2023

# 9. Binário

R2: SCRAPY Guide

Número do projeto: **2021-1-FR01-KA220-SCH-000031617**

Co-funded by the European Union



ECAM EPMI

30/04/2023

# Índice



*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

# 1 Introdução

Os sistemas numéricos são os métodos que usamos para representar números. Desde a escola primária, todos nós temos operado principalmente dentro dos limites confortáveis de um sistema de números de base 10, mas há muitos outros. Base-2, base-8, base-16, base-20, base... você entende o ponto. Há uma variedade infinita de sistemas de números de base por aí, mas apenas alguns são especialmente importantes para a engenharia elétrica.

Os sistemas de números realmente populares até têm o seu nome. Base-10, por exemplo, é comumente referido como o sistema de números decimais. Base-2, sobre a qual estamos aqui para falar hoje, também atende pelo apelido de binário. Outro sistema numérico popular, base-16, é chamado hexadecimal.

A base de um número é frequentemente representada por um inteiro subscrito à direita de um valor. Assim, na introdução acima, a primeira imagem seria 10010 algos, enquanto a segunda imagem seria 1002 algos. Esta é uma maneira prática de especificar a base de um número quando há qualquer possibilidade de ambiguidade.

# 2. Porquê Binary?

Por que binário você pergunta? Bem, porquê decimal? Temos usado decimais desde sempre e, na maioria das vezes, tomamos como certa a razão pela qual estabelecemos o sistema de números de base 10 para nossas necessidades diárias de números. É porque temos 10 dedos, ou é apenas porque os romanos o forçaram sobre seus antigos subjugos. Independentemente do que levou a isso, truques que aprendemos ao longo do caminho solidificaram o lugar da base-10 em nossos corações; todos podem contar por 10s. Estamos mesmo arredondando grandes números para o múltiplo mais próximo de 10. Somos obcecados por 10!

Computadores e eletrônicos são limitados no departamento de dedos das mãos e dos pés. No nível mais baixo, eles só têm duas maneiras de representar o estado de qualquer coisa: ON ou OFF, alto ou baixo, 1 ou 0. E assim, todos os eletrônicos dependem de um sistema de números de base 2 para armazenar, manipular e calcular números.

A grande dependência que a eletrônica deposita em números binários significa que é importante saber como funciona o sistema de números de base 2. Você normalmente encontrará binários, ou seus primos, como hexadecimais, em todos os programas de computador. A análise de circuitos lógicos digitais e outros eletrônicos de nível muito baixo também requer uso pesado de binário.

Nesta lição, você descobrirá que qualquer coisa que você possa fazer com um número decimal também pode ser feita com um número binário. Algumas operações podem ser ainda mais fáceis de fazer em um número binário (embora outras possam ser mais dolorosas). Vamos abordar tudo isso e muito mais nesta lição.

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

# 3 Contagem e conversão

A base de cada sistema numérico também é chamada de radix. O radix de um número decimal é dez e o radix de binário é dois. O radix determina quantos símbolos diferentes são necessários para dar corpo a um sistema numérico. Em nosso sistema de números decimais, temos 10 representações numéricas para valores entre nada e dez algo: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. Cada um desses símbolos representa um valor muito específico e padronizado.

Em binário, só nos são permitidos dois símbolos: 0 e 1. Mas usando esses dois símbolos, podemos criar qualquer número que um sistema decimal pode.

## 3.1 Contagem em binário

Você pode contar em decimais infinitamente, mesmo durante o sono, mas como você contaria em binário? Zero e um na base dois devem parecer bastante familiares: 0 e 1. A partir daí, as coisas ficam decididamente binárias.

Lembre-se de que só temos esses dois dígitos, então como fazemos em decimal quando ficamos sem símbolos, temos que deslocar uma coluna para a esquerda, adicionar um 1 e virar todos os dígitos à direita para 0. Então, depois de 1 temos 10, depois 11, depois 100. Vamos começar a contar...

| Decimal | Binário | ... | Decimal | Binário |
|---|---|---|---|---|
| 0 | 0 | | 16 | 10000 |
| 1 | 1 | | 17 | 10001 |
| 2 | 10 | | 18 | 10010 |
| 3 | 11 | | 19 | 10011 |
| 4 | 100 | | 20 | 10100 |
| 5 | 101 | | 21 | 10101 |
| 6 | 110 | | 22 | 10110 |
| 7 | 111 | | 23 | 10111 |
| 8 | 1000 | | 24 | 11000 |
| 9 | 1001 | | 25 | 11001 |
| 10 | 1010 | | 26 | 11010 |
| 11 | 1011 | | 27 | 11011 |
| 12 | 1100 | | 28 | 11100 |
| 13 | 1101 | | 29 | 11101 |
| 14 | 1110 | | 30 | 11110 |
| 15 | 1111 | | 31 | 11111 |

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

Isso começa a pintar o quadro? Vamos examinar como podemos converter desses números binários para decimais.

