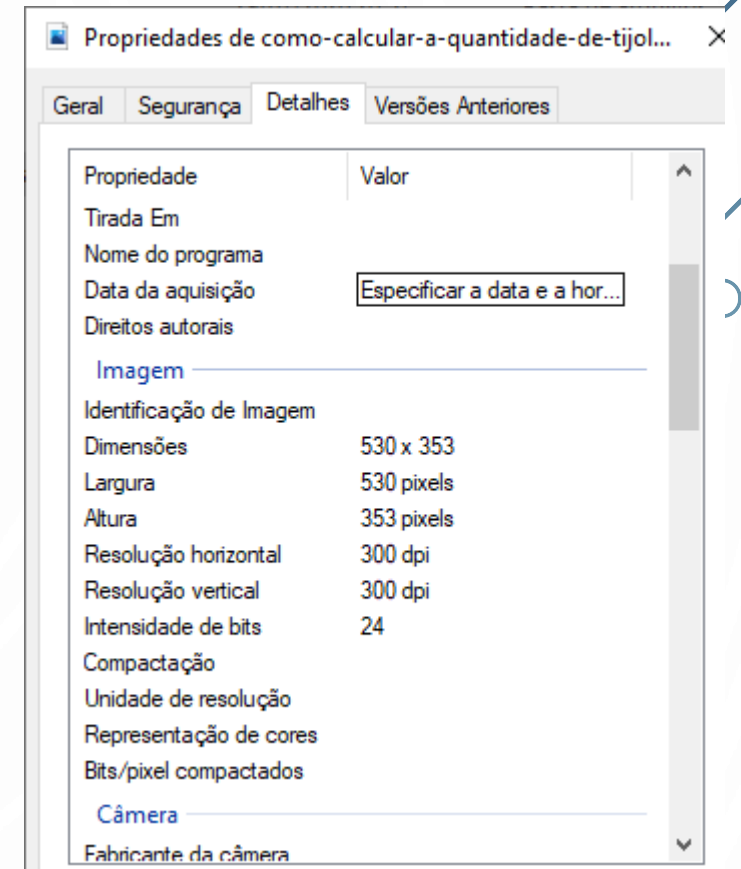
```
#include <bitset> // std::bitset
using namespace std;
#include <iostream>
int main() {
    bitset<9> d = 0b00001101;
    //gerar bit de paridade
    //supondo possível contar 1's
    //método count() retorna número de 1s
    //método set() marca bit com 1 ou 0 na posição
    //método size() retorna o tamanho do dado
    if (d.count() % 2 == 0) d.set(d.size()-1,0);
    else d.set(d.size()-1,1);
    cout << d << endl;

    //agora, como é gerado eletronicamente

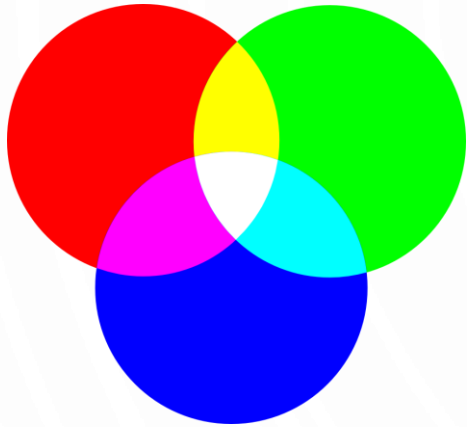
    bool bp;
    bp = (((((((d[0]^d[1])^d[2])^d[3])^d[4])^d[5])^d[6])^d[7]));
    d.set(d.size()-1,bp);
    cout << d << endl;
    return 0;
}
```

