Capítulo 2

Sistemas Numéricos e a Representação Interna dos Dados no Computador

2.0 Índice

2.1 Sistemas Numéricos

Sistemas numéricos são sistemas de notação usados para representar quantidades abstratas denominadas números. Um sistema numérico é definido pela base que utiliza. A base é o número de símbolos diferentes, ou algarismos, necessários para representar um número qualquer, dos infinitos possíveis no sistema. Por exemplo, o **sistema decimal**, utilizado hoje de forma universal, utiliza dez símbolos diferentes ou dígitos para representar um número e é, portanto, um sistema numérico na base 10.

Valores posicionais

Em um sistema de número posicional, um número é representado por uma seqüência de dígitos onde cada posição de dígito tem um peso associado. Tomando como exemplo o sistema decimal, ou base 10, que é sistema numérico que utilizamos diariamente (0, 1, 2, ... 9), o valor D de um número decimal de 4 dígitos $d_3d_2d_1d_0$ é $D = d_3*10^3 + d_2*10^2 + d_1*10^1 + d_0*10^0$. Cada dígito d_i tem um peso de 10^i. Por exemplo, o número 3.098.323 (base 10) é a representação de $3*10^6 + 0*10^5 + 9*10^4 + 8*10^3 + 3*10^2 + 2*10^1 + 3*10^0$.

2.1.1 Sistema Binário

O sistema binário, ou base 2, apresenta unicamente dois dígitos: 0,1. Neste sistema a contagem é realizada como segue: 0, 1, 10, 11, 100, 101, 110, 111, 1000, ...

Conversão Binário para Decimal

Sendo binário um sistema de número posicional, o valor B de um número binário de 8 dígitos $b_7b_6b_5b_4b_3b_2b_1b_0$ é $B = b_7*2^7 + b_6*2^6 + b_5*2^5 + d_4*2^4 + d_3*2^3 + d_2*2^2 + d_1*2^1 + d_0*2^0$. Cada dígito b_i tem um peso de 2^i. Assim o valor binário 10101010_b é calculado como segue $10101010_b = 0*1+1*2+0*4+1*8+0*16+1*32+0*64+1*128 = 170_d$. Esta é a conversão de um número binário para decimal. Outro exemplo $10011001_b = 1+8+16+128=153_d$

Conversão Decimal para Binário

No sistema decimal, por exemplo, o número 654 corresponde a 4 unidades, 5 dezenas e 6 centenas. Para verificar isto, divide-se o número pela sua base (que é 10):

654/10 =	65		Resto 4 (*1)
	/10 =	6	Resto 5 (*10)
		/10	Resto 6 (*100)

Para a conversão de decimal para binário utilizamos o mesmo processo. Por exemplo, para obtermos o correspondente binário do número 200_d, dividimos primeiramente este valor por 2 e anotamos o resto de cada divisão. Em seguida, dividimos novamente o dividendo da operação anterior por 2 e anotamos novamente o resto da divisão. Isto é repetido até que o resto da divisão seja 0, conforme abaixo:

200/2=100	Resto 0
100/2= 50	Resto 0
50/2 = 25	Resto 0
25/2 = 12	Resto 1
12/2 = 6	Resto 0
6/2 = 3	Resto 0
3/2 = 1	Resto 1
1/2 = 0	Resto 1

O correspondente binário de 200_d é obtido unindo-se os restos da divisão por 2 na ordem inversa, assim $200_d=11001000_b$.

2.1.2 Sistema Octal

O sistema binário ou base 8 apresenta oito dígitos: 0, 1, 2, 3, 4, 5, 6, 7. Neste sistema, a contagem é realizada como segue: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20,...

Conversão Octal para Decimal

Sendo o sistema octal um sistema de número posicional, o valor O de um número octal de 4 dígitos $o_3o_2o_1o_0$ é $O = d_3*8^3 + d_2*8^2 + d_1*8^1 + d_0*8^0$. Cada dígito o_i tem um peso de 8^i. Assim o valor octal 175_8 é calculado como segue $175_8 = 5*1+7*8+1*64 = 125_{10}$. Esta é a conversão de um número octal para decimal.

Conversão Decimal para Octal

Para a conversão de decimal para octal utilizamos o mesmo processo da conversão do sistema decimal para binário. Por exemplo, para obtermos o correspondente octal do número 200_d, dividimos primeiramente este valor por 8 e anotamos o resto de cada divisão. Em seguida, dividimos novamente o dividendo da operação anterior por 8 e anotamos novamente o resto da divisão. Isto é repetido até que o resto da divisão seja 0, conforme abaixo:

200/8= 25 Resto 0
25/8 = 3 Resto 1
3/8 = 0 Resto 3

O correspondente octal de 200_d é obtido unindo-se os restos da divisão por 8 na ordem inversa, assim $200_d=310_o$.

2.1.3 Sistema Hexadecimal

Na base hexadecimal tem-se 16 dígitos que vão de 0 à 9 e da letra A até F. Estas letras representam os números 10_d a 15_d. Assim nós contamos os dígitos hexadecimais da seguinte forma: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, ..., 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, ...

Conversão Binário para Hexadecimal

A conversão entre números binários e hexadecimais é simples. A primeira coisa a fazer é dividir o número binário em grupos de 4 bits, começando da direita para a esquerda, os lugares que faltam são complementados por zeros. Por exemplo, o número 101011_b ($1+2+8+32=43_d$), nós dividimos este em grupos de 4 bits e nós temos 10;1011. Nós completamos o último grupo com zeros: 0010;1011. Após nós tomamos cada grupo como um número independente e nós convertemos estes em dígitos decimais: 0010;1011=2;11. Mas desde que nós não podemos representar o número hexadecimal como 211 porque isto é um erro, nós temos que substituir todos os números decimais maiores que 9 pelas suas respectivas representações em hexadecimal, com o que nós obtemos: $2B_h$. A tabela abaixo pode auxiliar na conversão de números binário para hexadecimal.

Binário	Hexadecimal	Decimal
0000	00	0
0001	01	1
0010	02	2
0011	03	3
0100	04	4
0101	05	5
0110	06	6
0111	07	7
1000	08	8
1001	09	9
1010	0A	10
1011	0B	11

1100	0C	12
1101	0D	13
1110	0E	14
1111	0F	15

Afim de obter um número hexadecimal em binário é apenas necessário inverter os passos.

Conversão Hexadecimal em Decimal

Para converter um número hexadecimal em decimal, nós utilizamos a mesma fórmula utilizada na conversão binário para decimal, sendo que a base 2 é trocada por 16. Por exemplo, para converter $B2A_h$ em decimal:

B -> $11*16^2$ = 2816_d
2 -> $2*16^1$ = 32_d
A -> $10*16^0$ = 10_d
2858_d

Conversão Decimal para Hexadecimal

Para converter um número decimal em hexadecimal, nós utilizamos a mesma fórmula utilizada na conversão de um número decimal para binário, dividindo por 16 em vez de 2. Por exemplo, para converter 1069_d em hexadecimal:

1069/16= 66 Resto $13_d = D_h$
66/16 = 4 Resto $2_d = 2_h$
4/16 = 0 Resto $4_d = 4_h$
1069d = $42D_h$

2.2 Operações Aritméticas

2.2.1 Aritmética Binária

Esta seção apresenta as quatro operações básicas no sistema binário: adição, subtração, divisão e multiplicação.

Adição

Para somar dois números binários, fazem-se as contas coluna a coluna, da direita para a esquerda, como de costume, fazendo o transporte de um **(<e vai um>)** quando for o caso. Para isto, observe as seguintes operações básicas:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 1 = 10 (1 mais 1 é igual a 0 e vai 1)
- 1 + 1 + 1 = 11 (1 mais 1 mais 1 é igual a 1 e vai 1)

Exemplos:

+ 1 1		*1 11*	*111*
101	100100	11001	101110
+ 1101	+ 10010	+ 10011	+ 1110
10010	110110	101100	111100

Subtração

Existem duas formas para fazer a subtração binária:

- Como o conjunto de símbolos contém apenas 2 dígitos, ao se efetuar a subtração parcial entre 2 dígitos, um do diminuendo e outro do diminuidor, se o segundo (diminuidor) exceder o primeiro (diminuendo), subtrai-se uma unidade ao dígito imediatamente à esquerda no diminuendo (se existir e o seu valor for 1), convertendo-o a 0. Em seguida, substituímos o diminuendo por 2, que

corresponde à equivalência 1*2, da unidade extraída. Se o dígito imediatamente à esquerda for 0, procura-se nos dígitos consecutivos.

Exemplos: 11101 - 111

		02
	02	*02*
11101	11~~1~~01	1~~11~~01
-111	-111	-111
0	10	10110

Exemplos: 11000 - 111

0112	*0112*
01~~2~~0	*01~~2~~0*
0~~2~~00	*0~~2~~00*
1~~1~~000	1~~1~~000
-111	-111
	10001

- A segunda forma de realizar a subtração, por exemplo de a-b, e realizar a soma de a por -b. Esta subtração é feita pelo chamado método do complemento de dois. O complemento de dois transforma um número positivo em negativo. Neste método, o diminuendo (a) é somado com o complemento de dois do diminuidor (-b). Note que o número de dígito dos operandos devem ser o mesmo: para isto complemente o operando com menor número de dígitos com zeros a esquerda (antes do complemento). Para realizar o complemento de dois, basta trocar os uns pelos zeros e vice-versa e adicionar um ao resultado. Por exemplo, a subtração de 1110-101 é feita da seguinte maneira:
  1. Completa-se o número de dígitos do diminuidor: 0101
  2. Realiza-se o complemento de dois do diminuidor: 1010+1=1011.
  3. Soma-se os dois operandos 1110+1011=11001
  4. Despreza-se o transporte final: 1001

Multiplicação

A multiplicação na base 2 - ou em qualquer outra base - pode fazer-se por adições sucessivas; para calcular A*B basta somar A a si própria B vezes.

