Unidade 4: Conversões e Cálculos em Outras Bases

Prof. Daniel Caetano

Objetivo: Apresentar os métodos de conversão de decimal para diferentes bases e como realizar cálculos nas diferentes bases.

Bibliografia STALLINGS, 2003; MURDOCCA e HEURING, 2000.

INTRODUÇÃO

Alguns comportamentos do computador só são possíveis ao compreender como os nossos números são armazenados em sua memória e, em especial, ele realiza seus cálculos.

O objetivo desta aula é apresentar as conversões dos números decimais, usados pelos humanos, para os formatos binário e hexadecimal, usados pelos computadores. Adicionalmente, serão apresentados, simplificadamente, o processo de cálculo (soma) de números em outras bases e também a representação de caracteres.

1. CONVERSÃO DE DECIMAIS PARA BINÁRIOS

A conversão de números decimais para binários é bastante simples, sendo realizada com um processo de sucessivas divisões inteiras por dois, parando quando o divisor se tornar 0. Os restos das divisões vão compondo o valor em binário, da esquerda para a direita. Por exemplo: vamos transformar o valor 13 em sua representação binária:

13 / 2 = 6 e sobra... 1
6 / 2 = 3 e sobra... 0
3 / 2 = 1 e sobra 1
1 / 2 = 0 e sobra 1

Assim, o valor 13 é representado em binário como 1101b. Tentemos novamente com outro número maior, 118:

118 / 2 = 59 e sobra... 0
59 / 2 = 29 e sobra... 1
29 / 2 = 14 e sobra... 1
14 / 2 = 7 e sobra... 0
7 / 2 = 3 e sobra... 1
3 / 2 = 1 e sobra... 1
1 / 2 = 0 e sobra... 1

Assim, o valor 118 é representado em binário como 1110110b

2. CONVERSÃO DE DECIMAIS PARA HEXADECIMAIS

A conversão é similar à anterior, e é feita com divisões sucessivas por **dezesseis**, anotando os restos da divisão, que formam o número hexadecimal da direita para a esquerda. Exemplo: converter o valor 12092 em sua representação hexadecimal:

12092 / 16 = 755 e sobra... 12 (C)
755 / 16 = 47 e sobra... 3
47 / 16 = 2 e sobra... 15 (F)
2 / 16 = 0 e sobra... 2

Logo, 12092 = 0x2F3C

3. ARITMÉTICA EM OUTRAS BASES

Quando trabalhamos com números decimais, fazemos operações diretas. Por exemplo:

$$\begin{array}{r} {\scriptstyle 1} \\ 15 \\ +7 \\ \hline 22 \end{array}$$

No fundo, alinhamos as casas (unidade com unidade, dezena com dezena, centena com centena...) e depois somamos uma a uma, começando com a unidade e, em seguida, partindo para a dezena e centena. Quando o resultado de uma das casas é maior do que o valor da base (no exemplo, 5 + 7 = 12), subtraímos deste resultado o valor da base (12 - 10 = 2) e, fazemos o "vai um" (representado no exemplo como o pequeno algarismo 1 sobre o 15).

Realizar a soma em outras bases é exatamente a mesma coisa. Veja em binário:

```
 1  1
 1101b          =    13
+0101b          =   +5
10010b          =    18
```

Começando da direita para a esquerda:

Primeira Casa: 1 + 1 = 2; como 2 não pode ser representado em binário, subtraímos 2 (2-2 = 0) e fazemos o vai um.

Segunda Casa: 0 + 0 = 0, somando com o "1" que veio da casa anterior 0+1 = 1.

Terceira Casa: 1 + 1 = 2. Mais uma vez não é possível representar, deixamos zero (2-2) no lugar e vai um.

Quarta Casa: A soma é 1 + 0 = 1, mas ao somar com o "1" que veio da casa anterior, 1+1 = 2, deixando zero no lugar e, mais uma vez, "vai um".

Quinta Casa: Como ela não existe nos números originais, permanece apenas o "1" que veio da casa anterior.

