Representação de dados

Prof. Fabio Henrique Silva

Descrição

Organização e arquitetura de computadores. Utilização dos sistemas de numeração e representação de dados.

Propósito

Introduzir os fundamentos básicos da representação de dados desde o bit — a menor unidade de representação dos dados em um sistema de computação — até os caracteres, apresentados em aplicações, em um nível de abstração mais próximo do usuário.

Objetivos

Módulo 1

Sistemas de computação

Identificar as unidades de informação utilizadas pelos sistemas de computação.

Módulo 2

Sistemas de numeração e operações aritméticas

Descrever os sistemas de numeração a partir da prática de operações aritméticas.

Módulo 3

Conversão

Empregar a conversão entre os sistemas de numeração.

Módulo 4

Representação de dados

Categorizar as tabelas de representação de dados.

Introdução

É muito importante dominar conceitos fundamentais que determinam qualquer área de conhecimento. Um dos conceitos fundamentais da computação está relacionado ao termo bit. O computador é uma máquina concebida a partir de componentes eletrônicos. De um modo geral, o bit possibilita representar os sinais elétricos em informações que possuem significados úteis para a criação e para o processamento de tarefas.

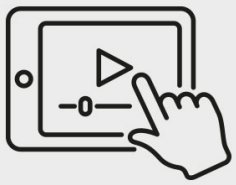
Veremos também os fundamentos dos sistemas de numeração posicionais. Eles são indispensáveis para o entendimento da formação dos números nas chamadas bases de numeração, ou seja, a quantidade de símbolos diferentes possíveis. Daremos um enfoque especial às seguintes bases: decimal, binária, octal e hexadecimal.

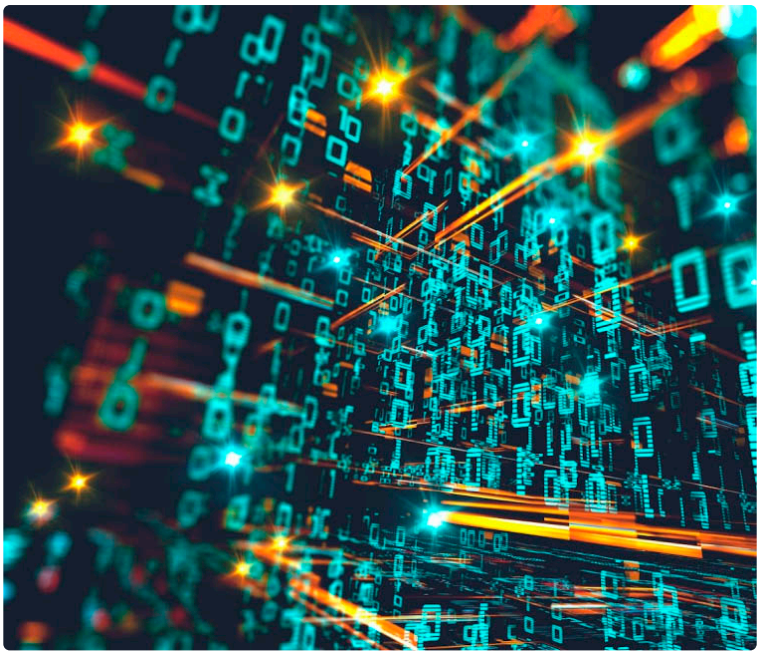
Entenderemos como são convertidos os valores entre bases numéricas distintas e, além disso, realizaremos operações aritméticas de adição e subtração em outras bases diferentes da base decimal. Dominar esse procedimento é importante para possibilitar a alternância entre os sistemas de numeração decimal (utilizados normalmente em nosso cotidiano) e três outros sistemas: binário, octal e hexadecimal (usados pelo computador).

Os bits constituem a forma básica da representação de dados de um computador. Suas informações são obtidas conforme padrões de representações. Afinal, o computador precisa utilizar valores numéricos para representar os sinais gerados no hardware. No entanto, estamos habituados a representar dados e informações por meio de letras, palavras, frases etc. Esse processo se repete quando um usuário comum deseja receber uma informação em caracteres. Apontaremos de que maneira ocorre a representação de conjuntos de bits em caracteres, além de apresentarmos tabelas de representação de dados.

Vamos lá!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



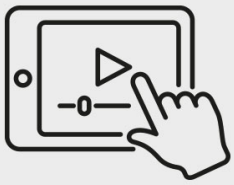


1 - Sistemas de computação

Ao final deste módulo, você será capaz de identificar as unidades de informação utilizadas pelos sistemas de computação.

Unidades de informação

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Você já comparou como são realizados os processos de comunicação entre o computador e o ser humano?

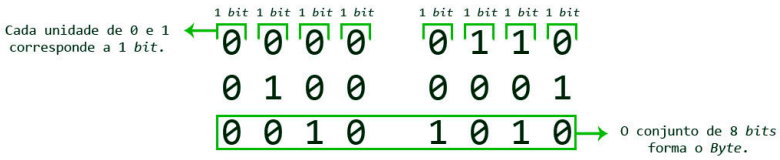
As pessoas normalmente podem criar textos formados por frases, as quais, por sua vez, utilizam palavras compostas por letras (caracteres).

> **Da mesma forma, um sistema de computação precisa usar representações simbólicas para que o processamento dos dados seja realizado corretamente e em conformidade com as expectativas de seus projetistas e usuários.**

Demonstraremos, a seguir, como as unidades de informação são fundamentais para a realização desse processo.

O computador armazena e move os dados eletronicamente sob a forma de voltagem ou corrente elétrica. A representação dos valores elétricos é feita na forma binária; portanto, ela utiliza somente dois valores: ***0 e 1***. Quando os bits são agrupados em um conjunto ordenado com **oito valores**, temos a unidade de medida denominada byte.

Você pode ver no exemplo a seguir várias sequências de 8 bits, que formam o byte. Essas sequências são entendidas pelo computador e decodificadas, dando origem ao caractere.



Por isso, a menor unidade de informação possível em um sistema de computação é o bit.

> ### Vamos entender como é feita a conversão entre bit e byte?

Sabemos que 8 bits = 1 byte. Então, como convertemos 512 bits em byte?

Fazendo uma regra de três simples:

$$8\ bits \Leftrightarrow 1\ byte$$

$$x\ bits \Leftrightarrow x$$

$$x = 512 \div 8 = 64\ bytes$$

Desse modo, a regra geral para a conversão é:

De bit para byte

Dividimos o valor por 8.

De byte para bit

Multiplicamos o valor por 8.

Apontaremos, a seguir, quatro conjuntos formados nas unidades de informação:

- Um conjunto ordenado de bytes, que representa uma informação útil para os computadores, constitui uma **palavra**.
- Um conjunto estruturado de palavras forma um **registro**.
- Um conjunto organizado de registros forma um **arquivo**.
- Um conjunto organizado de arquivos forma um banco de **dados**.