Exemplo: $101_b * 100_b = ?$ Lembrado que $100_b = 4_b$, então

101 * 100 =

```
      1
    1 1
    1 0 1
  1 1 0 1   100 vezes
  1 1 0 1
+ 1 0 1
1 0 1 0 0
```

Uma forma, e a ideal, é fazer a operação semelhante à multiplicação decimal, exceto pelo fato da soma final dos produtos se fazer em binário. Para tal, as seguintes igualdades devem ser respeitadas:

- 0*0=0; 0*1=0; 1*0=0; 1*1=1

Exemplos:

- Multiplicar os números 1011 e 1101.

```
    1 0 1 1  ———— 1 1
  * 1 1 0 1  ———— 1 3
  ---------
    1 0 1 1
  0 0 0 0
1 0 1 1
+ 1 0 1 1
---------------
1 0 0 0 1 1 1 1  ———— 1 4 3
```

- Multiplicar os números 1001 e 1101.

```
      1 0 0 1  ———————   9
    * 1 1 0 1  ——————— 1 3
    ---------
      1 0 0 1
    0 0 0 0
  1 0 0 1
+ 1 0 0 1
-----------
  1 1 1 0 1 0 1  ——————— 1 1 7
```

Divisão

Analogamente, a divisão pode ser feita por subtrações sucessivas, até obtermos uma diferença igual a zero (no caso de uma divisão exata), ou um número menor que o divisor.

Exemplo:

$$\frac{10000}{1000} = ?$$

```
  1 0 0 0 0
- 1 0 0 0
---------
  0 1 0 0 0  ——— 1 vez
- 1 0 0 0
---------
  0 0 0 0 0  ——— 2 vezes
```

O resultado é $2_{(10)}$, isto é, $10_{(2)}$

Mas esta divisão pode ser feita de maneira idêntica à divisão decimal, exceto pelo fato das multiplicações e subtrações internas ao processo serem feitas em binário.

Exemplo:

- Dividir 11011 e 101.

```
1 1 0 1 1 = 2 7
    1 0 1 = 5

  1 1 0 1 1   |1 0 1
- 1 0 1       -----
-------         1 0 1
  0 0 1 1 1
    - 1 0 1
    -------
      0 1 0      o quociente é   1 0 1
                 e o resto é     1 0
```

- Dividir 1010101 e 101.

```
1 0 1 0 1 0 1 = 8 5
      1 0 1 = 5

    1 0 1 0 1 0 1   |1 0 1
  - 1 0 1           ---------
  -------             1 0 0 0 1
    0 0 0 0 1 0 1
        - 1 0 1
        -------
          0 0 0
```

A prova é:

```
      1 0 0 0 1    quociente
        * 1 0 1    divisor
    -----------
      1 0 0 0 1
    0 0 0 0 0
+ 1 0 0 0 1
-----------
  1 0 1 0 1 0 1    dividendo
```

2.2.2 Aritmética Hexadecimal

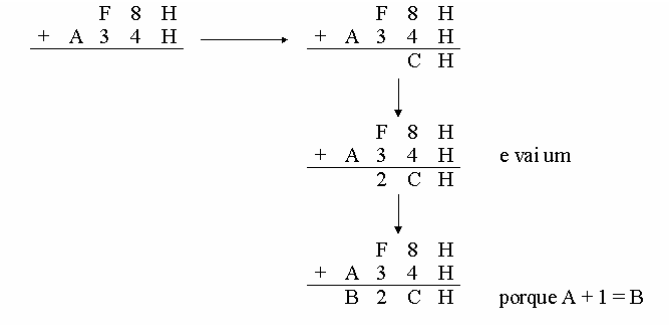
Adição

Como exemplo, suponha a adição de 8_h+5_h, se somada em decimal o valor seria 13. Em hexadecimal, o valor 13 é representado por D_h. Deve-se reparar que, tal como nos habituamos a fazer na Escola Primária, sempre que o resultado iguala ou ultrapassa a base, subtraímos esta ao resultado, e fazemos um transporte para a coluna seguinte («e vai um», neste caso). Suponha agora a adição de 19 por 9:

- Em decimal, o resultado seria 28;

- Em hexadecimal, inicialmente somamos os dígitos menos significativos: 9_h+9_h = 18; como o resultado é maior que a base (16), então 18-16 = 2 e vai um para o dígito mais significativo. Portanto, 19_h+9h=22_h;

Não é preciso converter os números $F8_h$ e $A34_h$ para decimal, somá-los e reconverter o resultado para a base 16. Podemos fazer a conta coluna a coluna. Então $F8_h$ + $A34_h$ é calculado da seguinte forma:



Subtração

Vamos ver a subtração a partir de um exemplo: 27H-1EH. Efetuamos a operação de subtração coluna a coluna. Na primeira coluna, o diminuidor (E) é superior ao diminuendo (7). Então, adicionamos a base ao diminuendo, executamos a subtração, e há transporte de uma unidade que somamos ao diminuidor da coluna seguinte.

```
   2 7 H
 - 1 E H
 -------
     9 H    «e vai um»
```

retirando o número transportado do diminuendo da coluna da esquerda, 2-1, obtemos 1, e subtraindo 1 do diminuidor, obtemos 0:

```
   2 7 H
 - 1 E H
 -------
   0 9 H
```

Multiplicação

Esta operação pode fazer-se facilmente por meio da tabela de dupla entrada apresentada:

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	03	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	04	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	05	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	06	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	07	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	09	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0E	1C	2A	38	4C	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Como se vê, temos todos os algarismos hexadecimais (exceto o zero) nas entradas verticais e horizontais da tabela. Se quiséssemos calcular 5_h*9_h, por exemplo, encontraríamos o resultado na intercessão da coluna 5 com a linha 9. Então, $5_h*9_h = 2D_h$. Uma vez que a multiplicação é comutativa, então, o mesmo resultado se verifica na intercessão da coluna 9 com a linha 5.

- 1º. Exemplo:

  $A_h* 2_h$ = _____ (complete)

  2_h*7_h = _____ (complete)

- 2º. Exemplo:

```
  6  7  A  H
  *  3  2  H
-------------
```

Procedendo como de costume, vamos começar pelo produto do multiplicando pelo algarismo mais à direita do multiplicador:

```
  6  7  A  H
  *  3  2  H
-------------
```

```
  6  7  A  H
  *  3  2  H
-------------
        4  H
```

Na tabela, vemos que A * 2 = 14; então, escrevemos «4» e há transporte de 1 unidade (e vai um)

```
  6  7  A  H
  *  3  2  H
-------------
     F  4  H
```

2H * 7H = 0EH; 0EH + 1 (de transporte) = 0FH; escreve-se «F», e não há transporte.

```
  6  7  A  H
  *  3  2  H
-------------
  C  F  4  H
```

2H * 6H = 0CH; escreve-se «C», e não há transporte.

Calculamos em seguida o produto do multiplicando pelo 2º. algarismo (contando a partir da direita) do multiplicador.

```
        6  7  A  H
        *  3  2  H
-------------------
        C  F  4
1  3  6  E
-------------------
```

Agora é só somar os produtos parciais. Fica:

```
        6  7  A  H
        *  3  2  H
-------------------
        C  F  4
1  3  6  E
-------------------
1  4  3  D  4  H
```

- Fazer o produto dos seguintes números hexadecimais: B12H e 3FCH

```
           B  1  2  H
        *  3  F  C  H
-------------------------
        8  4  D  8
     A  6  0  E
  2  1  3  6
-------------------------
  2  C  1  B  B  8  H
```

Divisão

Esta é a operação mais difícil de fazer sem recorrermos à tabela anterior. Veja alguns exemplos:

- 1º. Exemplo: dividir os números hexadecimais 2F por 12.

```
2  F  |1  2
   B     2
```

2 * 2 = 4; F - 4 = B

```
2  F  |1  2
0  B     2
```

2 * 1 = 2; 2 - 2 = 0

Para verificarmos se é verdade, nada melhor que fazer a prova:

```
  1  2  H     divisor
  *  2  H     quociente
-------------
  2  4  H
+ B  H        resto
-------------
  2  F  H     dividendo
```

✍ 2º. Exemplo: Dividir os números hexadecimais 3F4 por A1

```
3  F  4   | 1  A
   B      | 2          2 * A = 4 (e vai um); F - 4 = B

3  F  4   | 1  2
0  B      | 2          2 * 1 = 2 + 1 do transporte = 3; 3 - 3 = 0

3  F  4   | 1  A
0  B  4   | 2  6       6 * A = C (e vão três); 4 - C = 8 (e vão 4)
   1  8                6 * 1 = 6 + 4 do transporte = 10; B - A = 1
```

2.3 Operações Lógicas

Existem quatro tipos de operações lógicas que se podem operar sobre números binários: AND, OR, XOR (ou exclusivo), e NOT.

2.3.1 Operações lógicas com bits

AND

A operação lógica AND é uma operação que aceita dois operandos. Estes operando são binários simples (base 2). A operação AND é

0 and 0 = 0
0 and 1 = 0
1 and 0 = 0
1 and 1 = 1
Uma maneira compacta de representar a operação lógica AND é com a tabela verdade, apresentada abaixo. As duas colunas a esquerda representam os dois operandos da operação *AND Op1 OP2*.

Op1	Op2	AND Op1 Op2
0	0	0
0	1	0
1	1	1

Em português, a operação lógica AND é: "se o primeiro operando é 1 e o segundo operando é 1, o resultado é 1, senão o resultado é 0".

OR

A operação lógica OR também é uma operação com dois operandos. Ela é definida como:

0 or 0 = 0
0 or 1 = 1
1 or 0 = 1
1 or 1 = 1

A tabela verdade da operação OR tem a seguinte forma:

Op1	Op2	OR Op1 Op2
0	0	0
0	1	1
1	0	1
1	1	1

A operação lógica OR significaria: "Se o primeiro operando ou o segundo operando (ou os dois) forem 1, o resultado é 1, senão o resultado é 0. Esta operação também é conhecida como ou inclusivo *(inclusive-OR)*.