OUTRAS CODIFICAÇÕES

- **Imagens:** uma imagem em seu formato *raster* é vista como uma matriz de pontos, ou seja, a qualidade depende do número de pontos. Já uma imagem **VETORIAL** é descrita geometricamente, portanto, pode ser escalada (aumentada ou reduzida) sem perdas
- **Formato Bitmap (bmp):** diretamente proporcional ao tamanho e profundidade de cor.
 - Exemplo: imagem de 800 x 600 com profundidade (**intensidade**) de cor de 24 bits (RGB True Color)
1,44 MBytes (aprox.) sem compressão



NOÇÃO SOBRE RGB



Uma imagem colorida de 24 bits, significa que cada canal de cor possui 8 bits, pois $3 \times 8 = 24$. Assim, existem 2^{24} possibilidades ou combinações de cores, 16.777.216 cores possíveis

Sistema **RED** – **GREEN** – **BLUE**

A figura ao lado tem 400 x 400 pixels, ou seja, 160.000 pixels. Cada pixel tem 03 componentes, R, G, B, onde cada intensidade varia de **0 a 255**, para profundidade de 24 bits.

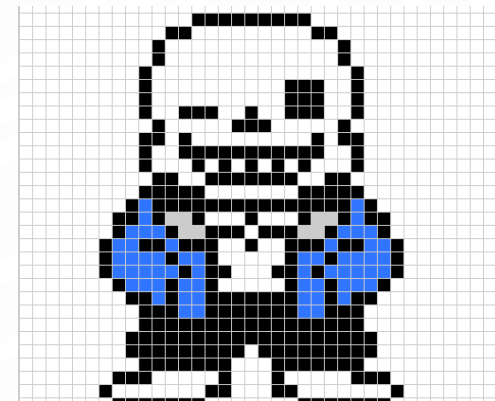
400 linhas (pixels)



400 colunas (pixels)



Exemplo de MATRIZ 8x8



OUTRAS CODIFICAÇÕES

- Graphics Interchange Format (GIF): algoritmo de compressão LZW
 - Reduz tamanho com qualidade semelhante (“lossless”)
 - Contudo máximo de 256 cores (8 bits)
 - Bom se não grandes variações de cores ou tons
 - Patenteado (Unisys)
 - **Permite animações**
- PNG (Portable Network Graphics): ideia de substituir GIF, 1996
 - Alta compressão sem limite de profundidade de cor
 - Sem proteção de patente
 - Permite também retirar o fundo das imagens
 - Animações com formato “companheiro” MNG e APNG

OUTRAS CODIFICAÇÕES

- Joint Photographic Experts Group (JPEG)
 - Reduz tamanho com perda de qualidade
 - Permite escolher taxa de compressão
 - Até 16 milhões de cores (24 bits)
 - Ideal para cenas com detalhes sutis
 - Permite animações
- Outros
 - RAW: usado em câmeras, arquivo “cru”, sem processamento ou filtragem pela câmera
 - CDR: (corel draw, vetorial)
 - PSD: photoshop, DWG (autocad), TIFF (impressoras industriais, câmeras)
 - SVG: vetorial para internet, BPG: ideia substituir JPEG

OUTRAS CODIFICAÇÕES - VÍDEO

- Sequência de quadros (frames) por unid. de tempo

- Cinema (24 fps – frames per second)
- Computador: 30 fps
- Exemplo: suponha cada imagem (quadro) de um

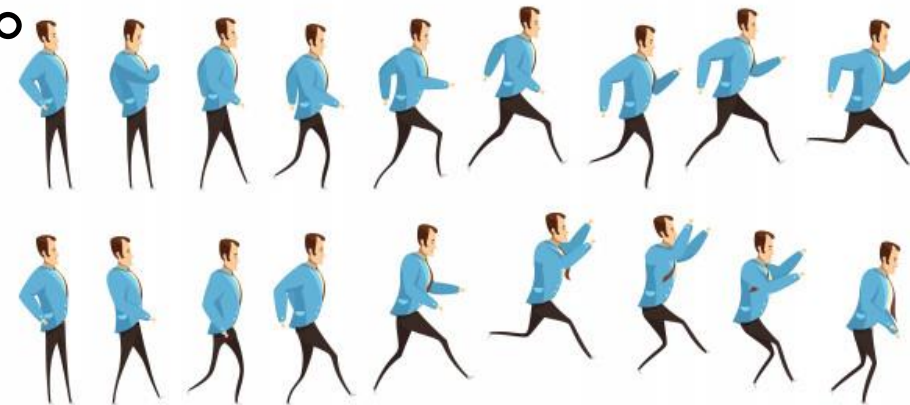
filme de tamanho 800 x 600 pontos (tamanho da tela)