O processo é análogo para outras bases:

```
   1
 0x25      =     37
+0x3C      =    +60
 0x61      =     97
```

Começando da direita para a esquerda:

Primeira Casa: 5 + C(12) = 17. 17 não pode ser representado... então indicamos 17-16 = 1 e vai um

Segunda Casa: 2 + 3 = 5. Somando com o "1" que veio da casa anterior: 5 + 1 = 6.

Será que podemos aplicar a mesma lógica para a subtração? É claro que sim! Vejamos primeiro com decimais:

```
 1
 25
-7
 18
```

Primeira Casa: temos 5 - 7; não é possível fazer, então "emprestamos um" da próxima casa, que aqui na unidade vale 10 e a nova conta é (10+5) - 7 = 8.

Segunda Casa: temos 2, mas que deve subtrair o "1" que foi emprestado pela primeira casa, então 2-1 = 1.

Vejamos agora em binário

```
  1 1 1
  1100b          =     12
 -0101b          =     -5
  0111b          =      7
```

Primeira Casa: 0 - 1; não é possível. Então "emprestamos 1" da próxima casa, que aqui na primeira casa vale 2. A nova conta é, então (2+0) -1 = 1

Segunda Casa: 0-0 = 0; entretanto, precisamos descontar o 1 que foi emprestado para a primeira casa; como 0-1 não é possível, somos obrigados a emprestar 1 da terceira casa, que aqui vale 2. A nova conta é: (2+0)-1 = 1.

Terceira Casa: 1-1 = 0... mas mais uma vez é preciso descontar o 1 que foi emprestado para a casa anterior... e, para isso, é preciso emprestar 1 da quarta casa! Daí (2+0)-1 = 1.

Quarta Casa: 1-0 = 1, que descontado o 1 que foi emprestado... 0.

O mesmo pode ser aplicado para a multiplicação. Façamos direto em binário:

```
   11b      =      3
  x10b      =     x2
   00b             6
  11b+
  110b
```

A divisão fica como exercício!

4. REPRESENTAÇÃO DE CARACTERES

Até o momento vimos como armazenar números na memória. Mas como armazenar letras? Bem, este foi um problema que surgiu nos primórdios da computação e, por esta razão, existe uma solução padrão, que é a chamada Tabela ASCII (ASCII significa *American Standard for Computer Information Interchange*). A tabela ASCII relaciona cada valor numérico de um byte a cada um dos códigos visuais usados por nós na atividade da escrita. A tabela de conversão é apresentada na página seguinte (fonte: Wikipédia).

Observe, porém, que nem todos os caracteres são definidos por essa tabela: em especial, os caracteres acentuados estão faltando. Mas não são apenas estes: também não estão presentes os caracteres japoneses, chineses, russos... dentre tantos outros.

Por essa razão, atualmente existem diversas outras "tabelas de código de caracteres" ou "páginas de código de caracteres" (do inglês *codepage*), que extendem a tabela abaixo indicando os símbolos faltantes aos códigos livres (não especificados pela tabela ASCII). Entretanto, com a grande troca de arquivos entre pessoas de países diferentes, isso começou a causar alguma confusão.

Foi assim que surgiram então os códigos Unicode, que são versões alternativas e universais à tabela ASCII. O padrão UTF (Unicode Transformation Format) define várias tabelas, sendo as mais conhecidas e usadas as tabelas UTF-8 e UTF16. A tabela UTF-8 define 256 caracteres, como a tabela ASCII, mas com um padrão que tenta alocar a grande maioria dos símbolos usados pela maioria das línguas. Já o UTF-16 define 65.536 caracteres, englobando a grande maioria dos caracteres de todas as línguas. Existe ainda o padrão UTF-32, com capacidade para definir até 4 bilhões de caracteres, mas que é muito pouco usado.

Binário	Decimal	Hexa	Glifo
0010 0000	32	20	
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binário	Decimal	Hexa	Glifo
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binário	Decimal	Hexa	Glifo
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

5. 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.