XOR

A operação lógica XOR (ou exclusivo) também é uma operação com dois operandos. Ela é definida como:

0 xor 0 = 0
0 xor 1 = 1
1 xor 0 = 1
1 xor 1 = 0

A tabela verdade da operação XOR tem a seguinte forma:

Op1	Op2	XOR Op1 Op2
0	0	0
0	1	1
1	0	1
1	1	0

Em português a operação lógica XOR significaria: "Se o primeiro operando ou o segundo operando, mas não os dois, for 1, o resultado é 1, senão o resultado é 0.

NOT

A operação lógica XOR (ou exclusivo) também é uma operação com um operando. Ela é definida como:

not 0 = 1
not 1 = 1

A tabela verdade da operação NOT tem a seguinte forma:

Op1	NOT Op1
0	1
0	0

Em português a operação lógica NOT significaria: "Se o operando for 1, o resultado é 0, senão o resultado é 1".

2.3.2 Operações Lógicas com números

As operações lógicas trabalham apenas com operandos com bit único. Para realizar estas operações sobre um número, por exemplo de 8, 16, 32 bits, é necessário realizar a operação bit-a-bit. Por exemplo se quisermos realizar a operação lógica AND com dois operandos de 8 bits cada, teríamos que executar a operação lógica AND sobre cada par de bits independentemente:

```
    1011 0101
AND 1110 1110
    1010 0100
```

Como as operações lógicas são definidas em termos de valores binários, deve-se converter os números decimais, hexadecimais, etc., para números binários antes de realizar as operações lógicas.

2.4 Tipos de Dados Tratados pelo Computador

Todos os dados e as instruções armazenados em memória são codificados sob a forma de sinais elétricos do tipo ligado e desligado, representado pelos números 1 e 0. Cada unidade de informação deste tipo é chamada de bit, abreviação de *Binary digit*. Assim o sistema numérico adotado em sistemas computacionais é o binário, ou base 2.

Os computadores podem receber valores decimais, através do teclado, e escrever valores decimais, através do vídeo, por exemplo. Mas internamente os valores são armazenados e processados no sistema binário.

Um bit pode representar dois valores: 1 ou 0, ou então verdadeiro ou falso. Como isto é muito pouco, nós podemos unir dois ou mais bits para representar mais de dois valores. Neste caso, a quantidade de valores representáveis por uma seqüência de n bits é de 2^n. Algumas *strings* de bits têm nomes próprio:

- uma seqüência de 8 bits são chamados de **byte**

- uma seqüência de 4 bits é chamada de ***nibble***.
- um grupo de 16 bits é chamado de **word**.
- um grupo de 32 bits é chamado de **double word**.
- um grupo de 64 bits é chamado de **quad word**.

Por razões de simplificação de hardware, o número 1024 foi o escolhido para representar o "k" da computação. Na vida cotidiana e na física, o "k" vale 1000:

- 1 km = 1000 metros
- 1 kg = 1000 gramas
- 1 kV = 1000 volts

Entretanto, na informática, o multiplicador "k" (lê-se "quilo" ou "ká") vale 1024. Da mesma forma, o multiplicador "M" (lê-se "mega"), que normalmente vale 1.000.000, na computação vale: 1 M = 1024 k = 1024x1024 = 1.048.576. Portanto, 1 MB (lê-se "um megabyte") são exatamente 1.048.576 bytes. Mas para efeitos práticos, podemos dizer que 1 MB é aproximadamente 1 milhão de bytes.

O multiplicador "G" (lê-se "giga"), que normalmente vale 1 bilhão, na computação vale: 1 G = 1024 M = 1024x1024x1024 = 1.073.741.824. Portanto, 1 GB (lê-se "um gigabyte") são exatamente 1.073.741.824 bytes, mas para efeitos práticos podemos dizer que 1 GB é aproximadamente 1 bilhão de bytes.

2.5 Representação Interna de Caracteres

Os bytes são usados para representar caracteres, números, figuras, ou qualquer outro tipo de dado armazenado ou processado em um computador. Esta seção apresenta estas diversas formas de representação interna de caracteres.

Na maioria dos códigos alfanuméricos cada caractere é representado através de um byte. Por exemplo, no código ASCII (visto mais adiante) a letra 'A' é representada pelo byte "0100 0001". Uma seqüência de caracteres é expressa por uma cadeia de bytes sucessivos. Nem todos os tipos de códigos utilizam os 8 bits de um byte para a representação de caracteres.

2.5.1 Código de 6 bits

Os primeiros códigos utilizados foram os de 6 bits, que permitiam a representação de 2^6 = 64 caracteres (Figura 1), que correspondem a:

- 26 letras maiúsculas.
- 10 algarismos (0 1 2 3 4 5 6 7 8 9).
- 28 caracteres chamados especiais, incluindo SP (caractere em espaço em branco).

Bits 210 / ***543 000***	***001***	***010***	***011***	***100***	***101***	***110***	***111***
000	@	C	K	S	)	*	0	8
001	[	D	L	T	-	(	1	9
010	]	E	M	U	+	%	2	'
011	#	F	N	V	<	:	3	;
100	^	G	O	W	=	?	4	/
101	SP	H	P	X	>	!	5	.
110	A	I	Q	Y	&	´	6	"
111	B	J	R	Z	$	\	7	-

Figura 1. Código de 6 bits.

Por exemplo, a codificação da frase "OLA!" é: X010100 X010001 X000110 X010101.

2.5.2 Códigos de 7 bits (ASCII)

Com o desenvolvimento das linguagens de programação de alto nível começaram a ser utilizados os códigos de 7 bits que permitiam a representação de minúsculas e de

caracteres cujo significado são ordens de controle para periféricos. Um exemplo desse tipo de códigos é o ASCII de 7 bits (Figura 2).

Bits 3210	***654 000***	***001***	***010***	***011***	***100***	***101***	***110***	***111***
0000	NUL	DLE	SP	0	@	P	\	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	´	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	I	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Figura 2. Código ASCII de 7 bits.

Os significados dos caracteres de controle são:

NUL *Null* (caractere nulo).
SOH *Start of heading* (início de cabeçalho).
STX *Start of text* (início de texto).
ETX *End of text* (fim de texto).
EOT *End of transmition* (fim de transmissão).
ENQ *Enquiry* (solicitação de transmissão).
ACK *Acknowledge* (reconhecimento de transmissão).
BEL *Bell* (sinal sonoro, campainha ou alarme).
BS *Backspace* (retrocesso).
HT *Horizontal tabulation* (tabulação horizontal).
LF *Line feed* (avanço de linha).
VT *Vertical tabulation* (tabulação vertical).
FF *Form feed* (avanço de página).
CR *Carriage return* (retorno de carro).
SO *Shift out* (desligar deslocador de bits).
SI *Shift in* (acionar deslocador de bits)
DLE *Data link escape* (escape de ligação de dados).
DC1 *Device control 1* (controle de dispositivo 1).
DC2 *Device control 2* (controle de dispositivo 2).
DC3 *Device control 3* (controle de dispositivo 3).
DC4 *Device control 3* (controle de dispositivo 4).
NAK *Negative acknowledge* (transmissão negativa).
SYN *Synchronous idle* (espera sincronizada).
ETB *End of transmission block* (fim de bloco de transmissão).
CAN *Cancel* (cancelar).
EM *End of medium* (final de meio básico de dados).
SUB *Substitute* (substituir).
ESC *Escape* (escape).
FS *File separator* (separador de arquivos).
GS *Group separator* (separador de grupos).
RS *Record separator* (separador de registos).
US *Unit separator* (separador de unidades).
DEL *Delete* (apagador).

ASCII é uma codagem a 7 bits, mas muitos computadores manipula uma quantidade de 8 bits (byte). Portanto, os caracteres ASCII devem ser freqüentemente armazenados um por byte, com o bit mais significante igual a 0. O bit extra é algumas vezes usado para propósitos específicos, dependendo da aplicação. Por exemplo, algumas impressoras reconhecem um ASCII estendido, com os caracteres adicionais iniciando pelo bit mais significativo a 1. Estes caracteres habilitam a impressora a imprimir símbolos adicionais, como o alfabeto grego ou fontes do tipo itálico.

2.5.3 EBCDIC

O EBCDIC (*Extended Binary Coded Decimal Interchange Code*) é uma codagem de caracteres de 8 bits (Figura 3) e se trata de um padrão proprietário desenvolvido pela IBM.

Bits 3210	*7654 0000*	*0001*	*0010*	*0011*	*0100*	*0101*	*0110*	*0111*	*1000*	*1001*	*1010*	*1011*	*1100*	*1101*	*1110*	*1111*
0000	NUL	DLE	DS		SP	&	-						{	}	\	0
0001	SOH	DC1	SOS						a	j	~		A	J		1
0010	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	ETX	DC3							c	l	t		C	L	T	3
0100	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	LC	BS	EOB	UC					f	o	w		F	O	W	6
0111	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000		CAN							h	q	y		H	Q	Y	8
1001	RLF	EM						\	I	r	z		I	R	Z	9
1010	SMM	CC	SM		¢	!	|	:							
1011	VT				.	$	´	#								
1100	FF	IFS		DC4	<	*	%	@								
1101	CR	IGS	ENQ	NAK	(	)	-	´								
1110	SO	IRS	ACK		+	;	>	=								
1111	SI	IUS	BEL	SUB	|	¬	?	"								

Figura 3. Código EBCDIC de 8 bits.

O significados dos caracteres de controle do EBCDIC são:

NUL *Null*
SOH *Start of heading*
STX *Start of text*
ETX *End of text*
PF *Punch off*
HT *Horizontal tabulation*
LC *Lower case*
DEL *Delete*
RLF *Reverse line feed*
SMM *Start of manual message*
VT *Vertical tabulation*
FF *Form feed*
CR *Carriage return*
SO *Shift out*
SI *Shift in*
DLE *Data link escape*
DC1 *Device control 1*
DC2 *Device control 2*
DC3 *Device control 3*
RES *Restore*
NL *New line*
BS *Backspace*
IL *Ilde*
CAN *Cancel*
EM *End of medium*
CC *Cursor control*
IFS *Interchange file separator*
IGS *Interchange group separator*
IRS *Interchange record separator*
IUS *Interchange unit separator*
DS *Digit select*
SOS *Start of significance*
FS *Field separator*
BYP *Bypass*
LF *Line field*
EOB *End of block*
ESC *Escape*
SM *Set mode*
ENQ *Enquiry*
ACK *Acknowledge*

2.5.4 ASCII Estendido

É um conjunto de códigos que estende o conjunto ASCII básico. Os caracteres ASCII estendido usam 8 bits para representar os caracteres. Os caracteres extras representam caracteres de línguas mortas e caracteres especiais para desenhas figures. Veja estes caracteres na figura abaixo.