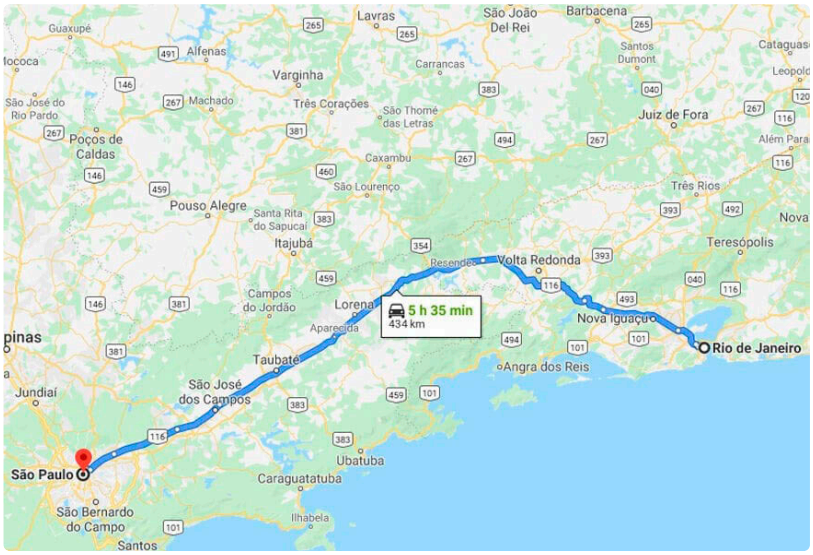
Múltiplos e submúltiplos

Da mesma forma que fazemos no nosso cotidiano, podemos empregar **múltiplos e submúltiplos** para grandezas nos sistemas de unidades a fim de exprimir valores de grandeza maiores ou menores que uma unidade.

Por exemplo, para medir a grandeza **comprimento**, é usada a unidade de base **metro**, conforme o no Sistema Internacional de Medidas. Veja o exemplo a seguir:

Medido na Via Dutra, o comprimento entre Rio de Janeiro e São Paulo é de aproximadamente 434.000 metros.

Esse valor pode ser abreviado usando o prefixo **quilo**, que é múltiplo do metro no fator de 10^3 (1.000). Assim, dizemos que 434 km (quilômetros) é igual a $434 \times 1.000 = 434.000$ metros.



Mapa indicando a distância entre Rio de Janeiro e São Paulo.

Também podemos utilizar prefixos de múltiplos e submúltiplos das unidades bit e byte. Em geral, a abreviação simbólica do bit é realizada com o “b” minúsculo e a do byte, com o “B” maiúsculo.

Observe estas representações:

4.000 Kb ($4.000 \times 10^3 = 4.000.000$ bits)

567 MB ($567 \times 10^6 = 567.000.000$ bytes)

Atenção!

No exemplo, a representação dos múltiplos e submúltiplos foi realizada com uma base numérica decimal, que possui 10 números (de 0 a 9), já que a utilizamos normalmente no nosso dia a dia. Entretanto, seus resultados expressam valores aproximados (e não exatos) da quantidade de bits ou bytes medidos em um computador.

Como vimos, o bit é a menor unidade de informação dos **computadores atuais**. Como só possui dois valores possíveis (0 e 1), ele é representado pela base numérica **binária**.

Se reescrevermos o exemplo anterior representando os valores decorrentes do cálculo com a base 2, obteremos:

4.000 Kb ($4.000 \times 2^{10} = 4.096.000$ bit)

567 MB ($567 \times 2^{20} = 594.542.592$ bytes)

A tabela a seguir mostra alguns prefixos empregados na computação a fim de abreviar valores medidos em uma unidade base (por exemplo, bit ou byte) para valores em potências de 2 (binário) e de 10 (decimal):

Unidade	Valor em potência de 2	Valor unitário
$1K$(quilo)	2^{10}	1024
$1M$(mega)	2^{20}	$1.048.576$
$1G$(giga)	2^{30}	$1.073.741.824$
$1T$(tera)	2^{40}	$1.099.511.627.$

Fabio Henrique Silva.

Esses são os prefixos tipicamente adotados na computação com unidades de medida para exprimir valores de grandezas muito maiores ou menores que aqueles da unidade usada sem um prefixo.

Os valores unitários podem ser obtidos a partir da notação dos múltiplos da grandeza em:

Potência de 2

Expressam os valores (em decimal) que, de fato, são manipulados pelo computador em binário.

Potência de 10

Exprimem os valores (em decimal) equivalentes àqueles manipulados pelo computador em binário, embora empreguem os múltiplos no sistema decimal comumente utilizado no nosso dia a dia.

Suponha que você queira mudar o prefixo do valor a seguir:

$$52{,}9\ GB \Rightarrow TB\ ?$$

Usando como referência os valores mostrados na tabela anterior, veja uma maneira de realizar isso:

1. Escrever os prefixos em sua sequência (B, KB, MB etc.).

2. Colocar o valor (no caso, $52{,}9$) embaixo da unidade com o prefixo dado. Esse será seu ponto de partida ou ponto de referência relacionado ao novo valor obtido.

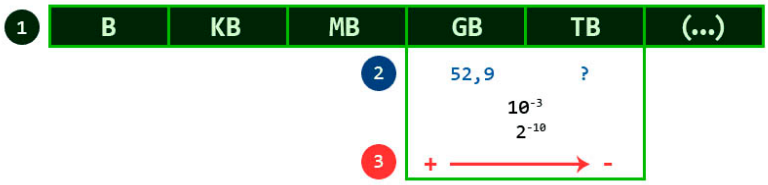
Desenhar uma seta no sentido do prefixo pretendido. Do lado esquerdo dela, fica o operador matemático "+"; do direito, o sinal "−". O sinal resultante fica ao lado do valor do expoente da base utilizada.

Seta

O sentido da seta pode ter duas direções:

Esquerda — determina o múltiplo do valor (expoente positivo).
Direita — indica o submúltiplo do valor (expoente negativo).

Assim, você terá:



Conforme os valores mostrados na tabela anterior, observemos sua aplicação com múltiplos nas potências de 10 e de 2:

Potência de 10

$$52{,}9 \times 10^{-3} = 0{,}0529\ TB$$

Isso ocorrerá se considerarmos os múltiplos em potência de 10.

Potência de 2

$$52{,}9 \times 2^{-10}\ TB \sim 0{,}05166$$

Como o cálculo do valor na potência de 2 não é tão “trivial” quanto o cálculo do valor na potência de 10, uma sugestão é deixá-lo no formato de notação científica. Se, ainda assim, você quiser realizá-lo, ele será de aproximadamente $0{,}05166\ TB$.

