

Arquitetura de Computadores

PROF. DR. ISAAC

Sistemas de numeração.

Representações Binárias, Octais e Hexadecimais

Números binários necessitam de mais dígitos que números em outras bases.

Converter para bases múltiplas de 2 , como 8 ou 16, é mais simples do que converter para base 10.

Representações Binárias, Octais e Hexadecimais

Sistema de Numeração Hexadecimal:

- O sistema de numeração hexadecimal é muito utilizado na programação de microprocessadores.
- Tal como nos demais sistemas de numeração, podemos desenvolver qualquer número em potências da sua base, neste caso 16.
- Inclui os valores de 0-9 e A-F

$$A_{16} = (10)_{10}$$

Representações Binárias, Octais e Hexadecimais

Binário	Octal	Decimal	Hexadecimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Código Binário Decimal (código BCD)

Um código BCD separa cada dígito em um código de 4 bits.

BCD – binary-coded decimal

BCD	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Código Binário Decimal (código BCD)

Exemplo:

Representar o número **842 7590**, em BCD.

Solução:

Decimal:	8	2	4	7	5	9	0
BCD:	1000	0010	0100	0111	0101	1001	0000

Designadores de Base em Assembler

Base	Designador	Exemplo
2	B	10010010B
8	O ou Q	373O ou 373Q
10	D	356D ou 356
16	H	7F54H

Conversão Binário para Octal

Particionar o número da base 2 em grupos de 3, e preencher os grupos mais externos com zeros.

Converter cada triplo para o octal equivalente.

Exemplo:

$$(\mathbf{10110})_2 = (010)_2(110)_2 = (2)_8(6)_8 = (\mathbf{26})_8$$

Conversão Binário para Hexadecimal

Segue a mesma ideia anterior, só que agora, agrupar o número binário em grupos de 4;

Exemplo:

$$(10110110)_2 = (1011)_2(0110)_2 = (B)_{16}(6)_{16} = (B6)_{16}$$

Conversão Hexadecimal para Binário

É o inverso do anterior, cada dígito em hexadecimal gera quatro dígitos binário;

Exemplo:

$$(A4)_{16} = (A)_{16}(4)_{16} = (1010)_2(0100)_2 = (\mathbf{10100100})_2$$

Representação de números inteiros.

Inteiro sem sinal: exemplo de 4 bits

Esta é a representação mais simples, os números de 0 a 15 podem ser representados. Não tem números negativos.

0000 0	0001 1	0010 2	0011 3
0100 4	0101 5	0110 6	0111 7
1000 8	1001 9	1010 10	1011 11
1100 12	1101 13	1110 14	1111 15

Números Inteiros sem Sinal

Exemplo:

Qual o intervalo de número podemos representar com 8 bits de números inteiros sem Sinal?

Números Inteiros sem Sinal

Exemplo:

Qual o intervalo de número podemos representar com 16 bits de números inteiros sem Sinal?

Números Inteiros sem Sinal

Exemplo:

Qual o intervalo de número podemos representar com 32 bits de números inteiros sem Sinal?

Números Inteiros sem Sinal :

Exemplo em C

```
1  #include <iostream>
2
3  int main() {
4      unsigned int valor = 4294967295;
5      unsigned int *ponteiro;
6      ponteiro = &valor;
7
8      printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

Endereço 0x7ffee71f8b0c, valor: 0xffffffff

```
1  #include <iostream>
2
3  int main() {
4      unsigned int valor = 4294967296;
5      unsigned int *ponteiro;
6      ponteiro = &valor;
7
8      printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

./main.cpp:4:24: **error:** implicit conversion from 'long' to 'unsigned int' changes value from 4294967296 to 0 [-Werror,-Wconstant-conversion]

unsigned int valor = 4294967296;

NNNNN ^NNNNNNNNNN

1 error generated.