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	⊛	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Figura 4. Caracteres ASCII estendidos

2.5.5 ISO Latin-1

ISO Latin-1 é uma codificação de caracteres padronizada pela Organização Internacional de Padronização (ISO), que recebe o código ISO 8859-1. Ele é um conjunto de caracteres ASCII estendido e é muito similar aos caracteres ANSI usado pelo Windows, embora os dois não sejam idênticos. A linguagem HTML (*Hypertext Meta Language*) adota também esta representação de caracteres.

A figura abaixo apresenta os caracteres ISO Latin-1, com os caracteres adicionais propostos pelo Microsoft® Windows Latin-1 Added Characters (em claro).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80			‚	*ƒ*	„	…	†	‡	ˆ	‰	Š	‹	Œ			
90		‘	’	“	”	•	–	—	˜	™	š	›	œ			Ÿ
A0		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Figura 5. Caracteres ISO Latin-1

2.5.6 Caracteres ANSI

O Windows 9x suporta o conjunto de caracteres ANSI (*American National Standards Institute*), que é uma representação de 8 bits (256 caracteres), numerados de 0 a 255. Os valores de 0 a 127 são os mesmos dos caracteres ASCII. Valores entre 128 a 255 são similares ao conjunto de caracteres ISO Latin-1, mas naturalmente tem extensões e incompatibilidades.

2.5.7 Caracteres Unicode

Windows NT usa o conjunto de caracteres Unicode 16-bits, que cobre grande parte dos caracteres das maiores línguas vivas, incluindo também caracteres de linguagens mortas que tem muito uso escolar. Veja detalhes em http://www.unicode.org.

2.6 Representação Interna de Números

Esta seção apresenta estas diversas formas de representação interna de números.

2.6.1 Representação de Números Inteiros

Para representar números positivos, utiliza-se normalmente o valor do próprio número binário. Por exemplo, o número 6 é representado por 0101 e o número 12 é representado por 1100.

Existem quatro maneiras de se representar números negativos: módulo e sinal (MS); complemento de 1 (C-1), complemento de 2 (C-2) e excesso de 2 elevado a (N-1)

Nessas representações de números utiliza-se o sistema binário e considera-se que temos um número limitado de bits para cada dado numérico. Esse número de bits disponíveis é representado nesta seção por N.

Módulo e Sinal (MS)

Neste sistema de representação, o *bit* que está situado mais à esquerda representa o sinal, e o seu valor será 0 para o sinal + e um para o sinal -. Os *bits* restantes (N-1) representam o módulo do número. Por exemplo, supondo que exista a limitação de 8 *bits* (N=8), o valor 00101010 representa o número +42 e o valor 10101010 representa o número -42.

Denomina-se **amplitude** ou **faixa** (*range*) de representação num determinado método o conjunto de números que podem ser nele representados. Para o sistema módulo e sinal, a faixa de representação para N dígitos é de $-2^{N-1}+1 \leq X \leq 2^{N-1}-1$. Assim:

- Para o caso de 8 bits (byte), a faixa é: $-127 \leq X \leq 127$
- Para 16 bits (word), a faixa é: $-32767 \leq X \leq 32767$
- Para 32 bits (double word), a faixa é: $-2147483647 \leq X \leq 2147483647$

A vantagem deste sistema em relação a outros é a de possuir faixa simétrica. Por outro lado, apresenta a inconveniência de possuir duas representações para o número 0. Para 8 bits o 0 tem as seguintes representações: 00000000 (+0) e 1000000 (-0).

Complemento de 1 (C-1)

Este sistema de representação também utiliza o *bit* mais à esquerda para o sinal, correspondendo o 0 ao sinal + e o 1 ao sinal -. Para os números positivos, os N - 1 *bits* da direita representam o módulo (assim como no MS). O simétrico de um número positivo é obtido pelo complemento de todos os seus dígitos (trocando 0 por 1 e vice-versa), incluindo o *bit* de sinal. Por exemplo, supondo que exista a limitação de 8 *bits* (N=8), o valor 00101010 representa o número +42 e o valor 11010101 representa o número -42.

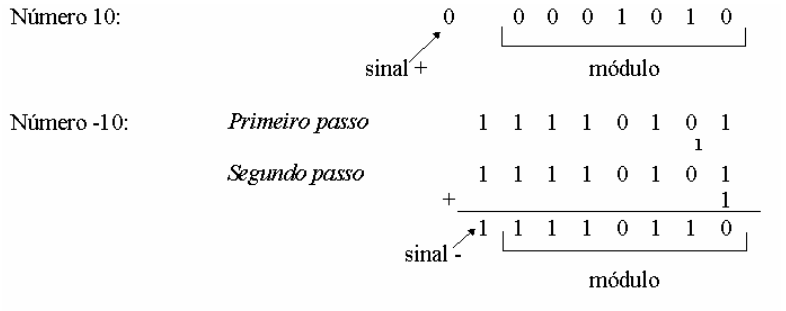
Este método tem a mesma faixa de representação para N dígitos do método MC, que é de $-2^{N-1}+1 \leq X \leq 2^{N-1}-1$. E tem a mesma desvantagem do anterior, que é de ter duas representações para o número 0: 00000000 (+0) e 11111111 (-0).

Complemento de 2 (C-2)

Este sistema também utiliza o *bit* mais à esquerda para o sinal, correspondendo o 0 ao sinal + e o 1 ao sinal -. Para os números positivos, os N-1 dígitos da direita representam o módulo (igualmente ao MS e C-1). O simétrico de um número é obtido em dois passos:

- *Primeiro passo*: Obtém-se o complemento de todos os bits do número positivo (trocando 0 por 1 e vice-versa) incluindo o bit do sinal, isto é, executa-se o Complemento de 1.
- *Segundo passo*: Ao resultado obtido no primeiro passo soma-se 1 (em binário), desprezando-se o último transporte, se existir.

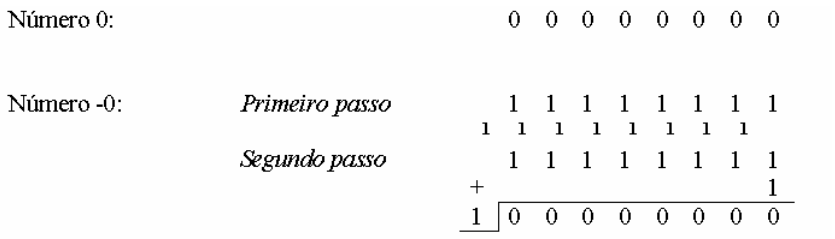
Vejamos a representação em Complemento de 2 dos números 10 e -10 para 8 bits:



A faixa de representação neste caso é assimétrica, o que constitui o seu maior inconveniente, e é dada pela fórmula $-2^{N-1} \leq X \leq 2^{N-1}-1$

- Para o caso de 8 bits (byte), a faixa é: $-128 \leq X \leq 127$
- Para 16 bits (word), a faixa é: $-32768 \leq X \leq 32767$
- Para 32 bits (double word), a faixa é: $-2147483648 \leq X \leq 2147483647$

A principal vantagem é ter uma única representação para o número 0. Para 8 bits, teremos:



O último transporte é desprezado. Portanto, o 0 e o -0 têm uma mesma representação.

O método C-2 é o mais utilizado para representar números negativos.

Excesso de 2 elevado a (N-1)

O método de representação em excesso não utiliza nenhum *bit* para o sinal, de modo que todos os *bits* representam um módulo ou valor. Esse valor corresponde ao número representado mais um excesso, que para N *bits* é igual a 2 elevado a N-1. Por exemplo, para 8 bits o excesso é 128 ($2^7 = 128$), logo, o número 10 é representado por 10+128 = 138 (10001010). O número -10 é representado por -10+128 = 118 (01110110). Neste método, o número 0 tem uma única representação, que para 8 bits corresponde a: 0+128 = 128 (10000000)

A faixa de representação é assimétrica (o que é um inconveniente) e é dada da mesma forma que o método C-2: $-2^{N-1} \leq X \leq 2^{N-1}-1$

É interessante observar que todo o número representado em excesso é igual à sua correspondente representação em Complemento de 2, exceto que o bit de sinal é invertido.

2.6.2 Vírgula fixa (Fixed Point)

Este nome vem da posição em que se supõe estar situado o ponto decimal, que é uma posição fixa. A vírgula fixa é hoje utilizada exclusivamente para a representação de números inteiros, supondo-se a vírgula decimal implicitamente situada à direita dos dígitos.

Existem quatro maneiras de representar números com vírgula fixa: binário puro, decimal, decimal não compactado, decimal compactado.

Binário Puro

O número é representado através dos métodos vistos anteriormente. Por exemplo, considerando um computador com palavra de 32 bits que utiliza o método Complemento de 2 (C-2), qual é a sua faixa de representação e qual a configuração interna dos números 0, 10, -10, 2147483647 e -2147483648?

- A faixa de representação é: $-2^{31} \leq X \leq 2^{31}-1$, ou então: $-2147483648 \leq X \leq 2147483647$.
- Representação de 0: 00000000000000000000000000000000
- Representação de 10: 00000000000000000000000000001010
- Representação de -10: 11111111111111111111111111110110
- Representação de 2147483647: 01111111111111111111111111111111
- Representação de -2147483648: 10000000000000000000000000000001

Decimal não Compactado

Neste sistema um número é armazenado com um byte para cada um de seus algarismos. Cada byte contém no seu quarteto da esquerda quatro 1's denominados bits de zona, e no quarteto da direita, o algarismo em BCD (*Binary-coded display* - codificado em binário), que é um número entre 0 e 9. Esses quatro bits são denominados bits de dígito. O quarteto da esquerda do último algarismo do número dado representa o sinal, e pode conter 1100 para o sinal + e 1101 para o sinal - (C e D em hexadecimal, respectivamente).