> **Entre dois prefixos de unidades de medida vizinhos tipicamente utilizados na computação, o expoente da**

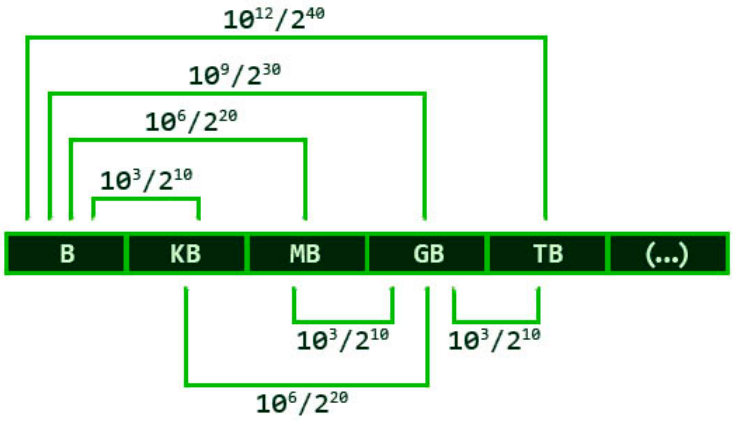
> **base decimal varia, em geral, de 3 em 3 (ou de 10 em 10, se a base for binária).**

Assim, por exemplo, entre giga (G) e mega (M), o valor para a potência de 10 é 10^3, e para a potência de 2 é 2^{10}.

Do mesmo modo, entre tera (T) e giga (G), permanecem os valores para as potências de 10 e 2: 10^3 e 2^{10}, respectivamente.

Já entre giga (G) e quilo (K), registramos o seguinte valor: $10^3 \times 10^3 = 10^6$, ou $2^{10} \times 2^{10} = 2^{20}$.

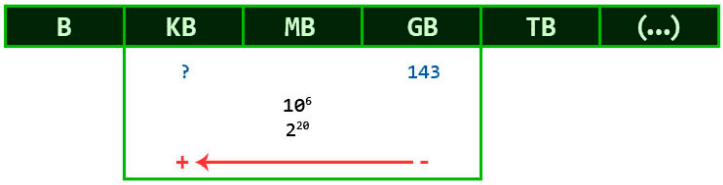
Esta imagem ilustra alguns exemplos:



Suponhamos que você deseje novamente mudar o prefixo do valor a seguir:

$$143\ GB \Rightarrow KB\ ?$$

Para isso, você deverá fazer o seguinte:



Vejamos, por fim, os cálculos nas potências de 10 e de 2:

Potência de 10

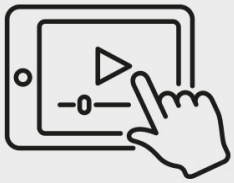
$$143 \times 10^6 = 143.000.000\ KB$$

Potência de 2

$$143 \times 2^{20} = 149.946.368\ KB$$

Representação das informações no computador

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



O primeiro passo **indispensável** para evoluir em qualquer área de atuação pretendida no ramo da computação é entender como um computador funciona, verificando como as informações são representadas dentro dele.

Por exemplo, para cada tipo de dado, serão amplamente utilizados os seguintes métodos padrão de codificação:

Padrão ASCII

O padrão ASCII pode associar um número binário de 7 bits a cada um dos 128 caracteres distintos possíveis.

Formato de arquivo MP3

O formato de arquivo MP3 especifica o modo de codificar um arquivo de áudio como uma sequência de 0s e 1s.

Formato de imagem .png

O formato de imagem .png especifica os pixels das imagens digitais como uma sequência de 0s e 1s.

Representação de dados e conversão de unidades de medida

Neste vídeo, o professor Fabio Henrique Silva reforça alguns conceitos da representação de dados e pratica a conversão entre as unidades de medida.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

A quantos *bits* equivalem *256 bytes*?

A *32 bits*

B *256 bits*

C *1.024 bits*

D *2.048 bits*

E *4.096 bits*

Parabéns! A alternativa D está correta.

Considerando que *8 bits* é igual a *1 byte* e fazendo uma regra de três simples, temos o seguinte resultado:

8 bits --- 1 byte
x --- 256 bytes

$x = 256 \times 8 = 2.048\ bits$

Questão 2

Passe $0,876\ MB$ para o prefixo de múltiplo ou submúltiplo da unidade indicada: TB.

Qual é o seu resultado?

A $0,000000876\ TB$

B $0,00876\ TB$

C $876\ TB$

D $876.000\ TB$

E $876.000.000\ TB$

Parabéns! A alternativa A está correta.

Note que estamos considerando os múltiplos em potências de 10. Desse modo, devemos fazer o cálculo segundo os valores mostrados na tabela que estudamos de prefixos adotados na computação: $0,876 \times 10^{-6} = 0,000000876\ TB$.

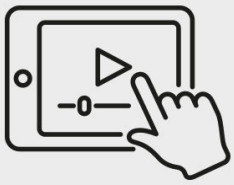
Além disso, devemos escrever os prefixos em sequência e colocar o valor de partida embaixo do prefixo dado, desenhando a seta no sentido do prefixo desejado. Se o sentido dela for para a esquerda, multiplicaremos o valor fornecido pela base elevada ao expoente positivo. Caso ela se volte para a direita, a multiplicação desse valor será feita com o expoente negativo. Veja:

2 - Sistemas de numeração e operações aritméticas

Ao final deste módulo, você será capaz de descrever os sistemas de numeração a partir da prática de operações aritméticas.

Sistemas de numeração

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Bases de numeração

Os fundamentos dos sistemas de numeração posicionais são indispensáveis para o entendimento da formação dos números nas chamadas bases de numeração. Daremos um enfoque às seguintes bases:

Decimal

Base numérica mais utilizada no nosso cotidiano.

Binária

Base usada pelos computadores no processamento de dados.

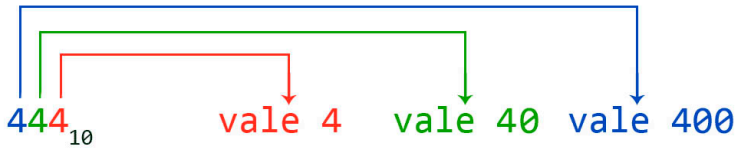
Octal e hexadecimal

Múltiplos da base binária.

Para entendermos o funcionamento dos sistemas de numeração, precisaremos apresentá-los.

Posicional

Cada algarismo tem um valor relativo (diferente) de acordo com sua posição no número. Observe:



Sistema decimal

Neste exemplo, à medida que os algarismos são acrescentados à esquerda, o número cresce de valor em grupos de 10.

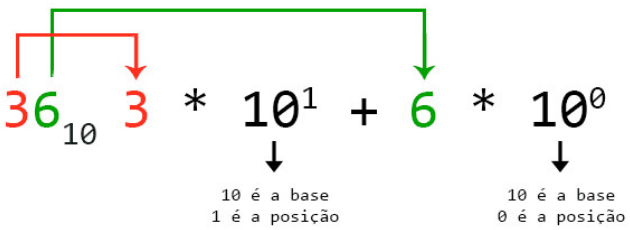
> Uma **base** é a quantidade de símbolos de um sistema posicional. Seu número indica o total de símbolos dela.