make: *** [Makefile:10: main] Error 1

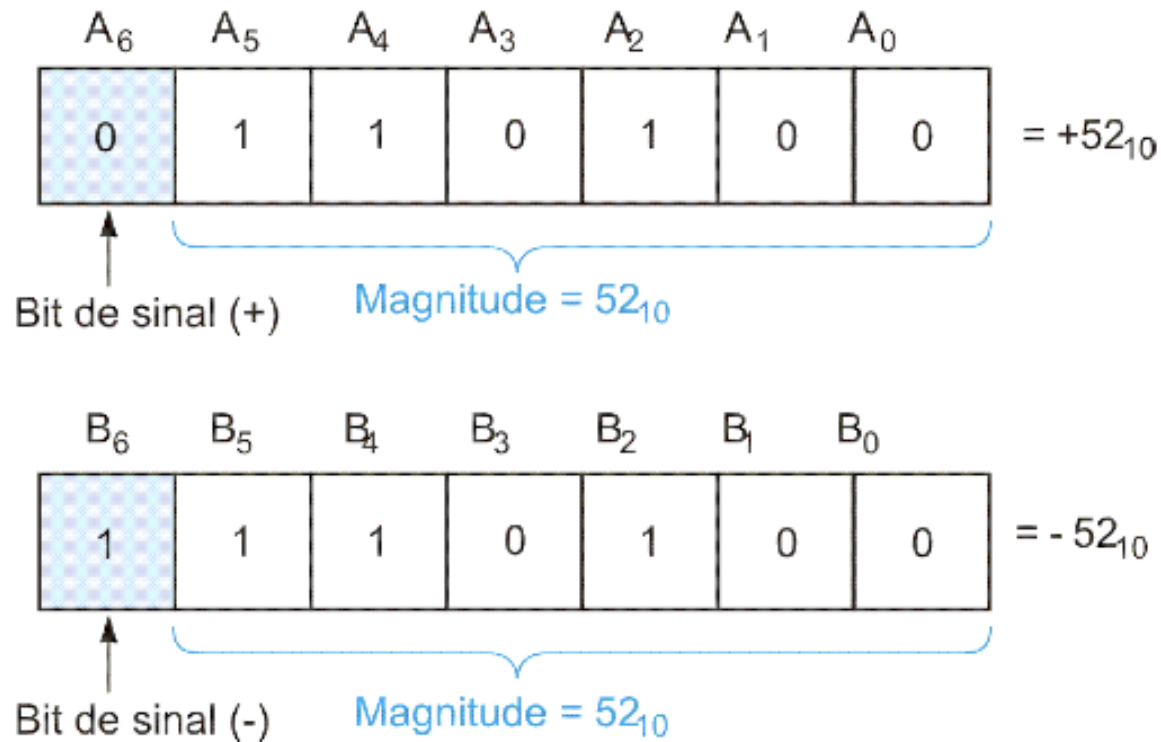
Números Inteiros

Formas de representar:

- 1. Bit de sinal (Sign-Magnitude).
- 2. Complemento de 2 (mais utilizado).
- 3. Representação Excessiva.

Números Inteiros – Bit de Sinal

Exemplo de 7 bits:



Números Inteiros – Bit de Sinal

exemplo de 4 bits

Bit mais significativo: 0 - positivo; 1 - negativo.

Números de [-7 a 7]. 0 é duplicado.

0000 0	0001 1	0010 2	0011 3
0100 4	0101 5	0110 6	0111 7
1000 0	1001 -1	1010 -2	1011 -3
1100 -4	1101 -5	1110 -6	1111 -7

Números Inteiros – Bit de Sinal

Exemplo:

Como é representado o número **-6** utilizando a representação por bit de Sinal em máquinas de 8 bits?

Solução:

- **10000110**

Números Inteiros – Bit de Sinal

Exemplo:

Qual intervalo de números podemos representar com 8 bits utilizando o bit de Sinal?

Números Inteiros – Bit de Sinal

Exemplo:

Qual intervalo de números podemos representar com 8 bits utilizando o bit de Sinal?

