

Sistema Binário

Objetivo da Aula

Compreender a importância da base binária para a arquitetura do computador.

Apresentação

Veremos agora como o computador armazena e processa as informações que são tão importantes para nós.

Como uma máquina que trabalha com sinais elétricos, o computador precisa “entender” aquilo que queremos e escrevemos através de programas.

Quando criamos programas, usamos linguagens de programação que se assemelham à linguagem humana. Mas como isso chega aos circuitos eletrônicos? A resposta é um sistema que vai traduzindo (compilando) de um nível mais alto até o nível mais baixo, o do computador.

Vamos agora ver como isso funciona, trabalhando com as bases numéricas.

1. Sistema Binário

Os computadores digitais, tão presentes em nosso dia a dia, trabalham em dois níveis de tensão e a forma mais adequada para representar isso é a notação ou sistema binário.

A primeira descrição conhecida de um sistema numérico binário foi apresentada pelo matemático indiano Pingala no século III a.C, representando os números de 1 a 8 com a sequência 001, 010, 011, 100, 101, 110, 111 e 1000.

Figura 1: Pingala, o “pai” da numeração binária



Foto: www.cuemath.com.

O termo **bit** é uma abreviação da expressão **binary digit**. O agrupamento de quatro bits é chamado de **nibble** e o agrupamento de oito bits é chamado de **byte**.

Um computador trabalha basicamente executando sequências de instruções, que conhecemos como programas. Essas instruções são buscadas na memória, decodificadas e executadas.

Porém, como elas e os dados ficam guardados na memória? Como o processador sabe o que elas significam?

Vamos ver como isso acontece.

Uma instrução é armazenada no sistema de memórias de uma forma que representamos utilizando zeros e uns, como a Figura 2.

Figura 2: Sequência de uma instrução em números binários

10011001101011001111010110010110

Fonte: Autor

No exemplo acima, temos uma instrução utilizando 32 bits que, para nós, pode não significar absolutamente nada a princípio, mas para o processador ela diz muito, na verdade, tudo, já que contém todas as informações de que ele precisa para executá-la.

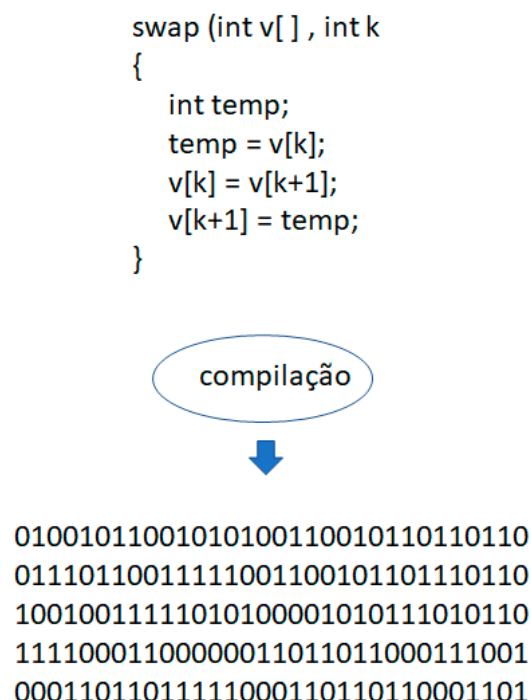
E que informações seriam essas? Por exemplo, a sequência de bits destacada em negrito, em uma instrução de um processador hipotético, pode representar o **opcode** (código de operação), que informa ao processador qual operação ele deverá executar. Esta operação pode ser uma soma, subtração, comparação, ou qualquer operação lógica, ou aritmética.

Além disso, os bits de uma instrução têm outras funções, tais como o endereço onde encontrar os dados que a instrução manipulará, o endereço onde o resultado será armazenado, bem como se a instrução trabalha com inteiros ou não inteiros, entre outras.

Todo processador tem uma espécie de vocabulário que ele consegue entender e executar e isso é chamado de **arquitetura do conjunto de instruções** (ISA – Instruction Set Architecture). Essa arquitetura define como as instruções são codificadas em binário e contém todas as instruções que o processador conseguirá executar.

Nós, humanos, temos nossa linguagem própria e as linguagens de programação se aproximam dessa linguagem. Mas os computadores não têm a capacidade de entender a linguagem humana diretamente. Tudo tem que ser “traduzido” para a linguagem que eles entendem. Esse processo é chamado de **compilação** e transforma tudo o que escrevemos em um código representado pela notação binária, conforme mostrado na Figura 3.

Figura 3: O processo de compilação



Elaborado pelo autor.

Na Figura 3 temos, na parte de cima, um programa escrito em uma linguagem de programação muito conhecida e usada por programadores. Após o processo de compilação, o programa é “traduzido” para um código em binário que o computador conseguirá interpretar e executar porque está numa linguagem adequada a ele.

