Arquitetura e Organização de Computadores

Caros alunos, as videoaulas desta disciplina encontram-se no AVA (Ambiente Virtual de Aprendizagem).

UniFil

Unidade 2

Representação de Dados em Sistemas de Computação.

UniFil

Introdução da Unidade

Olá, amigo(a) discente! Seja bem-vindo(a)!

Vamos aumentar nessa unidade, um pouquinho mais de conhecimento sobre como funciona os computadores. Quando um sistema computacional lida com a informação, ele na verdade está trabalhando com um grande volume de números e códigos. Os sistemas de numeração binário e os códigos digitais são fundamentais para os computadores e para os sistemas eletrônicos em geral.

Nessa unidade iremos estudar como um sistema digital manipula os números e alguns códigos existentes. Na aula 1 trataremos sobre o sistema de numeração binária e as suas relações com outros sistemas de numeração, como: decimal e hexadecimal.

Na aula 2, iremos tratar sobre a aritmética digital e códigos. Sabemos que o computador realiza centenas de cálculos muito rápido, saber basicamente como os sistemas digitais fazem cálculos são de essencial importância para quem quer trabalhar na área de computação, pois, todo código que entra pelo teclado de um computador é transformado em números e depois de processado, retorna em forma de algum código para o humano.

Objetivos

- Compreender sistemas numéricos que fazem parte da computação;
- Estudar como é feita a representação da informação em sistemas binários.

Conteúdo Programático

Aula 1: Sistemas de numeração posicional.

Aula 2: Aritmética e Códigos Digitais.

Você poderá também assistir às videoaulas em seu celular! Basta apontar a câmera para os **QRCodes** distribuídos neste conteúdo.

Pode ser necessário instalar um aplicativo de leitura QRcode no celular e efetuar login na sua conta Gmail.

Aula 1 Sistemas de numeração posicional

Na unidade anterior, estudamos os componentes que faz um computador funcionar. Nessa aula vamos dar um passo adiante. Veremos nessa unidade como que um computador internamente trabalha as informações.

Então, para começarmos, precisamos entender que a informação mais básica em um sistema digital é chamada de bit. *Bit* é uma contração de *binary digit* (digito binário). Em circuitos digitais, dois níveis de tensão diferentes são usados para representar os dois bits. Geralmente, 1 é representado pela tensão maior, a qual chamamos de nível ALTO, e o 0 é representado pelo nível de tensão menor, o nível BAIXO. Essa forma de representação é denominada lógica positiva e é usada para o entendimento da informação dentro dos circuitos. Então ALTO = 1 e BAIXO = 0 (FLOYD, 2007, p.22).

No início do desenvolvimento da computação, cada projeto de arquitetura definia a sua própria quantidade de bits que iria utilizar para lidar com a informação. Existiam códigos baseados em 4 bits, 6 bits, 8 bits e 12 bits. Para padronizar, a indústria acabou adotando ou convencionando o agrupamento em 8 bits. Esse agrupamento de 8 bits ficou conhecido como *byte.*

Então vejamos: com 1 *byte* conseguimos armazenar uma informação que contenha 8 dígitos de 0 e 1. Fazendo uma simples combinação, quantos valores podemos guardar em 8 posições que podem ter 2 valores diferentes? Então se multiplicarmos as possiblidades de cada posição teremos:

$2x2x2x2x2x2x2x2=2^8 = 256$

Portanto, com um *byte* de informação conseguimos representar 256 números diferentes, ou seja, a representação de uma sequência de números de 0 a 255.

Também temos na computação o uso do termo palavra (*Word)*. Palavra é um grupo de bits que representa uma certa unidade de informação. O tamanho da palavra vai depender do caminho de dados que determinada arquitetura vai conter. Por exemplo, seu celular pode trabalhar com 4 bytes de informação, ou seja, ele opera com uma palavra de 32 bits de informação. Assim também trabalha os computadores comuns (TOCCI, 2008).

Unidades de medida de informação

Como falado anteriormente, um *byte é* composto por exatamente 8 bits. Como o volume de informação foi crescendo de maneira gigantesca, houve a necessidade do surgimento dos diversos múltiplos do byte, conforme mostra a tabela 01 abaixo.

A base numérica de trabalho usada mundialmente é 10, isso faz com que, quando a base é elevada à terceira potência, atinja a milhar exatamente com 1000 unidades.

Quando estamos falando em bytes, grupos de bits, não estamos falando em base 10, mas sim, em uma estrutura fundamentada no código binário, ou seja, na base 2. Dessa forma, quando pretendemos atingir um quilo de bytes temos que elevar essa base a algum número inteiro, até conseguir atingir a milhar 1024.

Medida	Sigla		Caracteres	
Byte		2^0	1 (8 bits)	1 byte
Kilobyte	**KB**	2^{10}	1.024	1.024 bytes
Megabyte	**MB**	2^{20}	1.048.576	1.024 KBytes
Gigabyte	**GB**	2^{30}	1.073.741.824	1.024 MBytes
Terabyte	**TB**	2^{40}	1.099.511.627.776	1.024 GBytes
Pentabyte	**PB**	2^{50}	1.125.899.906.842.624	1.024 TBytes
Hexabyte	**HB**	2^{60}	1.152.921.504.606.846.976	1.024 PBytes
Yotabyte	**YB**	2^{80}	1.208.925.819.614.630.000.000.000	1.024 Hexabyte

Tabela 01: Tabela de medidas de informação.

Fonte: <https://www.matematica.pt/faq/unidades-medida-informatica.php>.
Acesso em: 20.04.2020.

Notação posicional

O sistema numérico decimal é o mais utilizado mundialmente nos tempos atuais. Isso pelo fato que se pode contar utilizando-se dez dedos. Notação posicional é um valor numérico que pode ser representado por potências crescentes de uma base. É um modo de representação numérica na qual o valor de cada algarismo depende da sua posição relativa na composição do número (NULL, 2010).