Por exemplo, a representação do número 1234 é 11110001 11110010 11110011 11000100, e a representação do número -2345 é 11110010 11110011 11110100 11010101.

Decimal Compactado

Cada dígito é representado num quarteto (sem bits de zona), exceto o primeiro quarteto da direita que representa o sinal com os mesmos valores (C e D).

Por exemplo, a representação do número 1234 é 00000001 00100011 11000100, e a representação do número -2345 é 00000010 00110100 11010101.

2.6.3 Ponto Flutuante

Uma vez que número de bits que representa um número real é limitado, os números reais sofrem truncamento na sua parte fracionária. É importante lembrar de que, por utilizar a vírgula flutuante, nem todos os números têm representação, razão pela qual estes números são representados de forma aproximada, acarretando pequenos erros de representação.

Números de ponto flutuante tem duas partes. A primeira parte contem a fração (algumas vezes chamada de mantissa) e a segunda parte define a posição do ponto decimal e é chamada de expoente. Por exemplo, o número decimal +6132,789 é representado em ponto flutuante como:

- Fração: +.6132789
- Expoente: +04

O valor do expoente indica que a posição real do ponto decimal é quatro pontos a direita do ponto decimal indicado na fração. Esta representação é equivalente a notação científica: $+.6132789 \times 10^{+4}$.

Generalizando, os números decimais ponto flutuante são representados na forma $Fx10^{E}$, onde F é a fração e E o expoente. Apenas a fração e o expoente são fisicamente representados em termos computacionais. A base 10 e o ponto decimal da fração são assumidos e não são mostrados explicitamente. Um número binário ponto flutuante é representado de uma maneira similar, exceto que ele usa a base 2 para o expoente. Por

exemplo, o número binário +1001.11 é representado por uma fração de 8 bits (01001110) e um expoente de 6 bits (000100).

Um número ponto flutuante é dito normalizado se o dígito mais significativo da fração não é zero. Por exemplo, a fração decimal 0.350 é normalizada, mas 0.0035 não é. Números normalizados fornecem a melhor precisão para números ponto flutuante.

O intervalo representado por um ponto flutuante é determinado pelo número de dígitos do expoente e a precisão pelo número de dígitos da fração. Para estudar as propriedades deste método de representação de números, considere uma representação R, que comporta uma fração sinalizada de três dígitos no intervalo 0,1?|f|?1 (ou zero) e um expoente sinalizado de dois dígitos. Esta representação permite expressar números nas seguintes regiões

- Números negativos entre $-0,999*10^{+99}$ até $-0,1*10^{-99}$.
- Números positivos entre $+0,100*10^{-99}$ até $+0,999*10^{+99}$.
- Zero

Os números fora desta faixa não podem ser representados.

Os números em vírgula flutuante expressos segundo a representação R podem ser utilizados para modelizar os números reais da matemática, mas ele impõem alguns problemas: os números reais fora das faixas apresentadas acima não podem ser representados. Se por exemplo a soma de dois números positivos ultrapassar a $+0,999*10^{+99}$, tem-se o que se chama de *overflow* (ultrapassagem do valor superior). Este erro é devido a natureza finita dos computadores. No caso de se somar dois números negativos e o resultado ultrapassar $-0,999*10^{+99}$, tem-se o que é chamado de underflow.

Uma outra diferença importante entre os reais e os ponto flutuante é sua densidade. Entre dois números reais distintos, x e y, existe sempre um outro número real, tão próximo que sejam x e y. Esta propriedade vem do fato que para todo par de números reais distintos x e y, z=(x+y)/2 é um número real, de valor intermediário entre x e y. Os números reais formam uma continuidade. Ao contrário, os números em vírgula flutuante não formam uma continuidade; no caso da representação R acima, não se pode expressar mais que 179100 números positivos diferentes, 179100 números negativos e 0, ou seja ao total 358201 números. Por exemplo, $+0,100*10^{3}$ dividido por 3 não pode ser expresso exatamente no nosso sistema de representação. Nós tomamos o número mais próximo que se pode representar: é realizado o arredondamento.

Normalização de um ponto flutuante

Seja a representação do número 432, na exponenciação em base 2, expresso sob a forma : 0 1010100 | 0000000000011011. Sendo o primeiro bit representando o sinal (+) da fração, os 7 bits seguinte representam o expoente codificado em excedente a 64 (84-64=20), e os 16 bits seguintes representando a fração. Observe que o valor da fração é dado por $0x2^{-1}+0x2^{-2}+0x2^{-3}+0x2^{-4}+0x2^{-5}+0x2^{-6}+0x2^{-7}+0x2^{-8}+0x2^{-9}+0x2^{-10}+0x2^{-11}+1x2^{-12}+1x2^{-13}+0x2^{-14}+1x2^{-15}+1x2^{-16}$. Portanto, o número 432 é expresso como $2^{20}x(1x2^{-12}+1x2^{-13}+1x2^{-15}+1x2^{-16}) = 432$.

Para normalizar o número acima, deve-se deslocar a fração 11 bits a esquerda e se subtrai 11 do expoente. Assim obtém-se a seguinte representação 0 1001001 | 1101100000000000.

A representação ponto flutuante: IEEE 754

Até o início dos anos 80, cada computador tinha sua própria forma de representação de números de ponto flutuante. Para eliminar esta situação, um comitê da IEEE (*Institute of Electrical and Electronics Engineers*) foi criado para definir um padrão para os cálculos aritméticos ponto flutuante. Sua meta não era somente permitir a troca de números entre computadores diferentes, mas sobretudo oferecer um modelo funcional preciso aos construtores de computadores. Os resultado destes trabalhos conduziram ao padrão IEEE 754.

O padrão IEEE define três formas de representação de ponto flutuante: a precisão simples (32 bits), precisão dupla (64 bits) e a precisão estendida (80 bits). Este último formato é destinado sobretudo para reduzir os erros de arredondamento em cálculos; eles são encontrados principalmente nas unidades de cálculo flutuante. O processador Pentium III suporta estes três precisões. Os formatos de simples e dupla precisão utilizam o binário para codificar a fração e o expoente. Eles são representados na Figura 6.

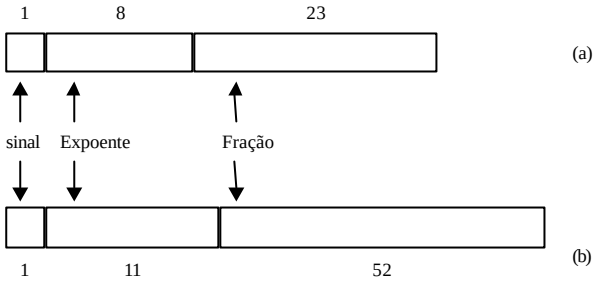


Figura 6. Formato dos números flutuante segundo o padrão IEEE: (a) Precisão simples (b) Precisão dupla

Como para todos os números, cada formato começa com um bit de sinal, que vale 0 para os números positivos e 1 para os números negativos. Em seguida vem o expoente, codificado em excedente a 127 para a precisão simples e em excedente a 1023 para a precisão dupla. Os expoentes variam de 0 a 255 ou 2047. Os números tendo como expoente os valores mínimos ou máximos acima tem uma especificidade própria e não são normalizado como os outros. Isto será visto mais adiante. Enfim, o último componente do formato, a fração, é codificada em binário de 23 ou 52 bits.

Uma fração é dita normalizada quando o primeiro bit que segue a vírgula vale 1. Considerando que o primeiro bit da fração é sempre igual a 1, o padrão define uma fração cuja significação difere um pouco das frações habituais: uma fração IEEE compreende um bit pressuposto a 1, que se chama bit escondido, após 23 ou 52 bits de valor. A vírgula também é implícita. O valor numérico da fração, para a precisão simples, é calculado da seguinte forma: $1x2^{0} + b_{22}x2^{-1} + b_{21}x2^{-2} + b_{20}x2^{-3} + b_{19}x2^{-4} + b_{18}x2^{-5} + b_{17}x2^{-6} + b_{16}x2^{-7} + b_{15}x2^{-8} + b_{14}x2^{-9} + b_{13}x2^{-10} + b_{12}x2^{-11} + b_{11}x2^{-12} + b_{10}x2^{-13} + b_{9}x2^{-14} + b_{8}x2^{-15} + b_{7}x2^{-16} + b_{6}x2^{-17} + b_{5}x2^{-18} + b_{4}x2^{-19} + b_{3}0x2^{-20} + b_{2}x2^{-21} + b_{1}x2^{-22} + b_{0}x2^{-22}$. Assim, os números reais associados aos pontos flutuantes de precisão simples são calculados da seguinte maneira: $(-1)^{S} \times 2^{(E - 127)} \times (1,F)$.

Quando todos os bits são 0, seu valor decimal é igual a 1,0. Se todos os bits são a 1, o valor da fração é igual a 2,0. Para evitar ambigüidade entre os formatos convencionais de representação de fração, no padrão IEEE diz-se pseudo-fração e não fração. Assim, os números normalizados tem uma pseudo-fração variando em 1?s?2.

Alguns exemplos de números flutuantes na precisão simples são apresentados abaixo:

- $0,5 = (-1)^{0} \times 2^{-1} \times (1,0) = 0,5x1$
  - sinal: 0
  - expoente: 127-1 = 126 = 01111110
  - fração: 1 = 1 00000000000000000000000 (primeiro 1 é implícito)
  - 0,5 =00111111000000000000000000000000
  - 0,5 = 3F0000000H
- $1 = (-1)^{0} \times 2^{0} \times (1,0) = 1x2^{0}$
  - sinal: 0
  - expoente: 127+0 = 127 = 01111111
  - fração: 1 = 1 00000000000000000000000 (primeiro 1 é implícito)
  - 1 =00111111100000000000000000000000
  - 1 = 3F8000000H
- $-1,5 = 1x2^{0} = (-1)^{1} \times 2^{0} \times (1,5)$
  - sinal: 1

- expoente: 127+0 = 127 = 01111111
- fração: 1 = 1 10000000000000000000000 (primeiro 1 é implícito)
- -1,5 =10111111110000000000000000000000
- -1,5 = BFC000000H

Características dos números flutuantes IEEE 754

As características dos números flutuantes representados segundo o padrão IEEE são apresentados na tabela abaixo.