Vale lembrar aqui que esses zeros e uns são apenas uma forma de representarmos os dois níveis de tensão em que os computadores trabalham, como mostramos acima.

Portanto, quando nos referimos a um programa em **linguagem de máquina**, estamos nos referindo àquele representado em código binário, ou seja, que o computador consegue entender e executar.

Importante observar que, embora seja muito difícil para um humano entender um código representado em binário, há profissionais que precisam fazer isso, já que o próprio compilador é um programa como outro qualquer e alguém tem que entender a correlação entre o código em alto nível (usado pelos programadores) e o código em baixo nível (interpretado e executado pelos computadores).

A notação binária é, como as notações decimal, octal e hexadecimal, que também estudaremos neste curso, uma notação posicional, o que significa que o valor de cada algarismo depende da sua posição no número.

Quando nos referimos à posição dos bits em um número, consideramos como bits mais significativos ou menos significativos e isso depende da posição de cada bit nesse número. Os bits mais significativos são aqueles que estão mais à esquerda e, conseqüentemente, os menos significativos estão mais à direita, conforme mostramos na Figura 4.

Figura 4: Bit mais significativo e menos significativo



Elaborado pelo autor.

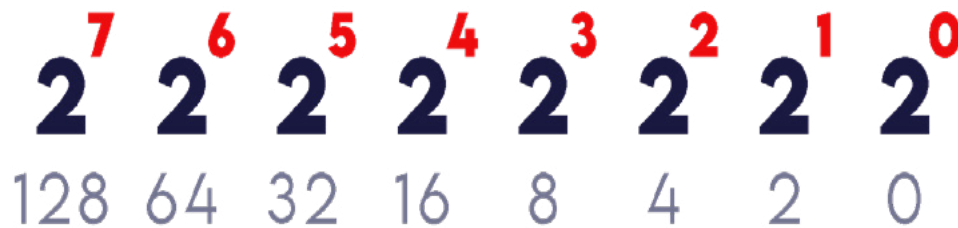
A posição de cada bit terá uma participação fundamental quando tratarmos da conversão entre bases.

Como a notação é binária (só admite dois valores) precisaremos trabalhar com as potências de 2, já que a posição de cada bit terá o peso da respectiva posição. É importante lembrar que a contagem das posições começa em zero.

Assim, o bit menos significativo será equivalente a 2^0 , o segundo bit menos significativo será equivalente a 2^1 , o terceiro bit menos significativo será equivalente a 2^2 e assim por diante.

Observe a Figura 5.

Figura 5: Os bits e seus valores posicionais



Elaborado pelo autor.

Na Figura 5 observamos o **peso** de cada bit relacionado à sua posição no conjunto e a respectiva potência de 2. Usaremos isso, na prática, quando estudarmos as conversões de base e aritmética em binário.

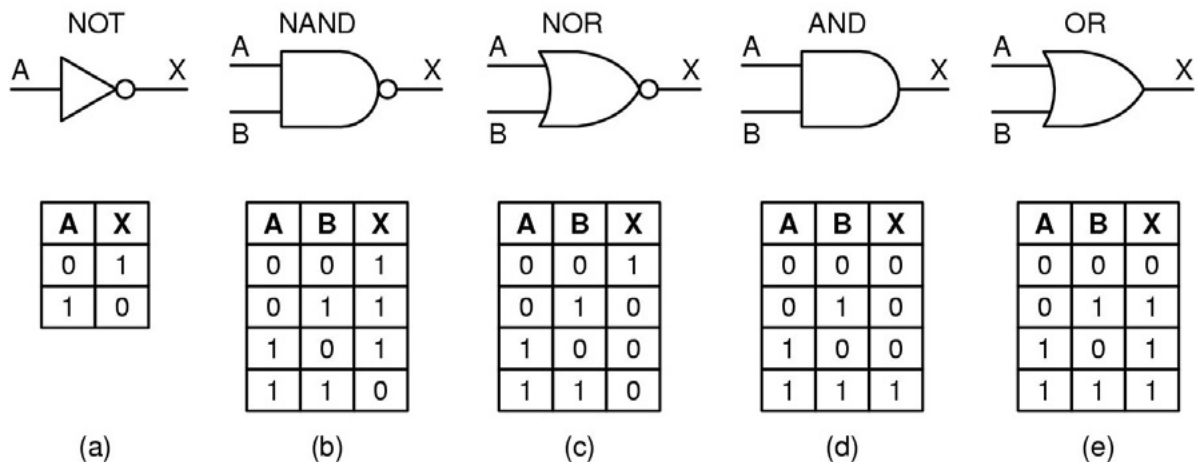
1.1. A Álgebra de Boole e a Lógica Binária

Na matemática e na ciência da computação, a Álgebra de Boole é composta de estruturas algébricas que representam as operações lógicas E, OU e NÃO, além das operações, envolvendo a Teoria dos Conjuntos. Ela também é a base da matemática computacional, baseada em números binários.

