Unidade 3: Sistemas de Numeração

Numerais Binários e Bases de Potência de Dois

Prof. Daniel Caetano

Objetivo: Apresentar as diferentes bases numéricas, preparando o aluno para compreender e trabalhar com lógica binária.

Bibliografia STALLINGS, 2003; MURDOCCA e HEURING, 2000.

INTRODUÇÃO

Como discutido na aula anterior, o computador trabalha com sinais elétricos do tipo "ligado" e "desligado" e, como uma consequência disso, quando usamos sinais em fios para representar números, torna-se mais natural uma representação em potências de dois.

Sendo assim, a capacidade de leitura/especificação de valores nas mais diversas bases de numeração torna-se fundamental a todo profissional que almeje trabalhar com lógica digital e a construção/manutenção de equipamentos baseados nesta.

Para propiciar familiaridade dos alunos com estas diferentes bases numércias, este será o assunto desta e das próximas aulas, antes de estudarmos o funcionamento interno dos equipamentos computacionais.

1. REPRESENTAÇÕES NUMÉRICAS

Primeiramente, é importante diferenciar o que são números do que são quantidades. A quantidade de elementos em um conjuto é um conceito abstrato oriundo da contagem dos elementos. É possível comparar quantidades - isto é, dizer se um conjunto é maior que outro - independentemente de existir um nome para essa quantidade. Por exemplo: os dois conjuntos abaixo possuem diferentes quantidades de bolinhas:

Conjunto 1	Conjunto 2
o o o o o	o o o o o o o o o o o o o o o

Os números são representações simbólicas convenientes para quantidades. Por exemplo: o Conjunto 1 tem 5 bolinhas e o Conjunto 2 tem 15 bolinhas. Entretanto, será que essa é a única forma de representar quantidades?

Na verdade, esta não apenas não é a única como também não foi a primeira. Esta forma de representação numérica é chamada "representação decimal com numerais indu-arábicos". Uma outra forma tradicional de representar os números é através dos

numerais romanos, que não seguem uma base numércia tradicional usando letras como I, L, C, X, M e V para representar as quantidades. Segundo a representação romana, o primeiro conjunto tem V bolinhas e o seguindo tem XV bolinhas.

Existem outras representações "não-decimais" usando números indú-arábicos, como a binária e a octal. Existem aquelas que usam caracteres alfanuméricos para representar as quantidades, como a hexadecimal. A tabela a seguir mostra a contagem de 0 a 15 representada em diferentes formas.

Decimal	Romana	Binária	Octal	Hexadecimal
0	-	0	0	0
1	I	1	1	1
2	II	10	2	2
3	III	11	3	3
4	IV	100	4	4
5	V	101	5	5
6	VI	110	6	6
7	VII	111	7	7
8	VIII	1000	10	8
9	IX	1001	11	9
10	X	1010	12	A
11	XI	1011	13	B
12	XII	1100	14	C
13	XIII	1101	15	D
14	XIV	1110	16	E
15	XV	1111	17	F

Observe que, em cada linha, temos diferentes representações para uma mesma quantidade!

Qual a razão para essa "confusão" toda? Bem, algumas representações, como a romana, são muito antigas e, posteriormente, foram substituídas na maioria dos usos pela numeração indú-arábica decimal. A numeração decimal, por sua vez, parece ser a mais lógica para nós, já que somos capazes de contar com as mãos até 10 elementos.

Entretanto, em alguns casos - como no caso dos computadores, temos de representar as quantidades usando apenas **fios**, pelos quais pode (ou não) passar uma corrente. Nestes casos, somos obrigados a usar vários fios para representar um número, e ele acaba sendo, obrigatoriamente, representado na forma binária.

Na ***eletrônica digital***, cada fio/conexão devem indicar apenas um de dois valores: ligado (com tensão) ou desligado (sem tensão). Essa decisão tem um impacto bastante relevante na forma com que representamos as informações em um computador: considerando que o fio é a "mão" do computador, ele considera apenas dois dígitos: 0 e 1 - diferentemente de nós, que consideramos os dígitos de 0 a 9!

Ocorre que, como a representação biária é um tanto desajeitada - devido ao grande número de dígitos - frequentemente usamos uma representação equivalente, de fácil conversão para o binário: o hexadecimal. Observe que, na representação hexadecimal, o segundo dígito só será necessário quando a representação binária tiver 5 dígitos (não aparece nessa tabela). É um considerável ganho para economia de escrita e, adicionalmente, reduz a probabilidade de erros de digitação - bastante alta quando os números envolvem apenas longas sequências de zeros e uns.

Como pode haver confusão se usarmos diferentes representações numéricas em um mesmo texto, usa-se a seguinte convenção:

Números Decimais: são escritos normalmentem, SEM zero à esquerda.
Exemplos: 5, 30, 44.
Números Binários: são escritos com o acréscimo de um "b" ao final ou com índice 2.
Exemplos: 101b, 11110b, 101100b,101100_2.
Números Hexadecimais: são escritos com o acréscimo de um "h" ao final ou com índice 16.
Exemplos: 5h, 1Eh, 2Ch, $2C_{16}$

Nota: uma forma alternativa de representar os números hexadecimais é usada nas linguagens C/C++, Java e outras. Nestas, ao invés de se acrescentar o "h" ao final, acrescenta-se o prefixo "0x" no início. Exemplos: 0x5, 0x1E, 0x2C.