O sistema decimal, ou base 10, utiliza dez algarismos para designar quantidades: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. Para determinar a quantidade representada por um algarismo em um número decimal, basta multiplicá-lo por uma potência de 10, com expoente igual à posição ocupada por este algarismo. No sistema decimal, as posições são numeradas da seguinte forma:

$638 = 6x10^2 + 3x10^1 + 8x10^0$

Como se pode notar, os pesos para os números inteiros são potências de dez positivas que aumentam da direita para a esquerda, começando com $10^0 = 1$.

$....10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$

Para números fracionários, os pesos são potências de dez negativas que diminuem da esquerda para a direita começando com 10^{-1}.

$...10^2\ 10^1\ 10^0\ ,10^{-1}\ 10^{-2}\ 10^{-3}$

O valor de um número decimal é a soma dos dígitos após cada um ser multiplicado pelo seu peso, vejamos os exemplos, que foram retirados de Floyd (2007):

Expresse o número decimal 47 como uma soma dos valores de cada dígito.

O dígito 4 tem um peso de 10, que é 10^1, conforme indicado pela sua posição. O dígito 7 tem um peso de 1, que é 10^0, conforme indicado pela sua posição.

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

Expresse o número decimal 568,23 como uma soma dos valores de cada dígito.

O dígito 5, na parte inteira do número, tem um peso de 100, que é 10^2, o dígito 6 tem um peso de 10, que é 10^1, o dígito 8 tem um peso de 1, que é 10^0, o dígito fracionário 2 tem um peso de 0,1, que é 10^{-1}, e o dígito fracionário 3 tem um peso de 0,01, que é 10^{-2}.

$$\begin{aligned} 568{,}23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0{,}1) + (3 \times 0{,}01) \\ &= \mathbf{500 + 60 + 8 + 0{,}2 + 0{,}03} \end{aligned}$$

Então, o sistema decimal utiliza-se da base 10. O sistema binário, ou base 2, utiliza-se apenas de dois algarismos para designar quantidades: 0 e 1. Devido a essa característica, o sistema binário é amplamente utilizado em computadores, pois pode ser representado através da presença/ausência de tensão ou corrente elétrica.

Indicação de Vídeo

Agora assista a esse vídeo, onde é bem discutido sobre notação posicional. É muito interessante.

Disponível em: <https://www.youtube.com/watch?v=J5q7s7l2EuI>. – Notação Posicional - Bases Numéricas #01 - 27 minutos.

Um número binário é um número em que os dígitos apresentam pesos. O bit mais à direita é o bit menos significativo (***LSB – least significante bit***) em um número inteiro binário e tem um peso de $2^0 = 1$. Os pesos aumentam da direita para a esquerda em potências de dois para cada bit (2^2 .... 2^1 ....2^0). O bit mais à esquerda é o mais significativo (***MSB – most significant bit***) seu peso depende do tamanho do número binário. Observe o exemplo abaixo, onde aparece um número binário (1011_2) com seus respectivos pesos e o resultado em decimal.

$1011_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 11_{10}$

Números fracionários também podem ser representados em binário colocando os bits à direita da vírgula binária, assim como os dígitos decimais fracionários são colocados à direita da

vírgula decimal. O bit mais à esquerda é o MSB em um número binário fracionário e tem um peso de $2^{-1} = 0,5$. Os pesos da parte fracionária diminuem da esquerda para a direita por uma potência negativa de dois para cada bit (FLOYD, 2007). Observe os exemplos abaixo, onde se faz transformações de binário para decimal.

Converta o número binário inteiro 1101101 para decimal.

Determine o peso de cada bit que for 1 e em seguida calcule o somatório desses pesos para obter o número decimal.

$$\begin{aligned} \text{Peso: } & 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 \\ \text{Número binário: } & 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

Converta o número binário fracionário 0,1011 para decimal.

Determine o peso de cada bit que vale 1 e então some os pesos para obter o número decimal fracionário.

$$\begin{aligned} \text{Peso: } & 2^{-1}\ \ 2^{-2}\ \ 2^{-3}\ \ 2^{-4} \\ \text{Número binário: } & 0,\ 1\ \ \ 0\ \ \ 1\ \ \ \ 1 \\ 0,1011 &= 2^{-1} + 2^{-3} + 2^{-4} \\ &= 0,5 + 0,125 + 0,0625 = \mathbf{0,6875} \end{aligned}$$

Curiosidades!

Os computadores usam números binários para selecionar posições memória. Cada posição de memória está associada a um único número denominado endereço. Alguns microprocessadores Pentium, por exemplo, têm 32 linhas de endereço, as quais possibilitam selecionar endereçar 2^{32} (4.294.967.296) posições únicas de memória.

Videoaula 1

Utilize o QRcode para assistir!