Solução:

- **+127 a -127**

Complemento de 2

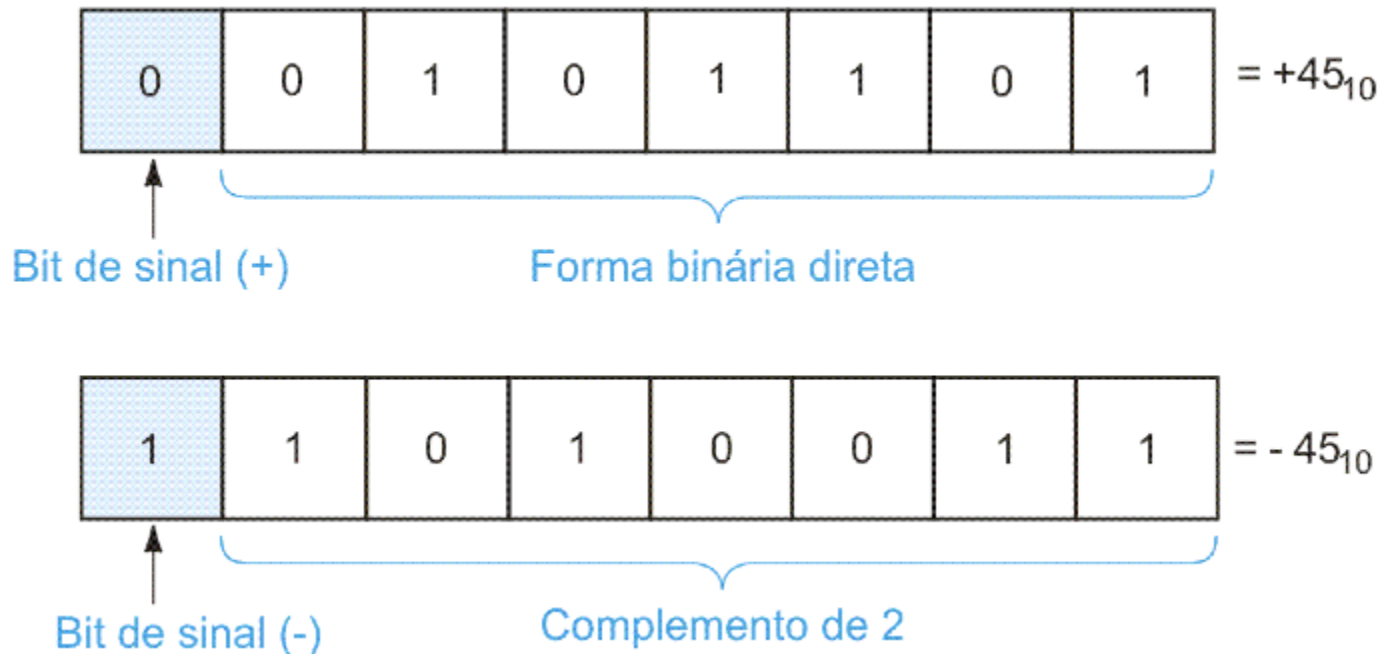
Passos:

1. Obter o complemento de 1 (inversão dos bits).
2. Adicionar 1.

$1\ 0\ 1\ 1\ 0\ 1_2$	Número binário original
$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$	
$0\ 1\ 0\ 0\ 1\ 0_2$	Complemento de 1
$\quad\quad\quad + 1$	
$\hline 0\ 1\ 0\ 0\ 1\ 1_2$	Complemento de 2

Números inteiros: Complemento de 2 com Sinal

Utiliza-se um bit de sinal acrescido da representação em complemento de 2 se o número for negativo.



Números Inteiros – Complemento de 2

exemplo de 4 bits

Números de [-8 a 7].

0000 0	0001 1	0010 2	0011 3
0100 4	0101 5	0110 6	0111 7
1000 -8	1001 -7	1010 -6	1011 -5
1100 -4	1101 -3	1110 -2	1111 -1

Números inteiros: Complemento de 2

Exemplo:

Como é representado o número **-5** utilizando a representação por complemento de 2 em máquinas de 8 bits?

Números inteiros: Complemento de 2

Exemplo:

Como é representado o número **-5** utilizando a representação por complemento de 2 em máquinas de 8 bits?