Veja o exemplo a seguir:

- Base 10 (decimal): $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.
- Base 2 (binária): 0, 1.
- Base 8 (octal): $0, 1, 2, 3, 4, 5, 6, 7$.
- Base 16 (hexadecimal): $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$.

Cada algarismo de um número demonstra o valor de sua **posição** em potências da base. A unidade de uma posição de um sistema de base "Z" tem valor equivalente a "Z" unidades da posição imediata à direita.

Observe:



Todo número cresce de valor da direita para a esquerda

Em qualquer sistema posicional, o **número** é formado da seguinte maneira:

Da direita para a esquerda

Cresce a partir do valor 0 (seguido do 1 até o último algarismo válido).

Exemplo: bases 3 (algarismos 0 a 2) e 10 (algarismos 0 a 9).

Retorna a 0

Quando a contagem chega ao último algarismo válido de uma posição, ela retorna a 0 e cresce uma unidade para a esquerda.

Exemplo: base 10 — cresce de 0 a 9. Depois, $10, 11, \ldots, 19$ — Na direita, retorna a 0 e cresce 1 para a esquerda: 20.

Os movimentos descritos neste exemplo se repetem infinitamente.

Não posicional

Todo algarismo tem valor fixo independentemente de sua posição no número.

No sistema de numeração romano, o algarismo X vale sempre 10:

Sistema de numeração romano

Este sistema de numeração é não posicional.

$$XXX = 30$$

$$XL = 40$$

$$LX = 60$$

Operações aritméticas com números inteiros em qualquer base

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Agora, vamos estudar as operações aritméticas em bases numéricas diferentes. Dentro de uma unidade especializada (**unidade lógica** e **aritmética**), o processador usa seus circuitos lógicos digitais para realizar operações lógicas e aritméticas, tendo o bit como unidade para representação de dados.

Veremos, a seguir, as operações de adição e subtração.

Adição

As parcelas são somadas algarismo por algarismo (pares da mesma coluna), partindo da posição mais à direita até a última posição à esquerda.

Se o resultado da soma de dois algarismos (fazendo em decimal) for igual ou maior que o valor da base, o excesso será subtraído do valor da base e o resultado será colocado na respectiva posição. Passamos uma unidade ("vai *1*") para a posição imediata à esquerda, que será somada aos dois algarismos seguintes. Observe os exemplos a seguir para entender melhor esse conceito.

+

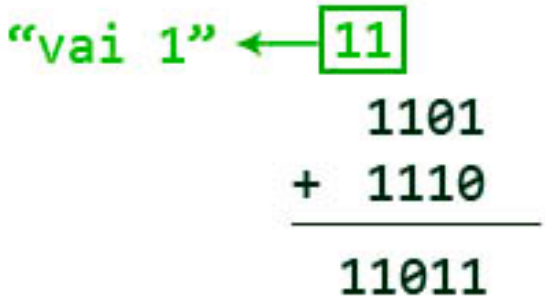
Na base *10*

Base numérica mais utilizada no nosso cotidiano.

$$\begin{array}{r} 11 \\ 574 \\ +\ 893 \\ \hline 1467 \end{array}$$

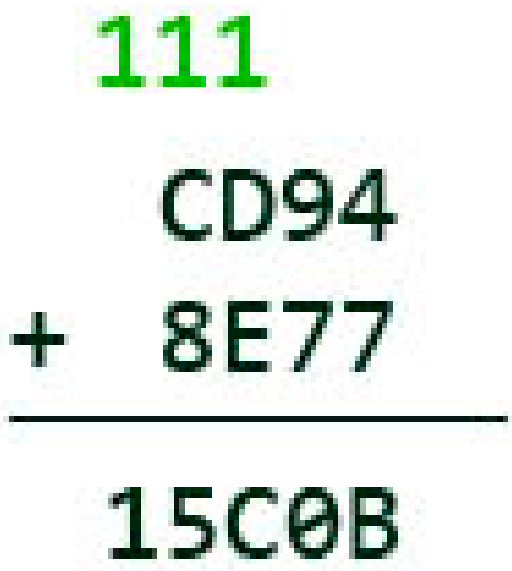
Na base 2

Na base binária, $1 + 1 = 0$, com o "vai 1" $= 10$
Logo, veja na imagem:



Na base 16

Antes de abordarmos as regras para cálculos com números hexadecimais, é importante sabermos a equivalência dos números decimais e seus respectivos hexadecimais.



Equivalência dos números decimais e seus respectivos hexadecimais

Base 10	0	1
Base 16	0	1

Base 10	16	17
Base 16	10	11

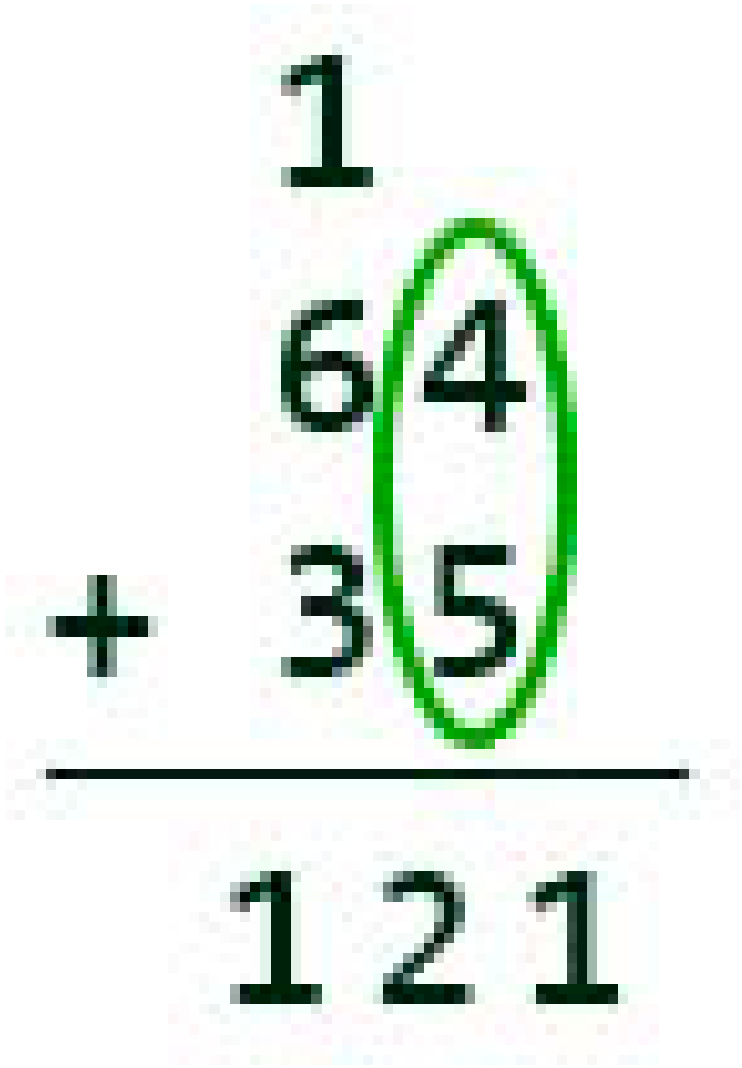
Uma maneira de pensar a adição em qualquer base é:

- Somar os números na posição na base 10.
- Diminuir o número obtido nessa base pelo valor dela se ele passar do seu último algarismo.
- Não se esquecer do "vai 1".