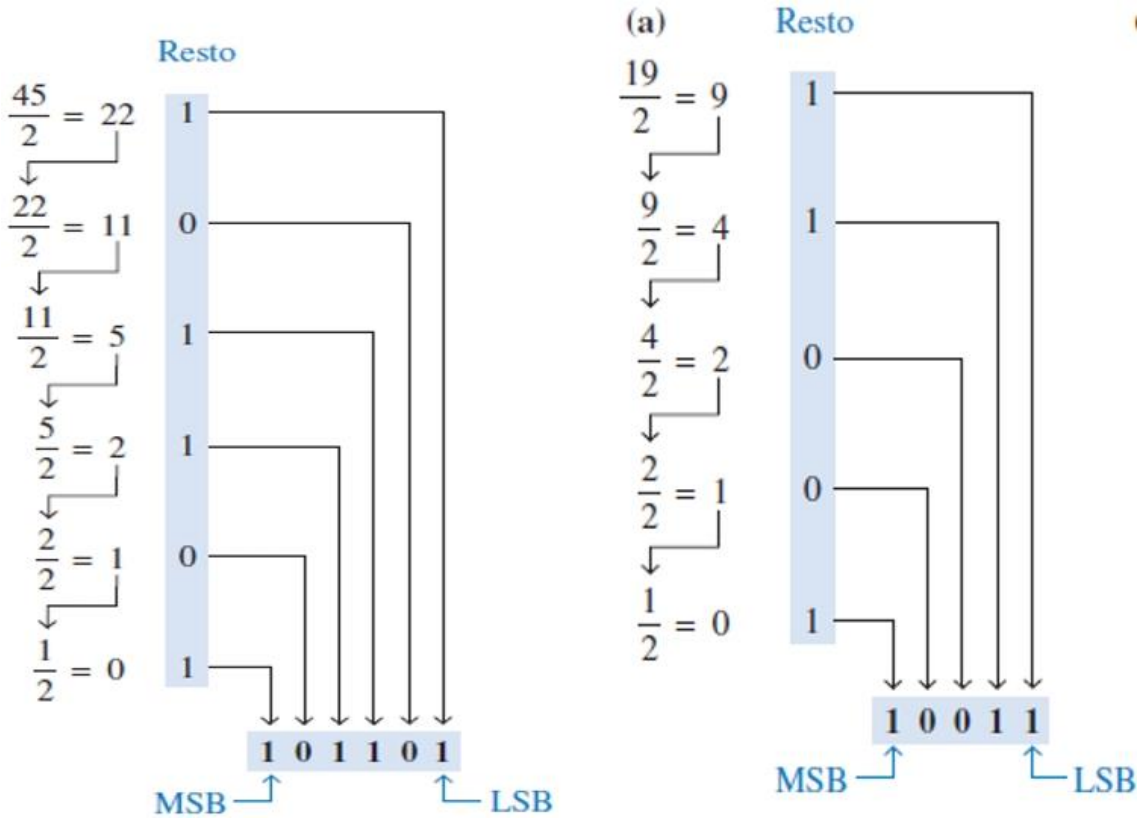
Agora, assista ao vídeo que aborda os assuntos acima estudados.

Converter números decimais em binário

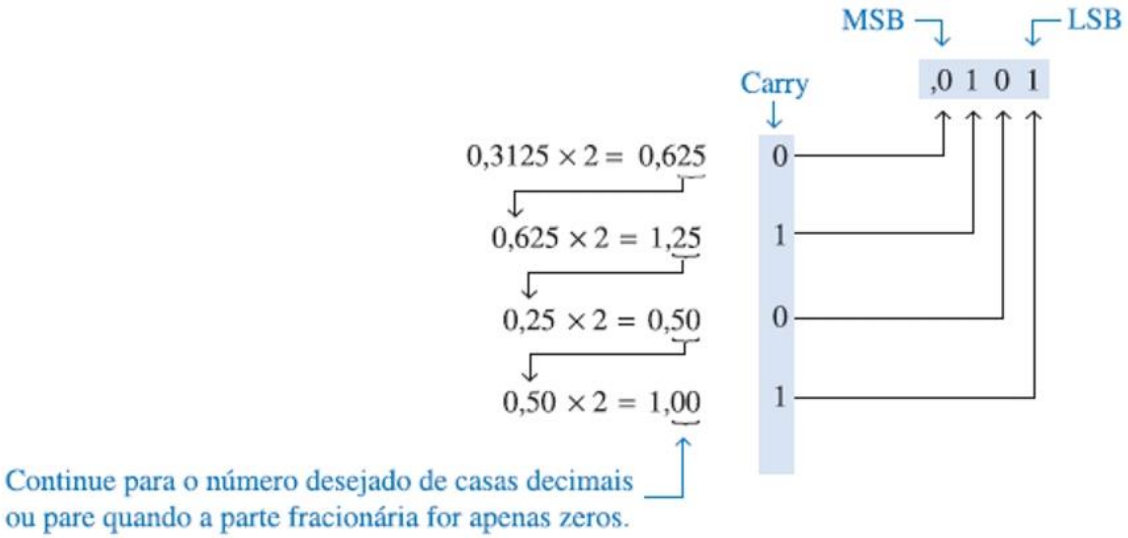
Como você notou, amigo(a) aluno(a), os seres humanos utilizam um código decimal e as máquinas utilizam um sistema binário. Então, como fazer que a máquina entenda a linguagem do ser humano? Simplesmente utilizando-se de conversões. Vamos começar convertendo decimal para o binário.

Uma forma muito comum de converter decimal para binário é a utilização da técnica das divisões sucessivas por 2. Existem outros métodos, mas vamos deixar com você, nobre aluno(a), essa pesquisa.

Exemplo: Converter o número 45 e 19 para o sistema binário. Exemplos pesquisados em Floyd (2007):



Agora vamos fazer a conversão de decimal fracionário para Binário. Como vimos, podemos transformar um inteiro decimal em binário fazendo divisões sucessivas. Para transformar decimais fracionários para binário, devemos fazer multiplicação sucessiva por 2. Vamos converter o número 0,3125. Observe o exemplo abaixo. Exemplo pesquisado em Floyd (2007):



Comece multiplicando 0,3125 por 2 e então multiplicar por 2 cada parte fracionária resultante do produto até que o produto seja 0 ou até que o número desejado de casas decimais seja alcançado. Os dígitos inteiros (*carry* mostrado no exemplo), gerados pela multiplicação formam o número binário. O primeiro inteiro gerado é o **MSB** e o último é o **LSB**.

Videoaula 2

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda os assuntos acima estudados.

Sistema de numeração Hexadecimal

O sistema de numeração hexadecimal usa a base 16. Dessa forma, ele terá 16 símbolos possíveis para os dígitos. Ele utiliza os dígitos de 0 a 9 mais as letras A, B, C, D, E e F como os 16 símbolos. O sistema hexadecimal é muito utilizado em computação, visto que ele utiliza bases em potência de 2. Essas bases são convertidas para binário facilmente e de binário de volta para ela. A tabela 2, mostra a relação entre Hexadecimal, Binário, Decimal.