Solução:

- **11111011**

Números inteiros: Complemento de 2 com Sinal

Exemplo:

Que faixa de números podemos representar com 8 bits utilizando o complemento de 2 ?

Números inteiros: Complemento de 2 com Sinal

Exemplo:

Que faixa de números podemos representar com 8 bits utilizando o complemento de 2 ?

Solução:

- **+127 a -128**

Números inteiros: Complemento de 2 com Sinal

Exemplo:

Que faixa de número podemos representar com 8 bits utilizando o complemento de 2?

Solução:

- **+127 a -128**

Binário	Decimal
10000000	-128
10000001	-127
10000010	-126
⋮	⋮
11111110	-2
11111111	-1
00000000	0
00000001	1
⋮	⋮
01111110	+126
01111111	+127

Números inteiros: Complemento de 2 com Sinal

Exemplo:

Que faixa de números podemos representar com 16 bits
utilizando o complemento de 2 ?

Números inteiros: Complemento de 2 com Sinal

Exemplo:

Que faixa de números podemos representar com 32 bits utilizando o complemento de 2 ?

Números inteiros:

Exemplo em C

```
1  #include <iostream>
2
3  int main() {
4      int valor = -10;
5      int *ponteiro;
6      ponteiro = &valor;
7
8      printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

Endereço 0x7ffc7a7c786c, valor: 0xffffffff6

Números inteiros:

Exemplo em C de overflow

```
1  #include <iostream>
2
3  int main() {
4      int valor = 2147483647;
5      int *ponteiro;
6      ponteiro = &valor;
7
8      printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

Endereço 0x7ffc4d92541c, valor: 0x7fffffff

```
1  #include <iostream>
2
3  int main() {
4      int valor = 2147483648;
5      int *ponteiro;
6      ponteiro = &valor;
7
8      printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

./main.cpp:4:15: **error:** implicit conversion from 'long' to 'int' changes value from 2147483648 to -2147483648 [-Werror,-Wconstant-conversion]

int valor = 2147483648;

NNNNN ^NNNNNNNNNN

1 error generated.

make: *** [Makefile:10: main] Error 1

Números inteiros:

Exemplo em C de underflow

```
1 #include <iostream>
2
3 v int main() {
4     int valor = -2147483648;
5     int *ponteiro;
6     ponteiro = &valor;
7
8     printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

Endereço 0x7ffc0ea3301c, valor: 0x80000000

```
1 #include <iostream>
2
3 v int main() {
4     int valor = -2147483649;
5     int *ponteiro;
6     ponteiro = &valor;
7
8     printf( "Endereço %p, valor: 0x%x\n", ponteiro, *ponteiro);
9
10 }
```