Veja o exemplo na base 8:

Somar na base 10 (decimal)$: 4 + 5 = 9$.
Como passou do último algarismo (7), diminuímos 9 da base $8: 9 - 8 = 1$.
"Vai-1" para a próxima posição, $6 + 3 + 1 = 10$. Como passou de $7: 10 - 8 = 2$.

"Vai-1" para a próxima posição. Logo, vale 1.



Subtração

As parcelas são subtraídas algarismo por algarismo (pares da mesma coluna) do mais à direita até o último à esquerda.

Em relação ao minuendo (algarismo superior) e ao subtraendo (inferior), há duas possibilidades no processo de subtração:

Minuendo maior que subtraendo

Realiza a operação e registra sua diferença embaixo.

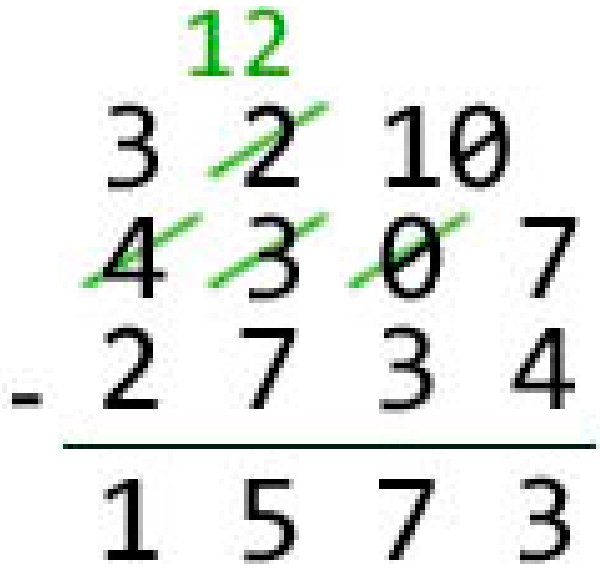
Minuendo menor que subtraendo

É subtraída uma unidade do algarismo à esquerda (trata-se do famoso "pedir 1 emprestado").

O **valor da base** é somado ao valor existente de minuendo à direita, efetuando-se, assim, a subtração.

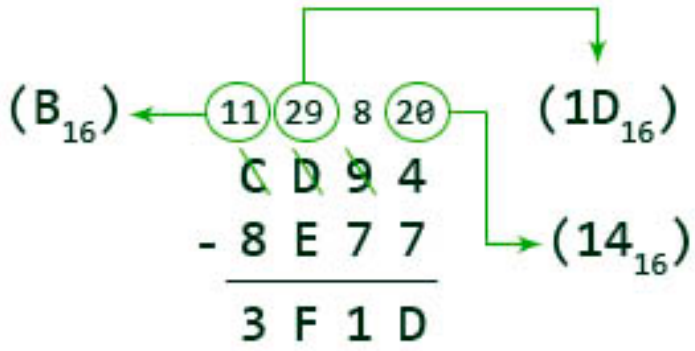
Observemos os exemplos a seguir para entendermos melhor esse conceito:

Na base 10



Na base 16

O “empresta 1” no hexadecimal tem valor igual a 16.



Na 1ª posição: em $4 - 7$, “pedimos emprestado" o valor da base para o dígito na posição ao lado: $16 + 4 = 20$. Logo: $20 - 7 = 13$. Em hexadecimal, 13 (na base 10) é igual a D (na base 16).

Na 2ª posição: o valor 9 vale 8 porque "emprestou" ao dígito na orimeira posição do minuendo: $8 - 7 = 1$.

Na 3ª posição: o valor D na base 16 vale 13 na base 10, enquanto o valor E na base vale 14 na base 10. "Pedindo

emprestado", o dígito da 2ª posição no minuendo será: $13 + 16 = 29$. Logo: $29 - 14 = 15$, que, na base 16, é representado por F.

Na 4ª posição: o valor C na base 16 vale 12 na base 10. Já o valor 12 vale 11, porque "emprestou" ao dígito na 1ª posição do minuendo: $11 - 8 = 3$.

Na base 2

No binário, a operação parece ser mais complicada. Vejamos a seguir:

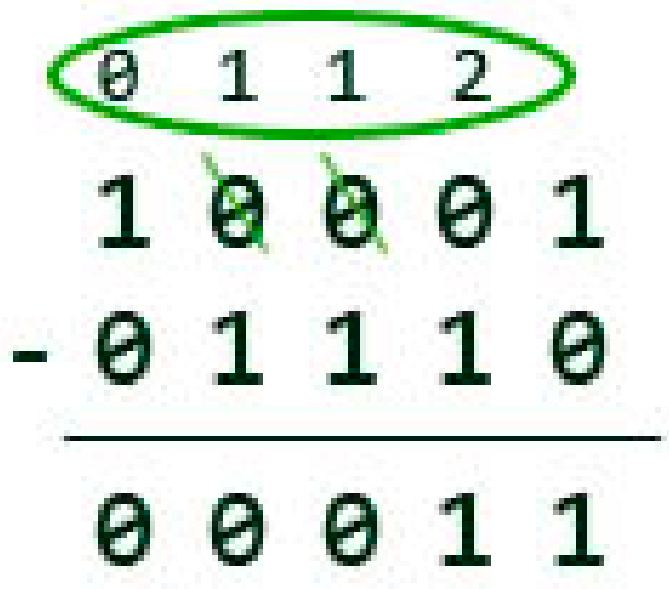
- $0 - 1 = 1$ (e "pede emprestado" do dígito seguinte);
- $1 - 1 = 0$;
- $1 - 0 = 1$;

Quando temos $0 - 1$, precisamos "pedir emprestado" do primeiro algarismo diferente de 0. Esse empréstimo **vale 2 (valor dois em decimal)** por ser um número binário.

Então, no caso da coluna $0 - 1 = 1$, a operação feita é $2 - 1 = 1$. Como esse processo se repete, o elemento que cedeu o "empréstimo" e valia 1 passa a valer 0.

É possível verificar que, quando encontramos $0 - 1$:

- O zero desta coluna vale 2.
- Todos os zeros à esquerda até o primeiro "número 1" valem 1.
- Este "primeiro número 1" vale 0 (zero).
- Se acontecer novamente $0 - 1$, o processo se repetirá.