## 3.2 Convertendo binário em decimal

Não há uma maneira de converter binário para decimal. Vamos descrever dois métodos abaixo, o método mais "matemático" e outro que é mais visual. Vamos abordar ambos, mas se o primeiro usar muita terminologia feia, pule para o segundo.

### Método 1

Há uma função útil que podemos usar para converter qualquer número binário em decimal:

$$a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_1 2^1 + a_0 2^0$$

Há quatro elementos importantes nessa equação:

an, an-1, a1, etc., são os dígitos de um número. Estes são os 0 e 1 com os quais você está familiarizado, mas em binário, eles só podem ser 0 ou 1.

A posição de um dígito também é importante observar. A posição começa em 0, no dígito mais à direita; este 1 ou 0 é o menos significativo. Cada dígito que você move para a esquerda aumenta em significância e também aumenta a posição em 1.

O comprimento de um número binário é dado pelo valor de n, na verdade, é n+1. Por exemplo, um número binário como 101 tem um comprimento de 3, e algo maior, como 10011110 tem um comprimento de 8.

Cada dígito é multiplicado por um peso: o 2n, 2n-1, 21, etc. O peso mais à direita - 20 equivale a 1, mova um dígito para a esquerda e o peso torna-se 2, depois 4, 8, 16, 32, 64, 128, 256,... e assim por diante. Poderes de dois são de grande importância para binário, eles rapidamente se tornam muito familiares.

Vamos nos livrar desses n e expoentes, e realizar nossa equação de notação posicional binária em oito posições:

$$a_7 \cdot 128 + a_6 \cdot 64 + a_5 \cdot 32 + a_4 \cdot 16 + a_3 \cdot 8 + a_2 \cdot 4 + a_1 \cdot 2 + a_0 \cdot 1$$

Levando isso adiante, vamos conectar alguns valores para os dígitos. E se você tivesse um número binário como 10011011? Isso significaria (um) valores de:

| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $a_7$ | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |

### Método 2

Outra maneira mais visual de converter números binários em decimais é começar classificando cada 1 e 0 em uma lixeira. Cada caixote tem uma potência sucessiva de dois pesos, o 1, 2, 4, 8, 16,... estamos acostumados a isso. Levá-lo para oito lugares seria mais ou menos assim:

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|

Então, se classificarmos nosso número binário 10011011 nesses compartimentos, ele ficaria assim:

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

Para cada compartimento que tenha um valor binário 0, basta riscar e removê-lo.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

E então adicione os pesos restantes para obter o seu número!

## 3.3 Conversão de decimal para binário

Assim como passar de binário para decimal, há mais de uma maneira de converter decimal em binário. O primeiro usa divisão e remanescentes, e o segundo usa subtração. Experimente ambos e mantenha um com o qual se sinta confortável!

### Método 1

Não é tão simples converter um número decimal em binário. Essa conversão requer a divisão repetida do número decimal por 2, até que você o reduza a zero. Toda vez que você divide o restante da divisão se torna um dígito no número binário que você está criando.

Não se lembra como fazer restos? Se já faz algum tempo, lembre-se que, como estamos dividindo por dois, se o dividendo for par, o restante será 0; um dividendo ímpar significa um restante de 1.

Por exemplo, para converter 155 em binário, você passaria por este processo:

***155 ÷ 2 = 77 R 1 (Esse é o dígito mais à direita, 1ª posição)***
***77 ÷ 2 = 38 R 1 (2ª posição)***
***38 ÷ 2 = 19 R 0 (3ª posição)***
***19 ÷ 2 = 9 R 1***
***9 ÷ 2 = 4 R 1***
***4 ÷ 2 = 2 R 0***
***2 ÷ 2 = 1 R 0***
***1 ÷ 2 = 0 R 1 (8ª posição)***

O primeiro restante é o dígito menos significativo (mais à direita), então leia de cima para baixo para detalhar nosso número binário da direita para a esquerda: 10011011. Combine-o com o exemplo acima... isso é um bingo!

### Método 2

Se dividir e encontrar restos não é a sua praia, pode haver um método mais fácil para converter decimal em binário. Comece por encontrar a maior potência de dois que ainda seja menor do que o seu número decimal e subtraia-o do decimal. Em seguida, continue

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

um subtrair pela maior potência possível de dois até chegar a zero. Cada posição de peso que foi subtraída, recebe um binário de 1 dígito; os dígitos que não foram subtraídos obtêm um 0.