./main.cpp:4:15: **error:** implicit conversion from 'long' to 'int' changes value from -2147483649 to 2147483647 [-Werror,-Wconstant-conversion]

```
    int valor = -2147483649;
```

~~~~~ ^~~~~~

1 error generated.

make: \*\*\* [Makefile:10: main] Error 1

# Números inteiros: Complemento de 2

---

Por que devemos nos preocupar com overflow e underflow?

# Números inteiros: Exemplo de Overflow

## Exemplo:

---

Em 2012, a música "Gangnam Style" do cantor sul-coreano Psy se tornou um fenômeno global e rapidamente quebrou recordes de visualizações do Youtube, ultrapassando 2 bilhões de visualizações.

O problema é que o contador de visualizações do YouTube atingiu o limite máximo de 32 bits do inteiro, que era de 2.147.483.647.

Com isso o Youtube foi obrigado a mudar seu contador para 64 bits, para que as visualizações do vídeo continuasse a contagem.

<https://www.bbc.com/news/world-asia-30288542>

# Números inteiros: Complemento de 2

Soma com complemento de 2.

---

- Soma os dois números.
- Despreza vai um que não cabe.

$$\begin{array}{r} 0011 \\ \underline{0100+} \\ 0111 \end{array} \quad \begin{array}{l} (3) \\ (4) \\ (7) \end{array}$$

$$\begin{array}{r} 0110 \\ \underline{1100+} \\ 0010 \end{array} \quad \begin{array}{l} (6) \\ (-4) \\ (2) \end{array}$$

$$\begin{array}{r} 1101 \\ \underline{1100+} \\ 1001 \end{array} \quad \begin{array}{l} (-3) \\ (-4) \\ (-7) \end{array}$$

# Números inteiros: Representação Excessiva

---

Na Representação Excessiva o número é "deslocado" por um valor chamado de *bias* (viés) ou *excess* (excesso).



# Números inteiros: Representação Excessiva

## Exemplo:

---

Representar os números  $(+12)_{10}$  e  $(-12)_{10}$  com 8 bits usando um *bias* de 128.

Solução:

Representação excessiva

- $(128 + 12 = 140)_{10}$

$$(+12)_{10} = (10001100)_2$$

- $(128 + -12 = 116)_{10}$

$$(-12)_{10} = (01110100)_2$$

# Números inteiros: Representação Excessiva

## Exemplo:

---

Como é representado o número **-5** utilizando a representação excessiva em máquinas de 8 bits usando um *bias* de **127** ?

# Números inteiros: Representação Excessiva

## Exemplo:

---

Como é representado o número **-5** utilizando a representação excessiva em máquinas de 8 bits usando um *bias* de **127** ?

Solução:

Representação excessiva

- $(127 + 5 = 132)_{10}$

$$(+5)_{10} = (10000100)_2$$

- $(127 + -5 = 122)_{10}$

$$(-5)_{10} = (01111010)_2$$

# Números Inteiros: representação em 3 bits

| <u>Decimal</u> | <u>Unsigned</u> | <u>Sign-Mag.</u> | <u>1's Comp.</u> | <u>2's Comp.</u> | <u>Excess 4</u> |
|----------------|-----------------|------------------|------------------|------------------|-----------------|
| 7              | 111             | —                | —                | —                | —               |
| 6              | 110             | —                | —                | —                | —               |
| 5              | 101             | —                | —                | —                | —               |
| 4              | 100             | —                | —                | —                | —               |
| 3              | 011             | 011              | 011              | 011              | 111             |
| 2              | 010             | 010              | 010              | 010              | 110             |
| 1              | 001             | 001              | 001              | 001              | 101             |
| +0             | 000             | 000              | 000              | 000              | 100             |
| -0             | —               | 100              | 111              | 000              | 100             |
| -1             | —               | 101              | 110              | 111              | 011             |
| -2             | —               | 110              | 101              | 110              | 010             |
| -3             | —               | 111              | 100              | 101              | 001             |