2. NOTAÇÃO POSICIONAL

Um grande avanço da notação indú-arábica decimal com relação à romana é o uso de notação posicional. A notação posicional significa que a quantidade que um número representa depende da posição em que ele aparece na reprentação completa. Por exemplo: qual a quantidade representada pelo símbolo "1"? Se você respondeu "1, oras!", errou! O símbolo 1 tem diferentes significados, de acordo com a posição no número!

Consideremos os números decimais. Neste caso, se o 1 estiver na primeira casa, ele vale **uma unidade**. Se estiver na segunda casa, ele vale **uma dezena**. Se estiver na terceira casa, ele vale **uma centena**... na quarta vale **uma unidade de milhar** e assim por diante. Observe:

1 : Um
10 : Dez
100 : Cem
1000 : Mil

1101 : Mil cento e um.

Neste último caso, observe que o número pode ser construído com uma soma de suas partes: 1101 = 1000 + 100 + 1. O mesmo vale quando temos outros números:

12345 = 10000 + 2000 + 300 + 40 + 5
4532 = 4000 + 500 + 30 + 2

Observe que a posição de um número indica quantos zeros devem ser acrescentados ao seu lado para identificarmos a quantidade que ele representa. Considere este exemplo:

4356 = 4000 + 300 + 50 + 6

Se conisderarmos esse número contando suas casas da direita para a esquerda, começando em zero, teremos uma correspondência direta:

Casa	3	2	1	0
Dígito	4	5	3	2
Quantidade	4.000	500	30	2

Observe: na casa 3, a quantidade real tem 3 zeros; na casa 2, a quantidade real tem 2 zeros... e assim por diante. Na tabela abaixo, escreveremos as quantidades de uma maneira diferente:

Casa	3	2	1	0
Dígito	4	5	3	2
Quantidade	4×10^3	5×10^2	3×10^1	2×10^0

Observe que o expoente do "10" é exatamente a posição do dígito em questão. Isso ocorre porque estamos usando, para construir os números, 10 dígitos diferentes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Sempre que não há mais dígitos em uma das posições, acrescentamos um à posição imediatamente à esquerda do número.

É por essa razão que esta representação é chamada de "decimal". A representação binária, por usa vez, usa apenas dois valores para os dígitos: 0 e 1. A representação hexadecimal usa dezesseis valores diferentes para os dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F.

3. A NOTAÇÃO BINÁRIA

Como apresentado anteriormente, acredita-se que os humanos trabalhem com números decimais por conta da quantidade de dedos que temos nas mãos; os computadores, entretanto, foram construídos com uma outra característica e, portanto, a representação mais natural neste caso é a binária.

A informação que pode ser representada por um "fio" - 0 ou 1 - é denominada **bit**, e é a menor unidade de informação de um computador. Se um processador tivesse apenas 1 bit, ele só seria capaz de representar os números 0 e 1. Mas, e se ele tiver 2 bits? A tabela abaixo mostra todas as combinações possíveis:

00 01 10 11

E se ele tiver 3 bits?

000 001 010 011 100 101 110 111

E se ele tiver 4 bits?

0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111

Observe que o número de bits representa o número de dígitos binários; além disso, considerando **n** bits, o número de combinações possíveis é dado por 2^n: um computador com 8 bits pode representar até $2^8 = 256$ números e um de 16 bits pode representar até $2^{16} = 65536$ números... e assim por diante.

Nota: como 256 variações eram suficientes para representar a maior parte das informações necessárias nos primeiros computadores, o conjunto de 8 bits ganhou um nome específico: byte. Assim, um **byte é um conjunto de 8 bits**.

3.1. Conversão de Números Binários para Decimais

Mas que números esse valores representam, em nossa notação decimal?

Existe uma regra de conversão muito simples. Lembremos como representamos o número decimal anterioremente:

Dígito	3	2	1	0
Número	1	5	3	7

$1537 = 1*10^3 + 5*10^2 + 3*10^1 + 7*10^0$

Se fizermos o mesmo com um número binário, por exemplo, 1101, teremos:

Dígito	3	2	1	0
Número	1	1	0	1

1101 binário $= 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

Observe que os 10^n foram substituídos por 2^n; a razão para isso é que antes estávamos em uma base decimal, agora estamos em uma base binária. Façamos essa conta:

$1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 8 + 4 + 0 + 1 = 13$

Logo: 1101 binário = 13 decimal

Observe que agora temos uma confusão de representação: um número binário poderia ser lido erradamente como um número decimal. Para evitar esse problema, é usual acrescentar a letra "b" após valores binários, para evitar confusão. Em outras palavras, a representação é:

Texto	Valor (em decimal)
1101	1101
1101b	13

A tabela a seguir mostra a conversão das 16 combinações de números de 4 bits de binário para decimal:

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

3. CONVERTENDO HEXADECIMAL PARA DECIMAL

Os números hexadecimais usam 16 símbolos para cada dígito, como já visto. As quantidades de 0 a 15 serão representadas com apenas um digito em hexadecimal: 0h a Fh. A partir da quantidade 16 serão necessarios mais dígitos: 16 = 10h, 17 = 11h, 18 = 12h, 31 = 1Fh e assim por diante.

3.1. Conversão de Números Hexadecimal para Decimais

Multiplica-se cada dígito pela correspondente potência de **dezesseis**. Exemplo: converter o número 0x2F3C para decimal:

Dígito	3	2	1	0
Número	2	F	3	C

$0x2F3C=2*16^3 + 15*16^2 + 3*16^1 + 12*16^0 = 2*4096 + 15*16 + 3*16 + 12*1 = 12092$

4. BIBLIOGRAFIA

STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.

MURDOCCA, M. J; HEURING, V.P. **Introdução à Arquitetura de Computadores**. S.I.: Ed. Campus, 2000.