Continuando com o nosso exemplo, 155 pode ser subtraído por 128, produzindo 27:

155 - 128 = 27

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |

Nosso novo número, 27, não pode ser subtraído por 64 ou 32. Ambas as posições recebem um 0. Podemos subtrair por 16, produzindo 11.

27 - 16 = 11

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | | | | |

E 8 subtrai de 11, produzindo 3. Depois disso, não teve essa sorte com 4.

11 - 8 = 3

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | | |

Nossos 3 podem ser subtraídos por 2, produzindo 1. E, finalmente, o 1 subtrai por 1 para fazer 0.

3 - 2 = 1
1 - 1 = 0

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

Temos um número binário!

### Bits, Nibbles e Bytes

Ao discutir a marca de um número binário, cobrimos brevemente o comprimento do número. O comprimento de um número binário é a quantidade de 1's e 0's que ele tem.

## 4 Comprimentos de números binários comuns

Os valores binários são frequentemente agrupados em um comprimento comum de 1's e 0's, este número de dígitos é chamado de comprimento de um número. Comprimentos de bits comuns de números binários incluem bits, petiscos e bytes (com fome ainda?). Cada 1 ou 0 em um número binário é chamado de **bit**. A partir daí, um grupo de 4 bits é chamado de **nibble**, e 8-bits fazem um **byte**.

Bytes são uma palavra da moda bastante comum quando se trabalha em binário. Os processadores são todos construídos para trabalhar com um comprimento definido de bits, que geralmente é este comprimento é um múltiplo de um byte: 8, 16, 32, 64, etc.

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

To sum it up:

| Comprimento | Nome | Exemplo |
|---|---|---|
| 1 | Bit | 0 |
| 4 | Nibble | 1011 |
| 8 | Byte | 10110101 |

**Palavra** é outro chavão de comprimento que é jogado fora de vez em quando. A palavra é muito menos deliciosa e muito mais ambígua. O comprimento de uma palavra geralmente depende da arquitetura de um processador. Pode ser 16 bits, 32, 64 ou até mais.

## 5. Preenchimento com zeros à esquerda

Você pode ver valores binários representados em bytes (ou mais), mesmo que fazer um número de 8 bits de comprimento exija a adição de zeros à esquerda. Zeros à esquerda são um ou mais 0's adicionados à esquerda do 1-bit mais significativo em um número. Você geralmente não vê zeros à esquerda em um número decimal: 007 não diz mais sobre o valor do número 7 (pode dizer outra coisa).

Os zeros à esquerda não são necessários em valores binários, mas ajudam a apresentar informações sobre o comprimento de bits de um número. Por exemplo, você pode ver o número 1 impresso como 00000001, apenas para dizer que estamos trabalhando dentro do domínio de um byte. Ambos os números representam o mesmo valor, no entanto, o número com sete 0's na frente adiciona informações sobre o comprimento de bit de um valor.

## 6 Operadores Bitwise

Há várias maneiras de manipular valores binários. Assim como você pode com números decimais, você pode executar operações matemáticas padrão - adição, subtração, multiplicação e divisão - em valores binários (que abordaremos na próxima página). Você também pode manipular bits individuais de um valor binário usando operadores bitwise.

Os operadores Bitwise executam funções bit a bit em um ou dois números binários completos. Eles fazem uso da lógica booleana operando em um grupo de símbolos binários. Esses operadores bitwise são amplamente utilizados em toda a eletrônica e programação.

## 7 Complemento (NÃO)

O complemento de um valor binário é como encontrar o exato oposto de tudo sobre ele. A função de complemento olha para um número e transforma cada 1 em um 0 e cada 0 se torna um 1. O operador do complemento também é chamado de NOT.

Por exemplo, para encontrar o complemento de 10110101:

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

NOT 10110101 (decimal 181)
-------- =
01001010 (decimal 74)

NOT é o único operador bit a bit que opera apenas em um único valor binário.

## 8 OR

OR pega dois números e produz a união deles. Aqui está o processo para OU dois números binários juntos: alinhe cada número para que os bits correspondam e, em seguida, compare cada um dos bits que compartilham uma posição. Para cada comparação de bits, se um ou ambos os bits forem 1, o valor do resultado nessa posição de bit será 1. Se ambos os valores tiverem um 0 nessa posição, o resultado também obterá um 0 nessa posição.