| -4             | —               | —                | —                | 100              | 000             |

# Código de Caracteres

---

Os caracteres são representados por números, as representações mais comuns de caracteres são:

- ASCII
- ISO 8859
- Unicode

# ASCII

Os caracteres são representados em apenas 7 bits.

|    |     |    |     |    |    |    |   |    |   |    |   |    |   |    |     |
|----|-----|----|-----|----|----|----|---|----|---|----|---|----|---|----|-----|
| 00 | NUL | 10 | DLE | 20 | SP | 30 | 0 | 40 | @ | 50 | P | 60 | ` | 70 | p   |
| 01 | SOH | 11 | DC1 | 21 | !  | 31 | 1 | 41 | A | 51 | Q | 61 | a | 71 | q   |
| 02 | STX | 12 | DC2 | 22 | "  | 32 | 2 | 42 | B | 52 | R | 62 | b | 72 | r   |
| 03 | ETX | 13 | DC3 | 23 | #  | 33 | 3 | 43 | C | 53 | S | 63 | c | 73 | s   |
| 04 | EOT | 14 | DC4 | 24 | \$ | 34 | 4 | 44 | D | 54 | T | 64 | d | 74 | t   |
| 05 | ENQ | 15 | NAK | 25 | %  | 35 | 5 | 45 | E | 55 | U | 65 | e | 75 | u   |
| 06 | ACK | 16 | SYN | 26 | &  | 36 | 6 | 46 | F | 56 | V | 66 | f | 76 | v   |
| 07 | BEL | 17 | ETB | 27 | '  | 37 | 7 | 47 | G | 57 | W | 67 | g | 77 | w   |
| 08 | BS  | 18 | CAN | 28 | (  | 38 | 8 | 48 | H | 58 | X | 68 | h | 78 | x   |
| 09 | HT  | 19 | EM  | 29 | )  | 39 | 9 | 49 | I | 59 | Y | 69 | i | 79 | y   |
| 0A | LF  | 1A | SUB | 2A | *  | 3A | : | 4A | J | 5A | Z | 6A | j | 7A | z   |
| 0B | VT  | 1B | ESC | 2B | +  | 3B | ; | 4B | K | 5B | [ | 6B | k | 7B | {   |
| 0C | FF  | 1C | FS  | 2C | ,  | 3C | < | 4C | L | 5C | \ | 6C | l | 7C |     |
| 0D | CR  | 1D | GS  | 2D | -  | 3D | = | 4D | M | 5D | ] | 6D | m | 7D | }   |
| 0E | SO  | 1E | RS  | 2E | .  | 3E | > | 4E | N | 5E | ^ | 6E | n | 7E | ~   |
| 0F | SI  | 1F | US  | 2F | /  | 3F | ? | 4F | O | 5F | _ | 6F | o | 7F | DEL |

# ISO 8859

---

ISO 8859 é uma série de conjuntos de caracteres multilíngues (8859-1, 8859-2, etc) codificados em um byte (8 bits) padronizados para escrita em determinados idiomas:

ISO 8859-1: West European

ISO 8859-2: East European

ISO 8859-3: South European

ISO 8859-4: North European

ISO 8859-5: Cyrillic

...

# ISO 8859-1

Codepage 819 - Latin 1 - ISO 8859-1

|    | -0   | -1   | -2   | -3   | -4   | -5   | -6   | -7   | -8   | -9   | -A   | -B   | -C   | -D   | -E   | -F   |
|----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0- |      | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 | 0008 | 0009 | 000A | 000B | 000C | 000D | 000E | 000F |
| 1- | 0010 | 0011 | 0012 | 0013 | 0014 | 0015 | 0016 | 0017 | 0018 | 0019 | 001A | 001B | 001C | 001D | 001E | 001F |
| 2- |      | !    | "    | #    | \$   | %    | &    | '    | (    | )    | *    | +    | ,    | -    | .    | /    |
| 3- | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | :    | ;    | <    | =    | >    | ?    |
| 4- | @    | A    | B    | C    | D    | E    | F    | G    | H    | I    | J    | K    | L    | M    | N    | O    |
| 5- | P    | Q    | R    | S    | T    | U    | V    | W    | X    | Y    | Z    | [    | \    | ]    | ^    | _    |
| 6- | `    | a    | b    | c    | d    | e    | f    | g    | h    | i    | j    | k    | l    | m    | n    | o    |
| 7- | p    | q    | r    | s    | t    | u    | v    | w    | x    | y    | z    | {    |      | }    | ~    |      |
| 8- |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| 9- |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| A- |      | ¡    | ¢    | £    | ¤    | ¥    | ¦    | §    | ¨    | ©    | ª    | «    | ¬    | ®    | ¯    |      |
| B- | °    | ±    | ²    | ³    | ´    | µ    | ¶    | ·    | ¸    | ¹    | º    | »    | ¼    | ½    | ¾    | ¿    |
| C- | À    | Á    | Â    | Ã    | Ä    | Å    | Æ    | Ç    | È    | É    | Ê    | Ë    | Ì    | Í    | Î    | Ï    |
| D- | Ð    | Ñ    | Ò    | Ó    | Ô    | Õ    | Ö    | ×    | Ø    | Ù    | Ú    | Û    | Ü    | Ý    | Þ    | ß    |
| E- | à    | á    | â    | ã    | ä    | å    | æ    | ç    | è    | é    | ê    | ë    | ì    | í    | î    | ï    |