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Tabela 2: relação entre Hexadecimal, Binário, Decimal.

Fonte: elaborado pelo próprio autor.

Curiosidades!

Com as memórias dos computadores na faixa de gigabytes (GB), especificar um endereço de memória em binário é bastante incômodo. Por exemplo, precisamos de 32 bits para especificar um endereço numa memória de 4 GB. É muito mais fácil expressar um código de 32 bits usando 8 dígitos hexadecimais.

Conversão Binário em hexadecimal

Vamos começar com essa conversão, pois ela nos mostrará algumas facilidades em conversões. A conversão binária em hexa é uma conversão direta. Apenas separe o número binário em grupos de 4 bits, começando do bit mais à direita. Vejamos alguns exemplos retirado de (FLOYD,2007).

Converta os seguintes números binários para hexadecimal:

(a) 1100101001010111 (b) 1111110001011010 01

(a) 1100 1010 0101 0111
↓ ↓ ↓ ↓
C A 5 7 $= CA57_{16}$

(b) 0011 1111 0001 0110 1001
↓ ↓ ↓ ↓ ↓
3 F 1 6 9 $= 3F169_{16}$

E daí, amigo(a) aluno(a)! Notou como é fácil essa conversão? Agora você sabe o porquê do uso desse sistema em computação. E agora se fizéssemos ao contrário? Binário para Hexa. Simplesmente é o inverso do Binário para o Hexadecimal. Precisamos apenas substituir cada símbolo hexadecimal por quatro bits correspondentes. Veja mais um exemplo:

Determine os números binários correspondentes aos seguintes números hexadecimais:

(a) $10A4_{16}$ (b) $CF8E_{16}$ (b) 9742_{16}

(a) 1 0 A 4 → 0001 0000 1010 0100

(b) C F 8 E → 1100 1111 1000 1110

(c) 9 7 4 2 → 1001 0111 0100 0010

Conversão hexadecimal em decimal

Vamos pensar como um computador agora? Na prática o computador pega números binários e transformam em decimais, ou seja, algo que o ser humano reconheça. Então, primeiramente converta o hexadecimal para binário, e depois binário para decimal. Observe o exemplo abaixo:

Converta o seguinte número hexadecimal em decimal:

(a) $1C_{16}$ (b) $A85_{16}$

Lembre-se, converta primeiro o número hexadecimal em binário e em seguida converta o número binário para decimal.

(a) 1 C

$0001\,1100 = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$

(b) A 8 5

$1010\,1000\,0101 = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$

Outra forma de converter é multiplicar o valor decimal de cada dígito hexadecimal pelo seu peso e realizar a soma dos produtos. Os pesos são potências de 16 crescentes da direita para esquerda: $16^3\,16^2\,16^1\,16^0$ = 4096..256..16..1. Observe o exemplo abaixo, conversão dos números hexa $E5_{16}$ e $B2F8_{16}$ em decimal. Exemplo retirado de Floyd (2007):

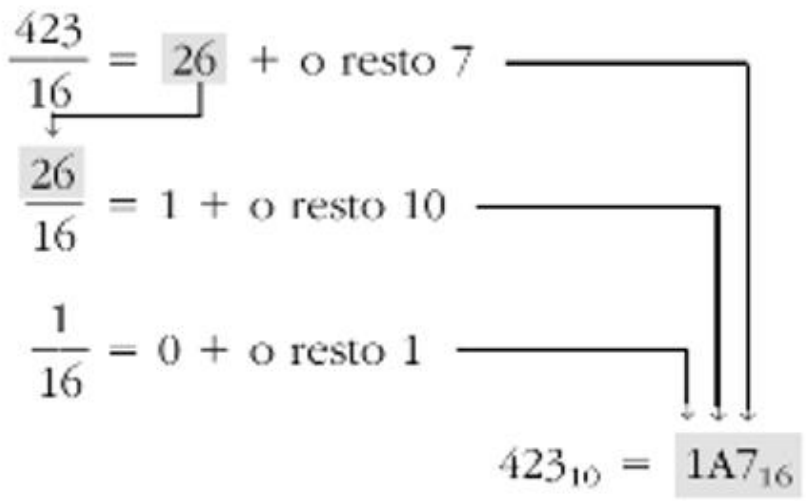
(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$

(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$
$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$
$= 45.056 + 512 + 240 + 8 = \mathbf{45.816}_{10}$

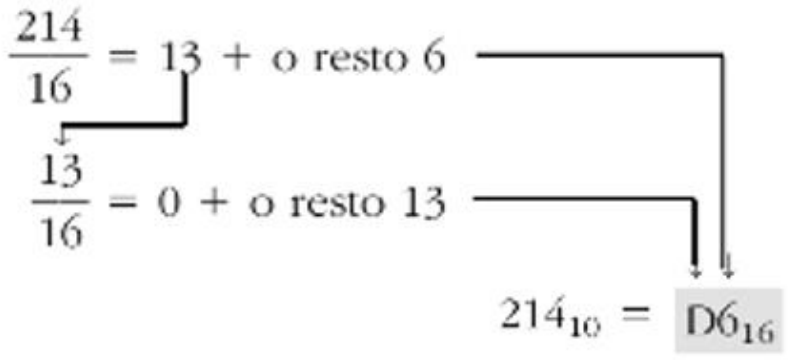
Conversão Decimal para hexadecimal

A lógica de conversão é semelhante à conversão de decimal para binário, ou seja, divisões sucessivas por 16. Sem muito segredo! Veja o exemplo abaixo:

(a) Converta 423_{10} para hexa.



(b) Converta 214_{10} para hexa.



Indicação de Vídeo

Agora assista a esses vídeos, onde é bem discutido sobre transformação de bases. Vale a pena assistir.

Disponível em: <https://www.youtube.com/watch?v=eD56zn5kYfU>. Acesso em: 20.04.2020. - Números Binários - Bases Numéricas #02 – 16 min.