com 24 bits de prof. cor. Logo, cada imagem do filme tem $800 \times 600 \times 24 = 1,44 \text{ MB}$

Seja a velocidade de 30 fps, logo, 1 segundo de filme tem $1,44 \times 30 = 43,2 \text{ MB}$

Um filme com 2 horas, 120 minutos, 7200 segundos, **tem 311 GB!!**

- Eis portanto a necessidade de compressão, através de CODECs, que é um hardware ou software para este fim (handbrake.fr). Exemplos:



OUTRAS CODIFICAÇÕES - VÍDEO

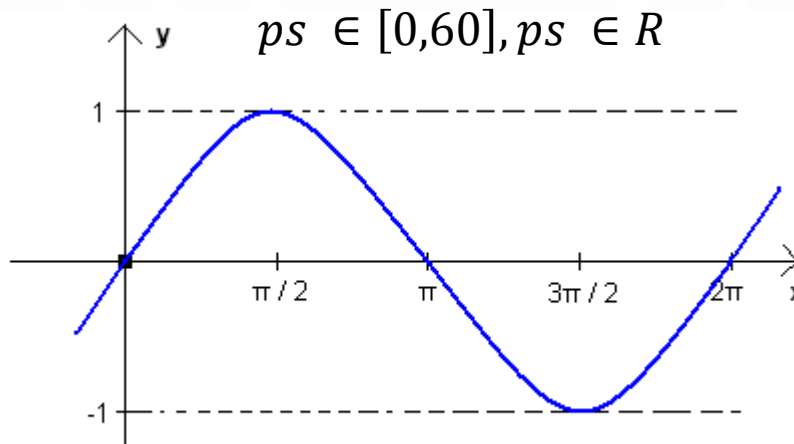
- Moving Picture Experts Group (MPEG):
 - Guarda quadros principais e simula intermediários
 - MPEG-1 (VCD): tamanho menor que um videocassete
 - MPEG-2 (1994): TV digital, DVD
 - MPEG-4 (1998): Internet, conversação
 - Quicktime (Apple), foi base para o MPEG-4, MP4
 - AVI (Microsoft) – usa conceito de quadro referência e semelhanças
 - DivX: derivado do MP4, compacta 6GB em 600MB
 - RMVB – perda de qualidade
 - WMV



No Brasil, início dos anos 80 (VHS) até mais ou menos 2006, com a chegada do DVD. Sistemas de cores NTSC, Pal-M

DIFERENÇAS ANALÓGICO X DIGITAL

- Digitalização analógico - digital



Este valor analógico pode ser de qualquer grandeza física, como temperatura, velocidade, pressão, aceleração, tempo, som, luz, etc.

<https://www.youtube.com/watch?v=eqX-56g6Vhw>

Imagine um relógio cujos ponteiros percorrem de modo **CONTÍNUO**, ou seja, **sem saltos**, os 360 graus do círculo. Giro contínuo. Isto representa a essência do conceito de analógico puro. Ou seja, na teoria, entre 12 e 1h, podem haver infinitas medidas de tempo, com qualquer precisão possível.



$ps \in [0,60], ps \in N$

Agora imagine um relógio que o ponteiro dos segundos se movimenta aos saltos, a cada 1s. Dizemos que este comportamento é **DISCRETO**, pois há um intervalo de tempo definido entre os saltos.