	Precisão simples	Precisão dupla
Bit de sinal	1	1
Bit do expoente	8	11
Bit da fração	23	52
Número total de bits	32	64
Codagem do expoente	Excesso de 127	Excesso de 1023
Variação do expoente	-126 a +127	-1022 a +1023
Menor número normalizado	2^{-126}	2^{-1022}
Maior número normalizado	Aprox. 2^{+128}	Aprox. 2^{+1024}
Escala de número decimais	Aprox. 10^{-38} a 10^{+38}	Aprox. 10^{-308} a 10^{+308}
Menor número não normalizado	Aprox. 10^{-45}	Aprox. 10^{-324}

Underflow

O que fazer quando o resultado de um cálculo é inferior ao menor número ponto flutuante normalizado que se pode representar? Existem duas soluções:

- dizer que o número vale zero (arredondamento), sem outra indicação
- gerar um desvio para causar uma ultrapassagem da borda inferior (*underflow*)

Nenhuma das abordagens acima é satisfatória. É por isso que o conceito de número não normalizado aparece no padrão IEEE. Os números não normalizados tem seus expoentes iguais a zero e a fração não é mais normalizada. Isto significa que não há mais o bit implícito a 1. A fração é codificada unicamente sobre 23 ou 52 bits, ela evolui então de 0 a 1 (e não de 1 a 2 como na pseudo-fração).

O menor número que se pode representar em precisão simples tem um expoente igual a 1 e a fração constituída de zeros, isto é o número $1,0*2^{-126}$. O maior número não normalizado tem seu expoente que todo a zero (-127), e todos os bits da fração iguais a 1, isto é o número $0,9999999*2^{-127}$. O menor número não normalizado tem uma fração em precisão simples com 22 bits a zero e um bit a 1, o mais a direita. Neste caso, o expoente representa 2^{-127} e a fração 2^{-23}, que corresponde ao número 2^{-150}. É assim que os números não normalizados existem afim de permitir uma ultrapassagem gradual para baixo para as operações produzindo resultados inferiores ao menor número normalizado, em vez de substituí-los por zero.

Representação do zero

Na representação IEEE existem duas representações para o zero: +0 e -0. Seus bits de sinal valem 0 ou 1. Seus expoentes valem 0 e todos os bits da fração são iguais a zero. Assim, na precisão simples, o valor zero corresponde a :

- 0 00000000 00000000000000000000000
- 1 00000000 00000000000000000000000

Overflow

As ultrapassagens de borda a esquerda são difíceis de serem geradas e não há nenhuma combinação particular de bits para representá-los. Uma representação específica é reservada ao valor do maior número possível que se possa representar. Diz-se que é infinito. O expoente deste número é composto de bits a 1, sua fração é composta de bits a zero. Ou seja, o infinito é representado por

0 ou 1	11111111	00000000000000000000000

Este número particular pode ser visto como um operando sobre o qual se aplicam o conjunto de regras de cálculo sobre os grandes números (ou números infinitos). Por exemplo, a soma de um número infinito com um número qualquer resulta em infinito. Da mesma maneira, a divisão de um número finito pelo infinito resulta em zero e a divisão de um número finito por zero resulta infinito.

O que se pode dizer da divisão de um número infinito por um número infinito? Neste caso o resultado é indefinido. Uma representação particular foi definida para isto: NaN (*Not a Number*), que é igual a

0 ou 1	11111111	Toda configuração menos todos a zero

2.7 Representação Digital de Áudio, Imagem e Vídeo

Esta seção apresenta como imagens, áudios e vídeos são capturados do mundo real a partir de sinais analógicos e como estes sinais são transformados numa forma digital.

2.7.1 Sinais Analógicos para representar informações

Esta seção apresenta uma introdução às formas de onda e sinais analógicos e seus uso na descrição de informações detectadas pelos sentidos humanos.

Informações percebidas e variáveis físicas

Informações que os sentidos humanos podem detectar podem ser descritas como uma ou várias variáveis físicas cujos valores podem ser funções do tempo e do espaço. Note que por informações é entendido aqui como a representação física em termos de estímulos para um sentido humano, e não o conteúdo semântico, que é o significado deste estímulo.

Descrevendo sons com formas de onda

Um som, que atravessa o ar, é uma onda de ar comprimido ou expandido cuja pressão altera no tempo e espaço. Na posição de um locutor ou de um detector, os sons podem ser descritos por valores de pressão que variam apenas com no tempo (valores dependentes do tempo). O padrão de oscilação, como mostrado na Figura 7, é chamado de forma de onda (*waveform*). A forma de onda é caracterizado por um **período** e **amplitude**. O período é o tempo necessário para a realização de um ciclo; intervalo de tempo que, num fenômeno periódico, separa a passagem do sistema por dois estados idênticos. A **freqüência** é definida como o inverso do período e representa o número de períodos em um segundo. A freqüência é normalmente medida em Hz (Hertz) ou ciclos por segundo (cps). A amplitude do som é define um som leve ou pesado.

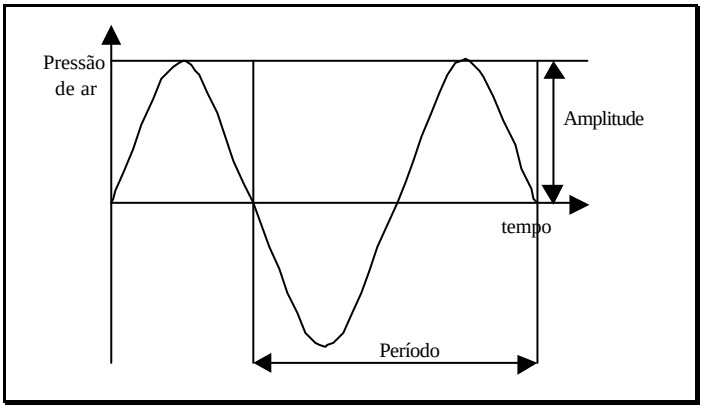


Figura 7. Forma de onda

Como a onda de som ocorre naturalmente, ela nunca é perfeitamente suave ou uniformemente periódica como a forma de onda da Figura 7.

Descrevendo imagens monocromáticas com variáveis físicas

As imagens refletem radiações eletromagnéticas (luz) incidentes que estimulam os olhos do observador. A intensidade de luz é uma função da posição espacial do ponto refletido sob a imagem. Portanto, a imagem pode ser descrita pelo valor da intensidade de luz que é uma função de duas coordenadas espaciais. Se a cena observada não foi plana, uma terceira coordenada espacial é necessária.

Descrevendo imagens coloridas com formas de onda

Se a imagem não é monocromática, ela reflete diferentes comprimentos de onda. Assim uma função simples não é suficiente para descrever imagens coloridas, mas um espectro completo de comprimento de onda refletida, cada um com sua própria intensidade, é necessário. Assim as imagens teriam que ser descritas pela conjunção de várias funções bidimensionais. Felizmente, o sistema visual humano tem certas propriedades que simplificam a descrição de imagens coloridas.

A Luz que consiste em uma distribuição espectral de intensidade estimula o sistema visual e cria uma resposta. A resposta nos olhos depende da sensibilidade do sistema visual aos comprimentos de onda. Testes realizados mostram que diferentes distribuições espectrais da luz pode dar a mesma resposta visual. Em outras palavras, é possível criar sensações de cores idênticas com diferentes combinações de comprimentos de onda (isto é diferentes combinações de cores).

A teoria da cor foi desenvolvida por Thomas Young no início de 1802 e afirma que qualquer sensação de cor pode ser reproduzida pela mistura em proporções apropriadas de três luzes coloridas monocromáticas primárias. Cores primárias são independentes no sentido que uma cor primária não pode ser obtida misturando outras duas cores primárias. A *Commission Internationale de l'Eclairage* (CIE) recomendou o uso de uma tripla particular de luz monocromática. Cada fonte de luz é definida pelo seu comprimento de onda (?1 = 70 nm, vermelho; ? 2 = 546.1 nm, verde; ?3 = 435.8 nm, azul). Em vez de ser descrita por uma infinidade de funções bidimensionais, qualquer imagem colorida plana pode ser representada por um conjunto de três funções bidimensionais.

2.7.2 Porque Digitalizar?

A utilização de informações na forma digital trás as várias vantagens.

Universalidade de representação

Sistemas computacionais manipulam apenas dados digitais. Quando áudio, imagens, vídeos estão na forma digital, eles podem ser facilmente armazenados e manipulados (processados, transmitidos e apresentados) pelos sistemas computacionais tal qual outros dados. Desta forma, como todas as mídias de apresentação (textos, imagens, som, etc.) são codificadas numa única forma, elas podem ser manipuladas de uma mesma forma e pelo mesmo tipo de equipamento. Além disso, informações de áudio e vídeo digitalizadas são facilmente integradas com outros tipos de dados e são de fácil interação com mídia digitais usando sistemas computacionais.