| F- | ð    | ñ    | ò    | ó    | ô    | õ    | ö    | ÷    | ø    | ù    | ú    | û    | ü    | ý    | þ    | ÿ    |

Sempre  
igual

Muda com a  
língua



# Unicode

---

O ASCII e o EBCDIC representam apenas caracteres do (Latin).

Existem muitos outros conjuntos de caracteres nas diferentes linguas, então o Unicode foi desenvolvido para diversos caracteres de diversas linguas.

# Unicode

---

Unicode é um padrão em evolução. Ele muda à medida que novos conjuntos de caracteres são introduzidos nele.

Na versão 2.0 do padrão Unicode, há 38.885 caracteres codificados distintos que cobrem as principais línguas escritas das Américas, Europa, Oriente Médio, África, Índia, Ásia e Pacífica.

# Unicode

|          |         |        |          |           |          |        |        |
|----------|---------|--------|----------|-----------|----------|--------|--------|
| 0000 NUL | 0020 SP | 0040 @ | 0060 `   | 0080 Ctrl | 00A0 NBS | 00C0 À | 00E0 à |
| 0001 SOH | 0021 !  | 0041 A | 0061 a   | 0081 Ctrl | 00A1 ¡   | 00C1 Á | 00E1 á |
| 0002 STX | 0022 "  | 0042 B | 0062 b   | 0082 Ctrl | 00A2 ¢   | 00C2 Â | 00E2 â |
| 0003 ETX | 0023 #  | 0043 C | 0063 c   | 0083 Ctrl | 00A3 £   | 00C3 Ã | 00E3 ã |
| 0004 EOT | 0024 \$ | 0044 D | 0064 d   | 0084 Ctrl | 00A4 ¤   | 00C4 Ä | 00E4 ä |
| 0005 ENQ | 0025 %  | 0045 E | 0065 e   | 0085 Ctrl | 00A5 ¥   | 00C5 Å | 00E5 å |
| 0006 ACK | 0026 &  | 0046 F | 0066 f   | 0086 Ctrl | 00A6 ¦   | 00C6 Æ | 00E6 æ |
| 0007 BEL | 0027 '  | 0047 G | 0067 g   | 0087 Ctrl | 00A7 §   | 00C7 Ç | 00E7 ç |
| 0008 BS  | 0028 (  | 0048 H | 0068 h   | 0088 Ctrl | 00A8 ¨   | 00C8 È | 00E8 è |
| 0009 HT  | 0029 )  | 0049 I | 0069 i   | 0089 Ctrl | 00A9 ©   | 00C9 É | 00E9 é |
| 000A LF  | 002A *  | 004A J | 006A j   | 008A Ctrl | 00AA ±   | 00CA Ê | 00EA ê |
| 000B VT  | 002B +  | 004B K | 006B k   | 008B Ctrl | 00AB «   | 00CB Ë | 00EB ë |
| 000C FF  | 002C ,  | 004C L | 006C l   | 008C Ctrl | 00AC ¬   | 00CC Ì | 00EC ì |
| 000D CR  | 002D -  | 004D M | 006D m   | 008D Ctrl | 00AD −   | 00CD Í | 00ED í |
| 000E SO  | 002E .  | 004E N | 006E n   | 008E Ctrl | 00AE ®   | 00CE Î | 00EE î |
| 000F SI  | 002F /  | 004F O | 006F o   | 008F Ctrl | 00AF ¯   | 00CF Ï | 00EF ï |
| 0010 DLE | 0030 0  | 0050 P | 0070 p   | 0090 Ctrl | 00B0 °   | 00D0 Ð | 00F0 ð |
| 0011 DC1 | 0031 1  | 0051 Q | 0071 q   | 0091 Ctrl | 00B1 ±   | 00D1 Ñ | 00F1 ñ |
| 0012 DC2 | 0032 2  | 0052 R | 0072 r   | 0092 Ctrl | 00B2 º   | 00D2 Ò | 00F2 ò |
| 0013 DC3 | 0033 3  | 0053 S | 0073 s   | 0093 Ctrl | 00B3 »   | 00D3 Ó | 00F3 ó |
| 0014 DC4 | 0034 4  | 0054 T | 0074 t   | 0094 Ctrl | 00B4 ´   | 00D4 Ô | 00F4 ô |
| 0015 NAK | 0035 5  | 0055 U | 0075 u   | 0095 Ctrl | 00B5 µ   | 00D5 Õ | 00F5 õ |
| 0016 SYN | 0036 6  | 0056 V | 0076 v   | 0096 Ctrl | 00B6 ¶   | 00D6 Ö | 00F6 ö |
| 0017 ETB | 0037 7  | 0057 W | 0077 w   | 0097 Ctrl | 00B7 ·   | 00D7 × | 00F7 ÷ |
| 0018 CAN | 0038 8  | 0058 X | 0078 x   | 0098 Ctrl | 00B8 ¸   | 00D8 Ø | 00F8 ø |
| 0019 EM  | 0039 9  | 0059 Y | 0079 y   | 0099 Ctrl | 00B9 ÿ   | 00D9 Ù | 00F9 ù |
| 001A SUB | 003A :  | 005A Z | 007A z   | 009A Ctrl | 00BA ˆ   | 00DA Ú | 00FA ú |
| 001B ESC | 003B ;  | 005B [ | 007B {   | 009B Ctrl | 00BB »   | 00DB Û | 00FB û |
| 001C FS  | 003C <  | 005C \ | 007C     | 009C Ctrl | 00BC 1/4 | 00DC Ü | 00FC ü |
| 001D GS  | 003D =  | 005D ] | 007D }   | 009D Ctrl | 00BD 1/2 | 00DD Ý | 00FD ý |
| 001E RS  | 003E >  | 005E ^ | 007E ~   | 009E Ctrl | 00BE 3/4 | 00DE ŷ | 00FE þ |
| 001F US  | 003F ?  | 005F _ | 007F DEL | 009F Ctrl | 00BF ¾   | 00DF Š | 00FF ŷ |


# UTF-8

**UTF-8** é denominado Formato de Transformação UTF-8 UCS, em que UCS significa Universal Character Set (conjunto de caracteres universal), que é Unicode na essência.

Códigos UTF-8 têm tamanho variável, de 1 a 4 bytes, e podem codificar cerca de dois bilhões de caracteres. Ele é o conjunto de caracteres dominante em uso na Web.

# Exemplo em Python