O "empresta 1" no binário tem valor 2.

Computação útil

Com recursos de computação cada vez mais baratos e ricos em possibilidades, nossa tendência é não nos preocuparmos tanto com seus aspectos fundamentais. Entretanto, o domínio desse estado da arte lhe possibilitará pensar como um projetista de máquinas e linguagens de programação, podendo desenvolver, no futuro, protótipos capazes de fazer a diferença do ponto de vista tecnológico.

> “
>
> Podemos realmente fazer computação útil com 4.096 bits de memória? [...] O avanço da tecnologia minimizou essas restrições. [...] Em um nível prático, você verá que é bastante viável desenvolver implementações em linguagem de máquina que realizam todos os tipos de tarefas.
>
> **(SEDGEWICK; WAYNE, 2017, p. 930, tradução nossa)**

A maior parte dos primeiros programas aplicativos foi implementada dessa maneira. Afinal, por muitos anos, a penalidade sofrida pelo desempenho do uso de uma linguagem de alto nível era grande demais para poder ser paga. A maioria desses códigos foi escrita em *assembly*, cujo funcionamento é similar ao da linguagem de máquina, exceto pelo fato de permitir nomes simbólicos para códigos de operação, registros e localizações de memória.

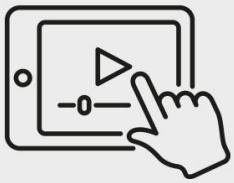
Comentário

Na década de 1970, era comum ver programas em linguagem *assembly* que se estendiam a dezenas de milhares de linhas de código.

Sistemas de numeração posicionais e não posicionais e operações aritméticas com números inteiros

Entenda os fundamentos dos sistemas de numeração posicionais e não posicionais. Ainda, compreenda como se realiza as operações aritméticas de adição e subtração com números inteiros em qualquer base.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

A partir do valor 2456 em base 7, os cinco valores seguintes são

A $2456; 2457; 2458; 2459; 2460.$

B $2456; 2457; 2460; 2461; 2461.$

C $2457; 2460; 2461; 2462; 2463.$

D $2460; 2461; 2462; 2463; 2464.$

E $2461; 2462; 2463; 2464; 2465.$

Parabéns! A alternativa D está correta.

Vamos à análise:

1º número
Partiremos do número **2456**. Lembre-se de que, quando a contagem chega ao último algarismo válido de uma posição, ela retorna a 0 e cresce uma unidade para a esquerda.

Na base 7, temos os seguintes algarismos: **0; 1; 2; 3; 4; 5; 6. Não existe número 7!**

A contagem já está no último algarismo válido de uma posição. Ele retorna a 0 e cresce uma unidade para a esquerda: **2460**.

2º, 3º, 4º e 5º números
Comece a contagem normalmente a partir do 0, voltando à "posição 0" mais à direita: **2461**; **2462**; **2463**; **2464**.

Questão 2

A partir do valor binário 1100101, os quatro números seguintes, saltando de dois em dois, são

A **1100111; 1101001; 1101011; 1101101**.

B **1100110; 1100111; 1101000; 1101001**.

C **1100101; 1100111; 1100101; 1100111**.

D **1100111; 1100112; 1100113; 1100114**.

E **1100113, 1100115, 1100117, 1100119**.

Parabéns! A alternativa A está correta.

Vamos à análise:

1º número
Partiremos do número **1100101**. Lembre-se de que, quando a contagem chega ao último algarismo válido de uma posição, ela retorna a 0 e cresce uma unidade para a esquerda.

Na base binária, temos os seguintes algarismos: **0; 1**. Siga os mesmos passos do exercício 1, mas "salte" a escrita das respostas de dois em dois. A dica, portanto, é escrever os oito números seguintes saltando de dois em dois.

Assim, temos:

1100110; 1100111; 11001000; 1101001; 1101010; 1101011; 1101100; 1101101.

"Saltando" de dois em dois números, resta esta sequência:

1100111; 1101001; 1101011; 1101101.

3 - Conversão

Ao final deste módulo, você será capaz de empregar a conversão entre os sistemas de numeração.

Conversão entre sistemas de numeração

Embora os sistemas de computação expressem seus valores na base 2 (binária), eles poderão ser representados em outras bases quando houver a necessidade de promover uma melhor visualização e compactar os seus valores. Geralmente, são usadas as bases 8 (octal) e 16 (hexadecimal), pois ambas são múltiplas de 2.

Esta tabela mostra a equivalência dos valores nas bases 2, 8, 10 e 16:

Base 2	Base 8	Base 10
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4

Base 2	Base 8	Base 10
101	5	5
110	6	6
111	7	7
1000	10	8
1001	11	9
1010	12	10
1011	13	11
1100	14	12
1101	15	13
1110	16	14
1111	17	15
10000	20	16
10001	21	17

Fabio Henrique Silva.

Em bases de valor superior a 10, usamos letras do alfabeto para representação de algarismos maiores que 9. As tabelas a seguir comparam os valores entre as bases 10 e 16:

Base 10	0	1
Base 16	0	1

Base 10	16	17
Base 16	10	11

Base 10	32	33
Base 16	20	21

Fabio Henrique Silva.

Agora conheceremos duas maneiras de realizar a conversão de valores.

De uma base X para outra Y

A conversão sempre precisa ser realizada tendo **a base 10** como intermediária, pois ela é a utilizada para efetuar cálculos aritméticos.

O vídeo abaixo mostra como converter um número escrito em uma base X para base 10.

Conversão entre sistema de numeração de uma base b para base 10

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Em seguida, converte-se o resultado da conversão para base 10 para a base Y, conforme vídeo abaixo.

Conversão entre sistema de numeração de uma base 10 para base b

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Acompanhe o passo a passo abaixo:

1

Vamos converter 234 da base 6 para um valor equivalente na base 8. O número a ser convertido é expresso em *N* produtos, em que *N* é igual à quantidade algarismo do número. Assim, temos:

$$234_6 = 2 \times 6^2 + 3 \times 6^1 + 4 \times 6^0$$

2

Calcularemos, agora, os produtos usando a aritmética da base 10. Desse modo, o resultado **será expresso em valores decimais**. Continuando a realizar o cálculo, teremos:

$$(2 \times 36) + 18 + 4 = 94_{10}$$

3

Dessa vez, o resultado (decimal 94) será convertido para a base desejada (8). Esse processo é o inverso do anterior, afinal, o inverso da multiplicação é a divisão. Dividimos o valor (94) pela base desejada (8). O resto obtido é o primeiro algarismo do número desejado (aquele mais à direita):

$94 \div 8 = 11$ e resto $= 6$

4

O quociente obtido na divisão é novamente dividido por 8. A seguir, o novo resto é acrescentado à esquerda do primeiro algarismo:

$11 \div 8 = 1$ e resto $= 3$

5

O número a ser obtido na base 8 é: 36.