Disponível em: <https://www.youtube.com/watch?v=gIXiFhEA-Qw>. Acesso em: 20.04.2020. - Números Octais e Hexadecimais - Bases Numéricas #03 – 24 min.

Videoaula 3

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda os assuntos acima estudados.

Indicação de Leitura

Para que o aprendizado seja efetivo e estejamos preparados para avaliações, leia o

Capítulo 2 do livro de Ronald Tocci. Nesse capítulo você irá complementar seus estudos.

Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/740/pdf/0?code=AzEak3e+GeKckigIt9WnVee4qLWnw19lRXWuKy+SRDrAYvBuT59ZoqYRnfRoQcmVUtMRn2ljodbfSl72bobcKQ==. Acesso em: 20.04.2020.

Aula 2 Aritmética e Códigos Digitais

Na aula anterior estudamos como converter números de uma base para outra. Nessa aula iremos nos concentrar nos princípios básicos necessários para entender como as máquinas digitais realizam as operações aritméticas básicas, em especial a adição e a subtração. Para um conhecimento ampliado de outras operações, você, amigo(a) aluno(a), poderá ler nossa indicação de leitura.

Adição Binária

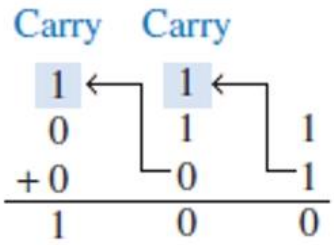
Em geral, a adição binária segue as mesmas regras da adição decimal, no entanto apenas quatro casos podem ocorrer na soma de dígitos binários em qualquer posição.

$0 + 0 = 0$	Resultado: 0; carry: 0
$0 + 1 = 1$	Resultado: 1; carry: 0
$1 + 0 = 1$	Resultado: 1; carry: 0
$1 + 1 = 10$	Resultado: 0; carry: 1

Atenção:

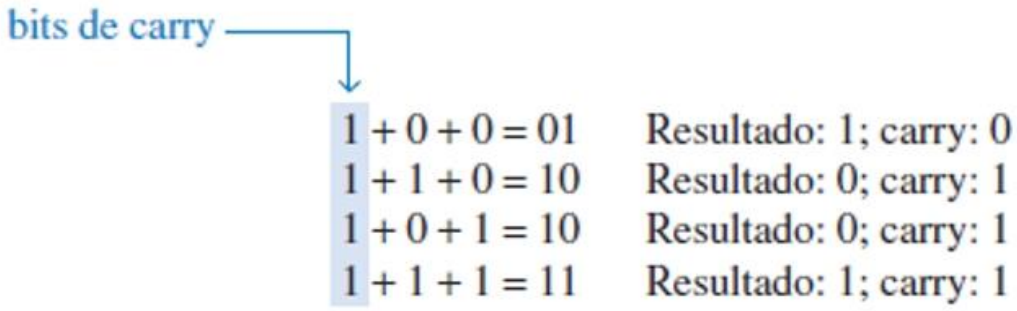
Carry *é uma palavra muito usada no estudo de sistemas digitais. Essa palavra equivale a "vai um".*

Lembre-se de que em binário 1+1=10 e não 2. Vamos a um exemplo:



Na coluna da direita, 1 + 1 = 0 com um *carry* de 1 para a próxima coluna à esquerda. Na coluna do meio, 1 + 1 + 0 = 0 com um *carry* de 1 para a próxima coluna à esquerda. Na última coluna, 1 + 0 + 0 = 1.

Quando existe um *carry* de 1, temos uma situação na qual três bits estão sendo somados (um bit de cada um dos dois números e um bit de *carry*), Veja o exemplo de como fica.



Subtração Binária

As regras para subtração binária segue as seguintes regras:

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$10 - 1 = 1$ sendo o empréstimo igual a 1

Veja um exemplo:

Efetue a subtração de 011 a partir de 101.

Solução

101	5
−011	−3
010	2

Não devemos esquecer, subtrai-se as colunas da direita para a esquerda, tal como uma subtração decimal.

Curiosidades!

Como é impossível tirar 1 de 0, o artifício é “pedir emprestado” 1 da casa de ordem superior, ou seja, na realidade o que se faz é subtrair 1_2 de 10_2 e encontramos 1_2 como resultado, devendo então subtrair 1 do dígito de ordem superior. Este algoritmo é exatamente o mesmo da subtração em decimal.

Indicação de Vídeo

Para fixar um pouco mais sobre a aritmética básica de computadores assista o vídeo sobre

Soma e Subtração Binária. Segue o link abaixo:

Disponível em: <http://youtu.be/NeQBC9Z5FHk>. Acesso em: 20.04.2020. 8:14 min.

Videoaula 1

Utilize o QRcode para assistir!

Agora, assista à Videoaula.

Aritmética do complemento de 2

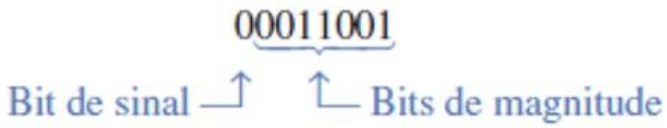
Como visto no tópico anterior, estudamos adição e subtração para números não sinalizados, porém, sabemos que os números são sinalizados, ou seja, podem ser positivos ou negativos. O sistema do complemento de 2 é o sistema mais usado para representar números binários com sinal, pois com essa técnica as operações aritméticas podem ser feitas apenas por um circuito.

Antes de aprender mais sobre esse sistema, é primordial aprender sobre o que é complemento de 1, complemento de 2 de um número binário e sinal- Magnitude:

Sinal magnitude:

> Quando um número binário sinalizado é representado na forma sinal-magnitude, o bit mais à esquerda é o bit de sinal e os bits restantes são os bits de magnitude. Os bits de magnitude estão na forma de binário verdadeiro (não-complementado) tanto para números positivos quanto para negativos.” (FLOYD,2007, p.78)

Por exemplo, o número decimal +25 é expresso como um número binário sinalizado de 8 bits usando a forma sinal-magnitude como a seguir:



O número decimal –25 é expresso como

10011001

Então, como pode perceber na forma sinal-magnitude, um número negativo tem os mesmos bits de magnitude que o número positivo correspondente, mas o bit de sinal é 1 em vez de zero.