Processamento

Informações de áudio e vídeo digitais são processadas, analisadas, modificadas, alteradas, ou complementadas por programas de computador tal qual outros dados. A seguir são apresentados alguns exemplos de processamentos possíveis graças a representação digital de informações de áudio e vídeo [Fluckiger, 95]:

- reconhecimento de conteúdos semânticos (voz, escrita a mão, formas e padrões);
- estruturas de dados, ligações usando apontadores entre elementos de informações podem ser criados para rápida obtenção de informações;
- editores poderosos com funções cut-and-paste para criar monomídia (p.e. som apenas) ou documentos multimídia são possíveis;

- qualidade da informação pode ser aumentada pela remoção de ruídos ou erros como a digitalização de velhos discos de vinil para criar CD's de alta qualidade;
- informações sintetizadas e vídeos podem ser mixadas

Qualidade

Sistemas digitais são mais confiáveis. Sinais digitais são mais tolerantes a ruídos e interferências que os analógicos. Na forma analógica, o valor do sinal é alterado se há ruídos ou interferências. Este erro é acumulativo de um estágio para outro do sistema. Na forma digital, há apenas dois níveis de sinal: alto (1) ou baixo (0). Se o erro causado pela interferência ou ruído é abaixo de um certo limiar, o sinal pode ser reconhecido corretamente. Além disso, em cada estado do processamento digital ou transmissão, o sinal digital são reconstruídos, assim erros não são acumulativos.

Segurança

Se segurança na comunicação é necessária, a representação digital da informação facilita a criptografia.

Armazenamento

A utilização unicamente de mídias digitais permite a existência de um dispositivo único de armazenamento de dados para todas as mídias, sendo que diferenças podem estar ligadas a requisitos de tamanho. Imagens e vídeos necessitam de uma maior capacidade de armazenamento que textos ou gráficos. Som necessita de uma capacidade de armazenamento um pouco menor que imagens.

Dispositivos digitais apropriados podem ser necessários, tal como CD-ROMs (*Compact Disk-Read Only Memories*).

Transmissão

Qualquer sistema de comunicação de dados podem ser (potencialmente) utilizado para a transmissão de informações de áudio e vídeo digitais. Uma única rede de comunicação suportando a transmissão digital das informações multimídia é possível (Rede Digital de Serviços Integrados). A este nível existem dificuldades causados pelos requisitos de certas aplicações, em particular aquelas que necessitam o respeito da fidelidade de dependências temporais dos sinais digitais.

A vantagem da transmissão digital em relação a transmissão de sinais analógicos é que ela é menos sensíveis a ruídos, a detecção de erros, recobrimentos e a criptografia são facilitadas.

2.7.3 Digitalização, Amostragem e Quantificação

Esta seção apresenta a forma de digitalização dos vários tipos de mídias de apresentação. Digitalização aqui é o processo envolvido na transformação de sinais analógicos em sinais digitais:

- **Sinal analógico** é uma medida física que varia continuamente com o tempo e/ou espaço. Eles são descritos por uma função dependente apenas do tempo (s=f(t)), dependente apenas do espaço (s=f(x,y,z)), ou dependente do tempo e do espaço (s=f(x,y,z,t)). Sinais analógicos são produzidos por sensores que detectam fenômenos físicos (que simulam os sensos humanos) e os transformam em uma medida que toma a forma de uma corrente ou tensão elétrica. A precisão é ditada pelas características dos sensores.
- **Sinais digitais** são seqüências de valores dependentes do tempo ou do espaço codificados no formato binário.

Para a conversão de sinais analógico em digital é necessário a realização de três passos: amostragem, quantificação e codificação. A figura 1 ilustra o processo de digitalização de um sinal analógico no domínio do tempo.

Amostragem

Nesta etapa um conjunto discreto de valores analógicos é amostrado em intervalos temporais (p.e., para sons) ou espaciais (p.e., para imagens) de periodicidade constante, como apresentado na figura 1a. A freqüência de relógio é chamado de taxa de amostragem ou freqüência de amostragem. O valor amostrado é mantido constante até o próximo intervalo. Isto é realizado através de circuitos *sampling and hold*. Cada uma das amostras é analógica em amplitude: ele tem qualquer valor em um domínio contínuo. Mas isto é discreto no tempo: dentro de cada intervalo, a amostra tem apenas um valor.

Segundo o teorema de Nyquist: se um sinal analógico contem componentes de freqüência até f Hz, a taxa de amostragem deve ser ao menos 2f Hz. Na prática, esta freqüência é um pouco maior que 2f Hz. Por exemplo, a taxa de amostragem de CD-audio é de 44,1 kHz, e dos tapes de áudio digital (DAT) é de 48kHz para cobrir uma faixa audível de freqüência de 20 kHz. Outro exemplo, os componentes principais de freqüência da voz humana estão dentro de 3,1 kHz, com isto os sistemas de telefonia analógicos limitam o sinal transmitido a 3.1 kHz; é comum usar uma freqüência de amostragem de 8 kHz para converter este sinal em digital.

Quantificação

O processo de converter valores de amostras contínuas em valores discretos é chamado de quantificação. Neste processo nós dividimos o domínio do sinal em um número fixo de intervalos. Cada intervalo tem o mesmo tamanho e recebe um número. Na figura 1c estes intervalos são numerados de 0 a 7. A cada amostra dentro de um intervalo é atribuído o valor do intervalo. O tamanho deste intervalo de quantificação é chamado de passo de quantificação. A técnica que utiliza o mesmo passo de quantificação é chamada modulação PCM (*Pulse Coded Modulation*). Algumas vezes, nem todos os valores amostrados são retidos após a quantificação. No caso ilustrado pela figura 1c, todos os valores amostrados foram retidos.

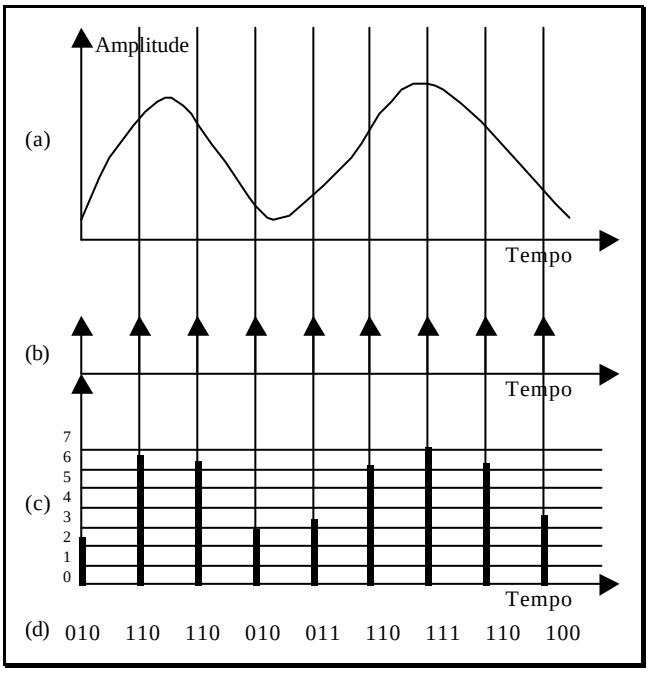


Figura 8. Conversão A/D [Lu, 96]: (a) sinal analógico; (b) pulsos de amostragem; (c) valores amostrados e intervalos de quantificação; (d) seqüência digital

Codificação

A codificação consistem em associar um conjunto de dígitos binários, chamado de *code-word*, a cada valor quantificado. No caso da figura 1d, oito níveis de quantificação são

usados. Estes níveis podem ser codificados usando 3 bits, assim cada amostra é representada por 3 bits.

Em algumas aplicações de telefonia, a digitalização da voz humana utiliza 16 bits por amostra, que então leva a 2^{16} ou 65.536 passos de quantificação. Em outras aplicações de compressão de voz, algumas vezes, apenas 8 quantificações por bits são necessários, produzindo apenas 256 passos de quantificação.

Taxa de bits

Taxa de bits é definida como o produto entre taxa de amostragem e o número de bits usados no processo de quantificação. Por exemplo, supondo uma freqüência de 8k Hz e 8 bits por amostra, a taxa de bits necessária à telefonia é igual a 8000x8 = 64 kbps.

Conversão analógica/digital e digital/analógica

Em sistemas computacionais, geralmente todas as informações são representadas internamente no formato digital. Mas humanos reagem a estímulos sensoriais físicos, assim a conversão digital-para-analógico (ou conversão D/A) é necessária na apresentação de certas informações (figura 2).

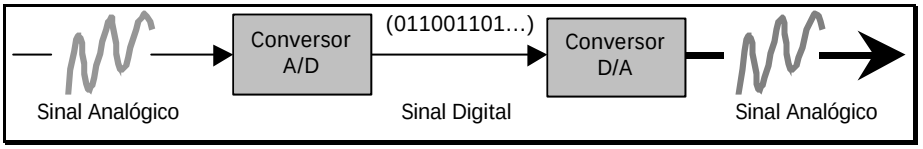
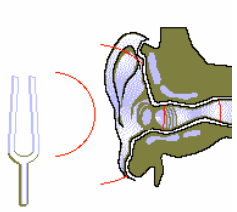


Figura 9. Conversão analógico/digital e digital/analógica

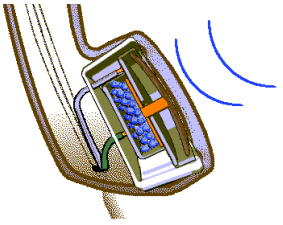
2.7.4 Áudio



Áudio é causado pelo distúrbio da pressão de ar que alcança o tímpano. Quando a freqüência do distúrbio de ar está na faixa de 20 Hz a 20.000 Hz ele é audível. A maioria dos sistemas trabalham com esta faixa de freqüência. Outro parâmetro usado para a medição do som é a amplitude (medido em decibéis - dB), variação que causa o som leve ou pesado. Por exemplo, o limiar da dor é de 100 a 120 dB.

A onda sonora é uma onda contínua no tempo e amplitude. A onda apresentada na Figura 7 pode ser um exemplo de onda sonora.

Representação digital de áudio

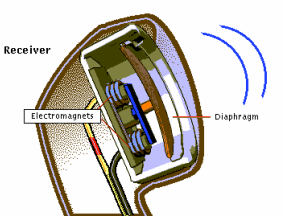


A forma de onda de áudio é convertida em um sinal elétrico contínuo (analógico) por um microfone. Este sinal elétrico é medido normalmente em volts. Para que sistemas computacionais processem e comuniquem sinais de áudio, o sinal elétrico deve ser convertido em um sinal digital. Este processo de conversão de um sinal analógico em digital foi apresentado na seção 2.7.3. O mecanismo que converte o sinal de áudio digital em analógico é chamado de Conversor Analógico para Digital (CAD).