```
print("\U0001F600")
```



The image shows a snippet of a Python code editor. On the left, a code editor window contains the line `print("\U0001F600")` in a monospaced font. To the right of the code editor is a vertical scrollbar. Further to the right is a panel with a dropdown arrow, a 'Run' button, and a yellow smiley face emoji (😄) below the dropdown arrow.

# Exercícios

---

# Exercícios.

## Exercício 1:

---

- a) Representar o número **-127** em complemento de 2 de 8bits.
- b) Representar o número **-2** em complemento de 2 de 8bits.
- c) Representar o número **6** no excesso de 127 de 8bits.
- d) Representar o número **-6** no excesso de 127 de 8bits.
- e) Representar o número **5** no excesso de 1023 de 11bits.
- f) Explique o que é overflow e underflow em relação à representação de números inteiros.

# Exercícios.

## Respostas:

---

- a)  $1000\ 0001 = 81h$
- b)  $1111\ 1110 = FEh$
- c)  $(127 + 6 = 133) = 1000\ 0101$
- d)  $(127 + -6 = 121) = 0111\ 1001$
- e)  $(1023 + 5 = 1028) = 100\ 0000\ 0100$



# Bibliografia

---

Murdocca, Miles, and Vincent Heuring. Computer Architecture and Organization. Vol. 271. New York, NY, USA: John Wiley & Sons, 2007.

ZELENOVSKY, R.; MENDONÇA, A. Microcontroladores Programação e Projeto com a Família 8051. MZ Editora, RJ, 2005.

Gimenez, Salvador P. Microcontroladores 8051 - Teoria e Prática, Editora Érica, 2010.