Forma de complemento de 1: complemento de 1 de um número binário é definido trocando-se todos 0´s por 1, e todos 1´s por 0's. Veja o exemplo:

10110 >> 01001.

Forma de complemento de 2: O complemento de 2 de um número binário é definido somando-se 1 ao bit mais à direita (LSB) do complemento de 1. Veja o exemplo:

Determine o complemento de 2 de 10110010.

10110010	Número binário
01001101	Complemento de 1
+ 1	Soma-se 1
01001110	Complemento de 2

Então, o sistema de complemento de 2 para representação de números sinalizados funciona seguindo as seguintes regras (TOCCI,2008):

- Se o número for positivo, a magnitude é representada pela forma binária direta, e um bit de que representa o sinal é colocado em frente ao bit mais significativo (MSB).
- Se o número for negativo, a magnitude é representada na forma do complemento de 2, e um bit de sinal 1 é colocado em frente ao bit mais significativo (MSB).

Vamos lá! Chega de regras. Vamos apresentar casos de adição e verificar como é feito os cálculos pelas maiorias das máquinas. Para isso, iremos usar exemplos do renomado Tocci (2008, p.253):

Caso I: Dois números positivos. A adição de dois números positivos é feita diretamente. Considere a adição de +9 com +4:

```
+9  →  0  1001   (1ª parcela)
+4  →  0  0100   (2ª parcela)
       0  1101   (soma = +13)
       ↑
       └── bits de sinal
```

Observe que os bits de sinal da **1ª parcela** e da **2ª parcela** são ambos 0, e o bit de sinal da soma é 0, indicando que é positiva. Observe também que a primeira e a segunda parcelas têm a mesma quantidade de bits. Isso *sempre* tem de ser feito no sistema do complemento de 2.

Caso II: Um número positivo e outro menor e negativo. Considere a adição de +9 com –4. Lembre-se de que –4 será representado na forma do complemento de 2. Assim, +4 (00100) tem de ser convertido para –4 (11100).

```
                 bits de sinal
                 ↓
+9  →  0  1001   (1ª parcela)
–4  →  1  1100   (2ª parcela)
   1   0  0101
   ↑
   └── Este carry é desconsiderado; o resultado é 00101 (soma = +5).
```

Nesse caso, o bit de sinal da segunda parcela é 1. Observe que ele também participa do processo de soma. Na verdade, um carry é gerado na última posição da soma. *Esse carry sempre é desconsiderado*; logo, a soma final é 00101, equivalente a +5.

Caso III: Um número positivo e outro maior e negativo. Considere a adição de –9 com +4:

```
–9  →  10111
+4  →  00100
       11011   (soma = –5)
       ↑
       └── bit de sinal negativo
```

Aqui, a soma tem um bit de sinal 1, indicando um número negativo. Como a soma é negativa, está na forma do complemento de 2; logo, os últimos quatro bits, 1011, de fato representam o complemento de 2 da soma. Para obter a magnitude direta da soma, temos de fazer a negação (complemento de 2) de 11011; o resultado é 00101 = +5. Assim, 11011 representa –5.

Caso IV: Dois números negativos

```
–9  →  10111
–4  →  11100
   1   10011
   ↑   ↑
   │   └── bit de sinal
   └────── Este carry é desconsiderado; o resultado é 10011 (soma = –13).
```

O resultado final é novamente negativo e está na forma do complemento de 2 com um bit de sinal 1. Efetuando a negação (complemento de 2), temos como resultado 01101 = +13.

Caso V: Números iguais e de sinais opostos

$$\begin{array}{rcl} -9 & \rightarrow & 10111 \\ +9 & \rightarrow & 01001 \\ \hline 0 & 1 & 00000 \end{array}$$

Este carry é desconsiderado; o resultado é 00000 (soma = +0).

Obviamente, o resultado é +0, conforme o esperado.

Como você pode notar, querido(a) aluno(a), utilizando-se o sistema de complemento de 2 conseguimos desenvolver tanto a adição como a subtração.

Esse assunto é muito interessante, por isso, é de essencial importância ler a indicação de leitura obrigatória para seu aprendizado.

Videoaula 2

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda os assuntos acima estudados.

Indicação de Leitura

Para um aprofundamento do nosso assunto leia o livro de **Ronald J.Tocci**, capítulo 6. Dos itens 6.1 a 6.6.

Disponível em: <https://plataforma.bvirtual.com.br/Acervo/Publicacao/740>. Acesso em: 20.04.2020.

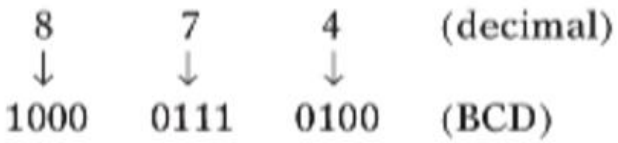
Códigos - Decimal codificado em binário (BCD) e ASCII

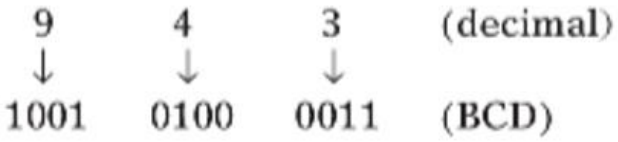
Todo sistema digital usa de alguma forma a numeração binária para fazer suas operações internas. O mundo analógico é um mundo decimal e por isso conversões entre sistemas decimal e binário são feitas com frequência.

Como notaram, as conversões entre decimal e binário podem se tornar longas e computacionalmente complicadas para números grandes. Então, muitas vezes utiliza-se códigos que contenham características decimais e binárias, que é o caso do código BCD (Decimal codificado em binário).