6

Novamente, faremos a divisão até obtermos um quociente 0 (zero):

$1 \div 8 = 0$ e resto $= 1$

7

O número obtido, portanto, é: 136_8.

Casos especiais

As conversões de valores entre as **bases 2 e 8; 2 e 16; e 8 e 16** podem ser realizadas com este método. Tenha em mente o seguinte: se você tomar o valor da base binária e o elevar à sua posição (em decimal), obterá o valor relativo da posição da base 2 representado em decimal.

A seguir, analisaremos dois exemplos.

Valor da base binária

Como esse valor está representado em decimal, escreva 2.

Observe estes três bits:

$$2^2 = 4$$

$$2^1 = 2$$

$$2^0 = 1$$

Toda vez que o dígito 1 for inserido em uma posição, ela corresponderá ao valor relativo da posição (escrito em decimal). Em outras palavras, toda vez que você colocar:

1

"Liga" o bit na posição.

0

"Desliga" o bit na posição.

Logo, o valor final em decimal será a soma de todos os valores (escritos em decimal) relativos das posições em que estiverem "ligados".

Para "ligar" algumas posições, colocaremos, agora, o valor 1 nelas:

$$(1)\ 2^2 = 4$$

$$(0)\ 2^1 = 2$$

$$(1)\ 2^0 = 1$$

Foram colocados valores 1 nas posições 0 e 2. Somando os relativos às posições em decimal, temos: $4 + 1 = 5$.

Desse modo, o **binário 101** corresponde ao **valor 5 em decimal**.

A tabela a seguir mostra algumas correspondências:

$2^0 = 1$	$2^7 = 128$

$2^1 = 2$	$2^8 = 256$
$2^2 = 4$	$2^9 = 512$
$2^3 = 8$	$2^{10} = 1024$
$2^4 = 16$	$2^{11} = 2048$
$2^5 = 32$	$2^{12} = 4096$
$2^6 = 64$	(...)

Valores relativos às posições em decimal para os números correspondentes em binário.
Fabio Henrique Silva.

Com essas informações, podemos observar, dentro dos casos especiais citados, como são realizadas as conversões entre as seguintes bases:

Bases 2 e 8

Como $8 = 2^3$, dividimos o valor, partindo da direita para a esquerda, em grupos de três bits:

	$(111010111)_2$	
(111)	(010)	$(111)_2$
7	2	7

Fabio Henrique Silva.

Bases 2 e 16

Como $16 = 2^4$, dividimos o valor, partindo da direita para a esquerda, em grupos de quatro bits:

	$(1011011011)_2$	
(0010)	(1101)	$(1011)_2$
2	D	B

Fabio Henrique Silva.

Bases 8 e 16

Converte-se primeiramente para a base 2 e, em seguida, para a desejada:

$(3174)_8 = (\ )_{16}$

Observemos, agora, as conversões para as bases:

- $\mathbf{2} \rightarrow (011)\ (001)\ (111)\ (100)_2 = (011001111100)_2$
- $\mathbf{16} \rightarrow (0110)\ (0111)\ (1100) = (67C)_{16}$

O resultado da conversão, portanto, será: $67C$.

Lógica booleana

Além de sabermos que um computador representa seus dados usando bits, já compreendemos como são realizadas algumas operações matemáticas, bem como a conversão de valores para outras bases numéricas responsáveis por compactar os números binários.

> **Os valores binários seguem a lógica booleana. Nela, dois valores são referidos como verdadeiro e falso, sim e não ou 0 e 1. Não importa a escolha feita para essa referência: seu conceito é o mesmo.**

A conexão íntima entre a lógica booleana e os circuitos que executam tarefas computacionais é um conceito fundamental na estruturação da

infraestrutura computacional da qual desfrutamos atualmente.

Entender essa conexão significa compreender a pergunta de Sedgewick e Wayne (2017): "**Como os circuitos calculam?**". Afinal, quando você passa a saber e "sentir" como é o funcionamento dos circuitos lógicos, também consegue entender a importância da representação dos valores binários 0 e 1 em outras bases numéricas.

Lógica booleana

Expressão cuja origem remete a George Boole (1815-1864), filósofo britânico responsável pela criação da álgebra booleana. Ela foi fundamental para o desenvolvimento da computação moderna.

Conversão de valores entre bases numéricas distintas

Compreenda como os valores entre bases numéricas distintas são convertidos.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Converta o seguinte valor para a base indicada:

$10011110001011_2 = ()_{16}$

A **2E5C**

B **7834**

C **AB3D**

D **278B**

E **3E5C**

Parabéns! A alternativa D está correta.

Vamos fazer a conversão de $\mathbf{10011110001011_2}$ da base 2 (binária) para a base 16 (hexadecimal).

Divida o binário da seguinte maneira, começando da direita para a esquerda:

10 0111 1000 1011

Acrescente zeros à esquerda para completar o último grupo:

0010 0111 1000 1011

Questão 2

Converta o seguinte valor para a base indicada:

$\mathbf{2BEF5_{16} = (\ )_8}$

A **573241**

B **852387**

C **635421**

D **537365**

E **923654**

Parabéns! A alternativa D está correta.

Vamos fazer a conversão de $\mathbf{2BEF5_{16}}$.

. Da base 16 (hexadecimal) para a base 2 (binária):

. Da base 2 (binária) para a base 8 (octal):

Reagrupe os bits, começando **da direita para a esquerda**, incluindo zeros à esquerda se for necessário para completar a posição. Assim, você terá:

4 - Representação de dados

Ao final deste módulo, você será capaz de categorizar as tabelas de representação de dados.

Tabelas de representação de dados

Em uma tabela de representação de dados, cada símbolo possui uma correspondência com um grupo de bits que identifica exclusivamente determinado **caractere**. Quando alguém escreve um texto no teclado, os caracteres do alfabeto são convertidos em outros codificados em bits.

Os tipos primitivos de dados podem ser classificados em:

Caractere

Representa símbolos (não numéricos). Modo primário de introduzir dados no computador. Serve para escrever um texto em algum idioma.

Lógico

Representa verdadeiro ou falso.

Numérico

Representa os números.

Apresentaremos, a seguir, dois conjuntos de códigos de caracteres:

ASCII

Tabela de representação de dados

ASCII

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

American Standard Code for Information Interchange (em português, Código Padrão Americano para Troca de Informações).

Cada caractere tem sete bits. O ASCII possui um total de 128 caracteres (2^7) que podem ser expressos em hexadecimal.

Os códigos 0 a $1F$, por sua vez, não são impressos, pois ambos são caracteres de controle.