Recebeu o nome de George Boole, matemático inglês, que viveu no século XIX, e a definiu como parte de um sistema de lógica. Hoje, a álgebra booleana tem aplicações na eletrônica e na computação.

Os operadores de álgebra booleana podem ser representados de várias formas e são usados na criação de circuitos lógicos, que veremos mais adiante.

Figura 6: As portas que compõem os circuitos e sua representação binária



Fonte: http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula_5.html.

1.2. A codificação de Caracteres em Binário

Há diversos códigos ou sistemas de representação de caracteres utilizando a base binária. Alguns exemplos são:

- **EBCDIC (Extended Binary Coded Decimal Interchange Code** – Código Decimal de Intercâmbio Codificado em Binário Estendido);
- **BCD (Binary Coded Decimal** – Decimal Codificado em Binário);
- **ASCII (American Standard Code for Information Interchange** – Código Padrão Americano para Intercâmbio de Informações), muitas vezes referido como ASC2 por questões de praticidade.

O ASCII utiliza inicialmente 7 bits para a representação de caracteres, o que possibilita 128 (2⁷) combinações possíveis, mas ainda há uma versão estendida, o Extended ASCII, que permite outras 128 combinações para a representação de símbolos e caracteres especiais.

Porém, como 256 caracteres não são suficientes para suprir todas as línguas existentes no mundo, com suas letras, símbolos e caracteres, o código conhecido como Unicode tem ganhado cada vez mais aplicações, já que permite representações com 8, 16 e 32 bits, aumentando consideravelmente as possibilidades de representação.

Figura 7: Codificação em binário de caracteres em ASCII

CODIFICACIÓN BINARIA ASCII

Character	Binary Code	Character	Binary Code	Character	Binary Code	Character	Binary Code	Character	Binary Code
A	01000001	Q	01010001	g	01100111	w	01110111	-	00101101
B	01000010	R	01010010	h	01101000	x	01111000	.	00101110
C	01000011	S	01010011	i	01101001	y	01111001	/	00101111
D	01000100	T	01010100	j	01101010	z	01111010	0	00110000
E	01000101	U	01010101	k	01101011	!	00100001	1	00110001
F	01000110	V	01010110	l	01101100	"	00100010	2	00110010
G	01000111	W	01010111	m	01101101	#	00100011	3	00110011
H	01001000	X	01011000	n	01101110	\$	00100100	4	00110100
I	01001001	Y	01011001	o	01101111	%	00100101	5	00110101
J	01001010	Z	01011010	p	01110000	&	00100110	6	00110110
K	01001011	a	01100001	q	01110001	'	00100111	7	00110111
L	01001100	b	01100010	r	01110010	(	00101000	8	00111000
M	01001101	c	01100011	s	01110011	)	00101001	9	00111001
N	01001110	d	01100100	t	01110100	*	00101010	?	00111111
O	01001111	e	01100101	u	01110101	+	00101011	@	01000000
P	01010000	f	01100110	v	01110110	,	00101100	_	01011111

Fonte: www.areatecnologia.com/informatica/codificacion-binaria.html.

A notação binária está bastante presente em outras áreas da TI, como, por exemplo, os sistemas operacionais e as redes de computadores, onde as divisões de redes em sub redes utilizam o cálculo de máscara de sub rede, feito em binário. Embora esse tópico fuja ao escopo deste curso, é importante saber o quanto o conhecimento da notação binária é fundamental para o profissional de TI.

Considerações Finais

Vimos nesta aula alguns aspectos muito importantes no estudo da notação binária:

- A representação de números e caracteres;
- A álgebra booleana e os circuitos digitais;
- As aplicações da notação binária em diversas áreas da TI.

Como vimos, o código binário representa a linguagem de máquina, adequada ao computador. O processo de compilação nos ajuda a transformar o que queremos passar para o computador na linguagem que ele entende (linguagem de máquina).

É fundamental que você tenha entendido a importância e as características da notação binária para nossa próxima aula, em que trataremos não só dela, mas também das notações octal e hexadecimal.

Até lá!

Material Complementar



Assista a esse pequeno vídeo sobre a comparação entre a linguagem binária e a linguagem humana, por Pierre Lévy. Link: <https://www.youtube.com/watch?v=JN2JrJYR1TA&t=21s>.

Referências

STALLINGS, William. **Arquitetura e organização de computadores: projeto para o desempenho**. 8ª edição. Editora Pearson. Livro (642 p.). ISBN 9788576055648.

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 3ª edição. Editora Pearson. Livro (674 p.). ISBN 9788576052371.

TANENBAUM, Andrew S. **Organização estruturada de computadores**. 6ª edição. Editora Pearson. Livro (628 p.). ISBN 9788581435398.