O BCD é um código onde cada dígito decimal, de 0 a 9, é representado por um código binário de 4 bits. Esse código tem como característica a facilidade da conversão entre decimal e binário.

Vamos ilustrar com um exemplo o uso desse código para ficar mais claro para você:





Como pode ser notado no exemplo anterior, cada dígito decimal é convertido em seu binário puro, sem fazer conversões. São usados apenas os binários de 4 bits entre 0000 e 1001. O código BCD não usa os números 1010, 1011,1100,1101,1110 e 1111.

Vamos para mais alguns exemplos e veja como é fácil essa conversão. Exemplos retirados de Floyd (2007).

Converta em BCD cada um dos seguintes números decimais:

(a) 35 (b) 98 (c) 170 (d) 2469

(a) 3 5 → 00110101

(b) 9 8 → 10011000

(c) 1 7 0 → 000101110000

(d) 2 4 6 9 → 0010010001101001

Converta cada um dos seguintes códigos BCD em decimal:

(a) 10000110 (b) 001101010001 (c) 1001010001110000

(a) 10000110 → 8 6

(b) 001101010001 → 3 5 1

(c) 1001010001110000 → 9 4 7 0

Uma das aplicações mais conhecidas do código BCD, é o seu uso em ***displays*** de 7 segmentos, que são usados em vários projetos de automação.

Vale a pena ressaltar, amigo(a) aluno(a), que o código BCD não é um **sistema de numeração**, como o hexadecimal, binário e decimal. BCD é um sistema digital, no qual cada digito é codificado no seu respectivo binário. Apenas para exemplificar essa diferenciação veja o exemplo abaixo. Exemplo pesquisado em Tocci (2008).

$137_{10} = 10001001_2$ (binário)

$137_{10} = 0001\ 0011\ 0111$ (BCD)

Código ASCII

Um computador não se limita apenas a manipular dados numéricos. Na verdade, essa máquina precisa ser capaz também de manipular dados não numéricos, pois o ser humano, necessita de letras e símbolos para se comunicar também.

Um código alfanumérico simples precisa de no mínimo de 10 dígitos decimais e 26 letras do alfabeto, acumulando-se assim 36 itens. Esse número de itens necessita de 6 bits em cada representação, dado que 5 bits não é o suficiente (2^5=32). Evidentemente sabemos que precisamos de mais símbolos para se ter uma comunicação completa. Precisamos de pontos,

espaços, virgulas, etc., portanto um código alfanumérico deve representar todos os caracteres encontrados em um teclado de computador. Vamos estudar agora o código alfanumérico mais utilizado mundialmente: o código ASCII.

ASCII é a abreviação *de American Standard Code for Information Interchange* (Código Padrão Americano para Troca de Informações). Esse código é aceito universalmente e usado na maioria dos computadores e aparelhos eletrônicos mundialmente.

Então, querido(a) estudante, o seu teclado, com certeza, tem o padrão ASCII. Esse código tem 128 caracteres representado por um código de 7 bits (2^7=128). Os primeiros 32 caracteres ASCII são comandos não gráficos, pois não são impressos ou mostrados e são usados para fins de controle (FLOYD,2007). São exemplos desses caracteres “próxima linha”, “inicio de texto” e “escape”. Os demais símbolos gráficos são os que podem ser impressos e incluem letras alfabeto (minúsculas e maiúsculas), os dez dígitos decimais, sinais de pontuação e outros símbolos normalmente usados.

Curiosidades!

O teclado de um computador tem um microprocessador dedicado que constantemente escaneia (“lê”) o circuito do teclado para detectar quando uma tecla foi pressionada e liberada. Uma única varredura é gerada pelo software do computador representando aquela tecla em particular. O código de varredura é então convertido em código alfanumérico (ASCII) para ser usado pelo computador (FLOYD,2007).

A tabela 01, mostra uma lista do código ASCII mostrando a representação de cada caractere e símbolo em decimal, hexadecimal e binário. A seção à esquerda da tabela é uma lista dos nomes dos 32 caracteres de controle (de 00 a 1F em hexadecimal). Os símbolos gráficos são apresentados no restante da tabela (de 20 a 7F em hexadecimal) (FLOYD, 2007).

CARACTERES DE CONTROLE				SÍMBOLO GRÁFICO											
NOME	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA
NUL	0	0000000	00	espaço	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	–	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

Tabela 01. Código Padrão Americano para Troca de Informações -ASCII.

Fonte: FLOYD (2007, p.107).

Sabemos agora que o código ASCII pode ser usado para fazer transferência alfanumérica entre computador e equipamentos externos, como por exemplo, a impressora. Mas também pode ser usado internamente para armazenar informações digitadas pelo operador de computador. Vamos ilustrar o uso do código ASCII, com um exemplo retirado de Floyd (2007). Observe abaixo:

Determine o código binário ASCII que é inserido pelo teclado do computador quando a seguinte linha de comando em BASIC é digitada. Expresse também cada código em hexadecimal.

20 PRINTI "A=";X

O código ASCII para cada símbolo é encontrado na Tabela 2–7.