As quatro combinações possíveis de RUP e o seu resultado são:

- 0 OR 0 = 0
- 0 OR 1 = 1
- 1 OR 0 = 1
- 1 OR 1 = 1

Por exemplo, para encontrar o 10011010 OU 01000110, alinhe cada um dos números pouco a pouco. Se um ou ambos os números tiverem um 1 em uma coluna, o valor do resultado também terá um 1:

10011010
OR 01000110
-------- =
11011110

Pense na operação OR como adição binária, sem uma transferência. 0 mais 0 é 0, mas 1 mais qualquer coisa será 1.

## 9 AND

E pega dois números e produz a conjunção deles. E só produzirá um 1 se ambos os valores em que está operando também forem 1.

O processo de AND'ing dois valores binários juntos é semelhante ao de OR. Alinhe cada número para que os bits correspondam e, em seguida, compare cada um dos bits que partilham uma posição. Para cada comparação de bits, se um ou ambos os bits forem 0, o valor do resultado nessa posição de bit será 0. Se ambos os valores tiverem um 1 nessa posição, o resultado também obtém um 1 nessa posição.

As quatro combinações AND possíveis e o seu resultado são:

- 0 AND 0 = 0

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

- 0 AND 1 = 0
- 1 AND 0 = 0
- 1 AND 1 = 1

Por exemplo, para encontrar o valor de 10011010 E 01000110, comece alinhando cada valor. O resultado de cada posição de bit será apenas 1 se ambos os bits nessa coluna também forem 1.

10011010
AND 01000110
-------- =
00000010

Pense em E como multiplicação. Sempre que multiplicar por 0 o resultado também será 0.

## 10 XOR

XOR é o exclusivo OR. XOR se comporta como OR regular, exceto que só produzirá um 1 se um ou outro número tiver um 1 nessa posição de bits.

As quatro combinações XOR possíveis e o seu resultado são:

- 0 XOR 0 = 0
- 0 XOR 1 = 1
- 1 XOR 0 = 1
- 1 XOR 1 = 0

Por exemplo, para localizar o resultado de 10011010 XOR 01000110:

10011010
XOR 01000110
-------- =
11011100

Observe o 2º bit, um 0 resultante de dois 1's XOR'ed juntos.

## 11 Deslocamentos de bits

Os deslocamentos de bits não são necessariamente um operador bit a bit como os listados acima, mas são uma ferramenta útil para manipular um único valor binário.

Há dois componentes para um bit shift - a direção e a quantidade de bits para mudar. Você pode deslocar um número para a esquerda ou para a direita, e você pode mudar por um bit ou muitos bits.

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*

Ao mudar para a direita, um ou mais dos bits menos significativos (no lado direito do número) simplesmente são cortados e deslocados para o infinito nada. Zeros à esquerda podem ser adicionados para manter o comprimento de bit o mesmo.

Por exemplo, deslocando 10011010 para a direita dois bits:

RIGHT-SHIFT-2 10011010 (decimal 154)
-------- =
00100110 (decimal 38)

Shifting to the left adds pushes all of the bits toward the most-significant side (the left side) of the number. For each shift, a zero is added in the least-significant-bit position.

Por exemplo, deslocando 10011010 para a esquerda um pouco:

LEFT-SHIFT-1 10011010 (decimal 154)
-------- =
100110100 (decimal 308)

Esse simples deslocamento de bits executa uma função matemática complicada. Deslocamentos para a esquerda n bits multiplicam um número por 2n (veja como o último exemplo multiplicou a entrada por dois?), enquanto um deslocamento em bits para a direita fará uma divisão inteira por 2n. Mudar para a direita para dividir pode ficar estranho - quaisquer frações produzidas pela divisão de turno serão cortadas, e é por isso que 154 deslocado para a direita duas vezes é igual a 38 em vez de 154/4=38,5. Os deslocamentos de bits podem ser uma maneira poderosamente rápida de dividir ou multiplicar por 2, 4, 8, etc.

## 12 Conclusão

O binário é o bloco de construção de todos os cálculos, cálculos e operações em eletrônica. Então, há muitos lugares para ir a partir daqui.

Agora que você pode converter entre decimal e binário, você pode aplicar esse conhecimento para entender como os caracteres são codificados universalmente: ASCII

Ou você pode aplicar seus novos conhecimentos brilhantes a circuitos de baixo nível e IC's:

- Lógica Digital
- Registos de turnos

Você também pode dar uma olhada em como o binário desempenha um papel importante nesses protocolos de comunicação:

- Comunicação Serial
- Interface Periférica Serial
- I2C

*Este projeto foi financiado com o apoio da Comissão Europeia. A presente publicação [comunicação] reflete apenas as opiniões do autor, não podendo a Comissão ser responsabilizada pela utilização que possa ser feita das informações nela contidas.*