<https://www.youtube.com/watch?v=4vtM5hpPmgU>

DIFERENÇAS ANALÓGICO X DIGITAL

- Digitalização analógico – digital: quando estes intervalos discretos são representados **digitalmente**, ou seja, com o sistema binário, tem-se um relógio digital
- Imaginemos cada segundo representado por seu equivalente binário



Horas	Minutos	Segundos
0 – 00000	0 – 000000	0 – 000000
1 – 00001	1 – 000001	1 – 000001
2 – 00010	2 – 000010	2 – 000010
...
23 – 10111	59 – 111011	59 – 111011

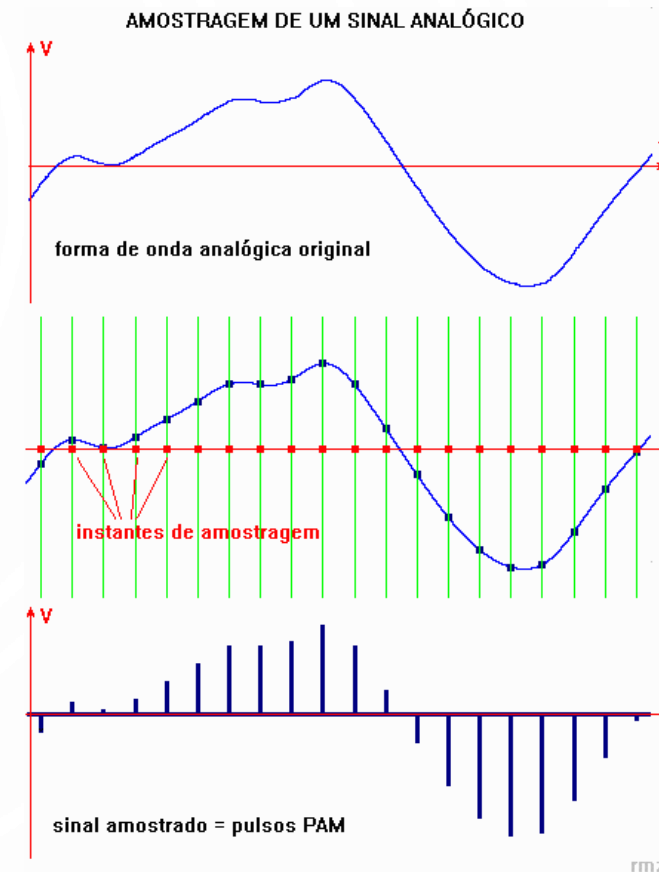
APLICAÇÃO - ÁUDIO

- Digitalização analógico - digital

- Taxa ou frequência de amostragem
- Sinal analógico $x(t)$ representado por sequência de números $x(kT)$, com $k = 0, 1, 2, \dots$ (instantes discretos) e $T =$ instante de amostragem

Por exemplo: se $T = 0.1 \text{ s}$, $f = 1/T = 10\text{Hz}$; se $T = 0.001\text{s}$ (1ms), $f = 1 \text{ kHz}$, 1000 Hz

- Além da frequência de amostragem, o número de bits utilizado para representar o sinal digitalizado tem influência na qualidade, pois permite armazenar “mais dados” (amplitudes) próximos ao original



OUTRAS CODIFICAÇÕES - ÁUDIO

- Digitalização analógico - digital
 - Quantização: representar amplitudes de cada amostra por um número binário (degraus). Quanto mais bits, mais degraus, ou seja, pode-se representar intervalos menores, portanto, com maior detalhes do áudio original, embora o arquivo tenha tamanho maior
- Exemplos:
 - 8 bits (256 níveis), 16 bits (65536 níveis)...
 - 10 kHz, 8 bits – telefone, AM
 - 20 kHz – FM
 - 40 kHz, 16 bits – CD
 - Estéreo ou mono também interfere no tamanho
 - Formatos WAV, AIFF, MP3, WMA, AAC etc.