Símbolo	Binário	Hexadecimal
2	0110010	32_{16}
0	0110000	30_{16}
Espaço	0100000	20_{16}
P	1010000	50_{16}
R	1010010	52_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
T	1010100	54_{16}
Espaço	0100000	20_{16}
"	0100010	22_{16}
A	1000001	41_{16}
=	0111101	$3D_{16}$
"	0100010	22_{16}
;	0111011	$3B_{16}$
X	1011000	58_{16}

Também utilizando outra linguagem de alto nível, como a linguagem C, podemos extrair a tabela ASCII. Observe o resultado das linhas de comandos em C abaixo. É gerado o seu número decimal seguido de seu símbolo em ASCII:

```
main() {
int c;
    for(c=32; c<255; c++) {
        if(c%10==0)
            printf("\n");
            printf("%d = %c\t",c ,c);
}}
```

32 =	33 = !	34 = "	35 = #	36 = $	37 = %	38 = &	39 = '												
40 = (	41 =)	42 = *	43 = +	44 = ,	45 = -	46 = .	47 = /	48 = 0	49 = 1										
50 = 2	51 = 3	52 = 4	53 = 5	54 = 6	55 = 7	56 = 8	57 = 9	58 = :	59 = ;										
60 = <	61 = =	62 = >	63 = ?	64 = @	65 = A	66 = B	67 = C	68 = D	69 = E										
70 = F	71 = G	72 = H	73 = I	74 = J	75 = K	76 = L	77 = M	78 = N	79 = O										
80 = P	81 = Q	82 = R	83 = S	84 = T	85 = U	86 = V	87 = W	88 = X	89 = Y										
90 = Z	91 = [	92 = \	93 =]	94 = ^	95 = _	96 = `	97 = a	98 = b	99 = c										
100 = d	101 = e	102 = f	103 = g	104 = h	105 = i	106 = j	107 = k	108 = l	109 = m										
110 = n	111 = o	112 = p	113 = q	114 = r	115 = s	116 = t	117 = u	118 = v	119 = w										
120 = x	121 = y	122 = z	123 = {	124 = |	125 = }	126 = ~	127 = ⌂	128 = Ç	129 = ü										
130 = é	131 = â	132 = ä	133 = à	134 = å	135 = ç	136 = ê	137 = ë	138 = è	139 = ï										
140 = î	141 = ì	142 = Ä	143 = Å	144 = É	145 = æ	146 = Æ	147 = ô	148 = ö	149 = ò										
150 = û	151 = ù	152 = ÿ	153 = Ö	154 = Ü	155 = ø	156 = £	157 = Ø	158 = ×	159 = ƒ										
160 = á	161 = í	162 = ó	163 = ú	164 = ñ	165 = Ñ	166 = ª	167 = º	168 = ¿	169 = ®										
170 = ¬	171 = ½	172 = ¼	173 = ¡	174 = «	175 = »	176 = ▒	177 = ▒	178 = ▓	179 = |									
180 = ┤	181 = Á	182 = Â	183 = À	184 = ©	185 = ╣	186 = ╠	187 = ╗	188 = ╝	189 = ¢										
190 = ¥	191 = ┐	192 = └	193 = ┴	194 = ┬	195 = ├	196 = ─	197 = ┼	198 = ã	199 = Ã										
200 = ╚	201 = ╔	202 = ╩	203 = ╦	204 = ╠	205 = ═	206 = ╬	207 = ¤	208 = ð	209 = Ð										
210 = Ê	211 = Ë	212 = È	213 = ı	214 = Í	215 = Î	216 = Ï	217 = ┘	218 = ┌	219 = ▒										
220 = ▄	221 = ¦	222 = Ì	223 = ▀	224 = Ó	225 = ß	226 = Ô	227 = Ò	228 = õ	229 = Õ										
230 = µ	231 = þ	232 = Þ	233 = Ú	234 = Û	235 = Ù	236 = ý	237 = Ý	238 = ¯	239 = ´										
240 = -	241 = ±	242 = _	243 = ¾	244 = ¶	245 = §	246 = ÷	247 = ¸	248 = °	249 = ¨										
250 = ·	251 = ¹	252 = ³	253 = ²	254 = ■															

Videoaula 3

Utilize o QRcode para assistir!

Agora, assista ao vídeo que aborda os assuntos acima estudados.

Encerramento

Chegamos ao fim de mais uma unidade, estimado(a) aluno(a). Espero que tenha gostado do texto que trouxemos para você. Nessa unidade mostramos um pouco de como o computador representa as informações. Em outras palavras, como o computador processa determinadas informações.

Na aula 1, você estudou que os sistemas computacionais lidam com a informação de maneira matemática, usando determinados sistemas numéricos. O principal sistema numérico, é o sistema binário. Vimos também que o mundo real é um mundo do sistema decimal, então para que o computador entenda o mundo real é necessário que haja algumas conversões numéricas. Recapitulando então, estudamos principalmente: Bases numéricas e conversões entre essas bases.

Em nossa aula 2, discutimos um pouco sobre a aritmética digital. Em especial a adição e a subtração. Como o computador é uma máquina de calcular, teríamos que ter a noção desse mecanismo. Então, foi apresentado uma técnica de como o computador faz as continhas internas, essa técnica é conhecida como complemento de 2.

Notamos também que o computador precisa de determinados códigos para se comunicar, entre esses códigos estudamos o BCD que se parece a um sistema numérico mais não é, e o conhecido ASCII. O código ASCII é código que faz a interface entre você e o seu computador. É muito importante para você, amigo(a) aluno(a), fazer as leituras obrigatórias para ter um aprofundamento dos assuntos tratados nessa unidade.

Bons estudos a todos!

Referências

FLOYD, Thomas L**. Sistemas digitais:** fundamentos e aplicações. 9. ed. Porto Alegre, RS: Bookman, 2007. 888 p. ISBN 978-85-60031-93-1**.**

TOCCI, Ronald J.; WIDNER, Neal S.; MOSS, Gregory L. **Sistemas digitais:** princípios e aplicações. 10. ed. São Paulo: Pearson, 2008. 804 p. ISBN 978-85-7605-095-7.

NULL, Linda; LISBÔA, Maria Lúcia B.; LISBÔA, Carlos Arthur L. (Rev. téc.). **Princípios básicos de arquitetura e organização de computadores.** 2. ed. Porto Alegre: Bookman, 2010. 821 p. ISBN 978-85-7780-737-6 – Acervo físico da Universidade.

Esperamos que este guia o tenha ajudado compreender a organização e o funcionamento de seu curso. Outras questões importantes relacionadas ao curso serão disponibilizadas pela coordenação.

Grande abraço e sucesso!