Tanembaum (2007) informa que o ASCII ainda possui:

Letras maiúsculas

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Letras minúsculas

a b c d e f g h i j k l m n o p q r s t u v w x y z

Sinais de pontuação

* ~ . , : ; " + = ! ? @ / # ( ) { } [ ]

Símbolos matemáticos

+ × ÷ − =

As tabelas a seguir exibem o conjunto de caracteres ASCII:

Hex	Nome	Significado
0	NUL	Null
1	SOH	Start Of Headinç
2	STX	Start Of Text
3	ETX	End Of Text
4	EOT	End Of Transmission
5	ENQ	Enquiry
6	ACK	ACKnowledgem
7	BEL	BELI
8	BS	BackSpace
9	HT	Horizontal Tab
A	LF	Line Feed
B	VT	Vertical Tab
C	FF	Form Feed
D	CR	Carriage Return
E	SO	Shift Out
F	SI	Shift In

TANEMBAUM, 2007, p. 73.

Hex	Car.	Hex
20	(Space)	30
21	!	31
22	"	32
23	#	33
24	$	34
25	%	35
26	&	36
27	'	37
28	(	38
29	)	39
2A	*	3A
2B	+	3B
2C	,	3C
2D	-	3D
2E	.	3E
2F	/	3F

TANEMBAUM, 2007, p. 73.

Unicode

Tabela de representação de dados UNICODE

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

O ASCII é um bom conjunto de caracteres para textos de língua inglesa, mas não tanto para os demais idiomas, como japonês, chinês, árabe etc. Isso levou um grupo de empresas a criar o Unicode, padrão internacional no qual cada caractere possui um único valor: 16 bits.

Com isso, podemos ter um total de 65.536 símbolos (2^{16}).

Além do alfabeto e dos símbolos de pontuação, existem códigos para:

Símbolos monetários

£ ¥ ₣ ₤ € $

Símbolos matemáticos

+ * / - =

Formas geométricas

■ ▭ △ ◯ ◆ ▱

Emojis

😀 😂 😇 😍 😎 😜

Segundo Tanembaum (2007), o consórcio Unicode estuda e decide todas as novas propostas de inclusão de novos caracteres.

Quando um usuário digita no teclado a palavra SABER (em letras maiúsculas), por exemplo, seus caracteres são convertidos conforme a tabela a seguir:

	S	A
ASCII	01010011	01000001
Unicode	U+0053	U+0041

Tabela: Conjuntos de códigos de caracteres ASCII e Unicode
Fabio Henrique Silva.

Evolução do caracteres

Os dispositivos que você usa hoje em dia suportam muitos caracteres especiais. Se você acompanhou a evolução daqueles apresentados pelos programas, lembra-se de algumas incompatibilidades que aconteciam entre aplicativos diferentes ou que estavam relacionadas a idiomas distintos.

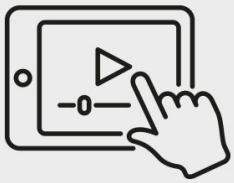
> **Essas incompatibilidades ocorriam quando não havia padronizações (como as que existem atualmente) nem uma adequação às aplicações. Isso prejudica tanto a possibilidade da melhor exploração de um negócio quanto sua acessibilidade e usabilidade.**

Tenha sempre em mente a preocupação de promover a maior compatibilidade possível entre os conjuntos de caracteres, caso você venha a trabalhar com isso em funções como programador de aplicativos de rede, programador web, webdesigner etc.

Códigos ASCII e Unicode

Neste vídeo, o professor Fabio Henrique Silva comenta a diferença entre os conjuntos de códigos ASCII e Unicode.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Assinale a alternativa que apresenta um padrão de códigos de caracteres:

A SCHOOL

B ZYXFG

C Codee

D Unicode

E Zabix

Parabéns! A alternativa D está correta.

Devido à limitação na quantidade de caracteres suportados, um novo padrão internacional para a elaboração de um conjunto de códigos deles foi criado. Visando ao suporte a caracteres de inúmeros idiomas e símbolos, no Unicode, cada caractere possui um único valor: 16 bits.

Questão 2

O trecho de uma tabela possui as seguintes codificações para caracteres:

Código decimal	Caractere
65	Letra A maiúscula
77	Letra M maiúscula
79	Letra O maiúscula
82	Letra R maiúscula

Como é a codificação dos caracteres ROMA em binário?

A 1101101 0001101 0011001 1010110

B 1111011 0111001 1010011 0101100

C 1010010 1001111 1001101 1000001

D 1100100 0000001 0111001 0110010

E 1000001, 1001101, 1001111, 1010010

Parabéns! A alternativa C está correta.

Convertendo os códigos decimais para binário, temos:

Considerações finais

O estudo de assuntos mais profundos (do ponto de vista abstrato) do computador nos permite a assimilação — ainda que de forma primária — de tudo o que ocorre tanto no nível de hardware quanto em todos os seus desdobramentos em software.

Sabendo, por exemplo, como são os sistemas de numeração binário, octal e hexadecimal, podemos compreender melhor como os tamanhos dos arquivos são medidos, de que forma as linguagens de programação podem usar a memória do computador, ou como os endereços IP em redes de computadores são construídos.

Desse modo, ao entendermos a maneira como são realizados os cálculos aritméticos, estaremos capacitados para atuar, por exemplo, em estudos e projetos de chips, além de dispositivos embarcados e de automação. Também estaremos habilitados a realizar uma interseção com outras áreas de estudo, como a Engenharia, a Eletrônica e as Telecomunicações.

Podcast

Para finalizar, ouça os principais tópicos abordados neste material.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Para conhecer os conjuntos de caracteres Unicode, pesquise na internet e acesse a página **Unicode – the world standard for text and emoji**.

Referências

HARRIS, D.; HARRIS, S. **Digital design and computer architecture**. 2. ed. San Francisco: Morgan Kaufmann, 2012.

HENNESSY, J. L. **Organização e projeto de computadores**: a interface hardware/software. 2. ed. Rio de Janeiro: LTC, 2000.

Instituto Nacional de Metrologia, Qualidade e Tecnologia. INMETRO. **Resumo do Sistema Internacional de Unidades** - SI. Tradução da publicação do BIPM. Consultado em meio eletrônico em: 7 abr. 2020.

MONTEIRO, M. **Introdução à organização de computadores**. 5. ed. Rio de Janeiro: LTC, 2007.

MURDOCCA, M. J.; HEURING, V. P. **Introdução à arquitetura de computadores**. Rio de Janeiro: Campus, 2000.

PATTERSON, D. A. *et al*. **Computer architecture, a quantitative approach**. 5. ed. San Francisco: Morgan Kaufmann, 2011.

PATTERSON, D. A. *et al*. **Organização e projeto de computadores**. 2. ed. Rio de Janeiro: LTC, 2000.

SEDGEWICK, R.; WAYNE, K. **Computer science**: an interdisciplinary approach. New York: Pearson Education, 2017.

STALLINGS, W. **Arquitetura e organização de computadores**. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

TANEMBAUM, A. S. **Organização estruturada de computadores**. 5. ed. Rio de Janeiro: LTC, 2007.

Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

O que você achou do conteúdo?

☆ ☆ ☆ ☆ ☆

Relatar problema