Áudio digital necessita ser amostrado continuamente em uma taxa fixa. Cada amostra é representada por um número fixo de bits. A tabela abaixo mostra a taxa de amostragem e o número de bits usados para cada amostra para várias aplicações de áudio. Relembrando, quanto maior a taxa de amostragem e maior o número de bits por amostragem, maior é a qualidade do áudio restituído, mas com isso maior é a taxa de bits. Note na tabela que para áudio estéreo, tal como CD-Audio, dois canais são necessários.

Aplicações	Nº de canais	Taxa de amostragem	Bits por amostragem	Taxa de bits
CD-Audio	2	44.1 kHz	16	1,41 Mbps
DAT	2	48 kHz	16	1,53 Mbps
Telefone Digital	1	8 kHz	8	64 Kbps
Rádio digital, long play DAT	2	32 KHz	16	1,02 Mbps

Para a apresentação do áudio digitalizado é necessário realizar a transformação de uma representação artificial do som em uma forma de onda física audível pelo ouvido humano. Para isto são utilizados Conversores Digital para Analógico (CDA).

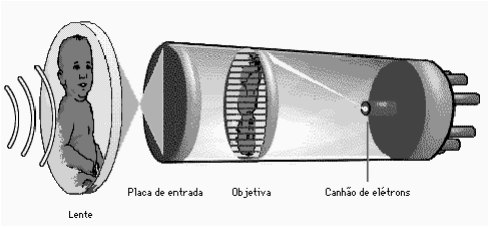


Normalmente os conversores CAD e CDA são implementados em uma única placa. Um exemplo de placa de audio é Creative Sound Blaster AWE64, possibilitando até 16 bits por amostras, produzindo áudio qualidade CD.

2.7.5 Vídeos e Imagens Analógicos

Captura e reprodução de imagens e vídeos analógicos

As imagens são capturadas usando câmeras da seguinte maneira: as lentes da câmera focam uma imagem de uma cena em uma superfície foto-sensível de sensores CCD (*Charge-Coupled Device*); o brilho de cada ponto é convertido em uma carga elétrica por uma camada foto-sensível, estas cargas são proporcionais ao brilho nos pontos; a superfície foto-sensível é rastreada por um feixe de elétrons para capturar as cargas elétricas, devendo ser feito rapidamente antes que a cena mude. Desta maneira a imagem ou cena é convertida em um sinal elétrico contínuo.



Nesta seção, por simplificação assume-se a captura e reprodução de vídeos monocromáticos, onde apenas um sinal de luminância é produzido (apenas a luminosidade é capturada, temos a imagem em tons de cinza). Neste caso são usadas **câmeras de Luminância**, que captam a imagem em tons de cinza, e gera um sinal só com a luminância da imagem. A imagem é gerada por um CCD monocromático que capta o tom de cinza que incide em cada célula do circuito. Este tipo de câmera é utilizada em geral para aplicações em visão computacional e nos casos onde a informação sobre a luminosidade da imagem é suficiente.

O dispositivo de apresentação de imagens mais utilizado é o tubo de raios catódicos (CRT). Eles são usados nos aparelhos de TV e monitores de computadores. Há uma camada de fósforo fluorescente no interior do CRT. Esta camada é rastreada por um feixe de elétrons na mesma forma do processo de captura na câmera. Quando o feixe toca o fósforo ele emite luz durante um curto instante. O brilho da luz depende da força do feixe. Quando quadros repetem-se suficientemente rápidos a persistência da visão resulta na reprodução de um vídeo. Em implementação prática, o sinal elétrico enviado da câmera para o dispositivo de apresentação deve conter informações adicionais para assegurar que o rastreamento esteja sincronizado com o rastreamento do sensor na câmera. Esta informação é chamada *sync information.*

Vídeos e Imagens Coloridos

Nesta seção é discutido a captura e a reprodução de imagens e vídeos coloridos.

Todos os sistemas de TV a cores são baseados na teoria *Tristimulus* de reprodução da cor. Esta teoria afirma que qualquer cor pode ser reproduzida com a mistura das três cores primárias: vermelho, verde e azul. Para capturar imagens coloridas, uma câmera divide a luz nos seus componentes vermelho, verde e azul. Estas três componentes de

cor são focalizados em sensores de vermelho, verde e azul, que convertem estes três componentes em sinais elétricos separados.

Em um monitor colorido, há 3 tipos de fósforos fluorescentes que emitem luzes vermelha, verde e azul quando tocadas por 3 feixes de elétrons. Estes fósforos são arranjados de tal forma que cada posição do vídeo tem 3 tipos de fósforo. A mistura da luz emitida destes 3 fósforos produz um ponto de cor.

A visão acima descreve o sinal RGB. Na realidade, o sinal analógico pode ser gerado da seguinte maneira [França Neto, 98]:

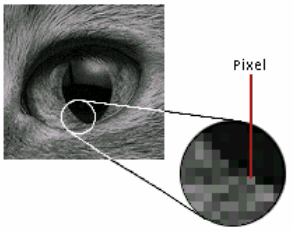
- Sinal RGB (red, green, blue) - O sinal é separado pelas cores básicas, com isso é possível ter uma imagem mais pura. Ele é utilizado em câmeras e gravadores profissionais, imagens geradas por computador, etc.
- Sinal de vídeo composto colorido - os sinais das cores (RGB) são codificados em um único sinal seguindo um determinado padrão (NTSC, PAL-M, SECAM, etc) ;
- Sinal de luminância e crominância ou Y/C (S-video) - o sinal é composto por duas partes, a luminância e a crominância; como isso a imagem tem uma melhor qualidade do que no vídeo composto. Muito usado por vídeos SVHS, laser disc, DVD e outros aparelhos que geram imagens de boa qualidade (acima de 400 linhas);

2.7.6 Representação digital de imagens e vídeos

A seção anterior discutiu muitos conceitos e nomenclaturas de vídeos analógicos. Esta seção discute a representação digital de imagens e vídeos.

Imagens Digitais

Imagens não são revisáveis porque seu formato não contêm informações estruturais. Elas podem resultar de capturas do mundo real (via escaneamento de uma página impressa ou foto, câmeras digitais) ou elas podem ser sintetizadas pelo computador (via programas de *paint*, captura da tela, conversão de gráficos em imagens bitmap). Após digitalizadas, as imagens podem ser manipuladas com editores de imagens (por exemplo, Photoshop), que não produzem documentos que retém a estrutura semântica.



Formatos de Imagens

Imagens no computador são representadas por *bitmaps*. Um bitmap é uma matriz bidimensional espacial de elementos de imagem chamados de pixeis. Um pixel é o menor elemento de resolução da imagem, ele tem um valor numérico chamado de amplitude. O número de bits disponíveis para codificar um pixel é chamado de profundidade de amplitude (ou de pixel). Exemplos típicos de profundidade de pixel é 1 (para imagens preto&branco), 2, 4, 8, 12, 16 ou 24 bits. O valor numérico pode representar um ponto preto e branco, um nível de cinza, ou atributos de cor (3 valores) do elemento de imagem em imagens coloridas.

O número de linhas da matriz de pixeis (m) é chamado de resolução vertical da imagem, e o número de colunas (n) é chamado de resolução horizontal. Denominamos resolução espacial, ou resolução geométrica, ao produto m x n da resolução vertical pela resolução horizontal. A resolução espacial estabelece a freqüência de amostragem final da imagem. Dessa forma, quanto maior a resolução mais detalhe, isto é, altas freqüências, da imagem podem ser captadas na representação matricial. A resolução espacial dada em termos absolutos não fornece muita informação sobre a resolução real da imagem quando realizada em dispositivo físico. Isso ocorre porque ficamos na dependência do tamanho físico do pixel do dispositivo. Uma medida mais confiável de resolução é dada pela densidade de resolução da imagem que fornece o número de pixels por unidade linear de medida. Em geral se utiliza o número de pixels por polegada, ppi ("pixels per inch") também chamada de dpi ("dots per inch").

Formatos bitmap necessitam mais capacidade de armazenamento do que gráficos e textos. Como bitmaps ignoram a semântica, duas imagens de mesma dimensão (altura e largura) ocupam o mesmo espaço. Por exemplo, um quadrado ou uma foto digitalizada com dimensões idênticas ocupam o mesmo espaço. Os gráficos, como eles consideram a semântica, ocupam menos espaço.

Imagens e Gráficos Animados

As imagens e os gráficos podem ser apresentados na tela do computador como uma sucessão de imagens/gráficos que podem criar a sensação de movimento.

Uma imagem ou gráfico individual de uma animação é chamado de quadro (ou *frame*). Para ser compreensível, os quadros que compõem a animação devem ser apresentados geralmente em uma taxa aproximadamente fixa. O número de quadros apresentados por segundo é definido como freqüência de quadros e é medido em termos de quadros por segundo (fps – *frames per seconds*).

A taxa de quadros de uma animação é determinada por 3 fatores:

- A taxa deve ser alta suficiente para produzir a sensação de movimento. Para isto, taxas maiores ou iguais a 25 fps devem ser utilizadas. A tabela abaixo resume as principais freqüências de quadro utilizadas atualmente.

Fps	Comentários
<10	Apresentação sucessiva de imagens
10 à 16	Impressão de movimento mas com sensação de arrancos
>16	Efeito do movimento começa
24	Cinema
30/25	Padrão de TV americana/européia
60	Padrão HDTV

- Maior a freqüência de quadros utilizada mais alto é a largura de banda necessária à transmissão. Isto pois maior a taxa, maior é o números de quadros que devem ser enviados. Portanto a rede utilizada pode ditar a freqüência de quadros a ser utilizada.
- Problema de freqüência de restauração (refreshing) de tela: a tela deve ser restaurada 50 vezes por segundo para evitar tremulações. Mas se a freqüência de quadros for 50 fps, a largura de banda necessária aumentará substancialmente. Para evitar problemas de tremulação utiliza-se vídeos entrelaçados, onde reduz-se pela metade o número de quadros requeridos por segundo, ou seja, 25 fps.

Imagens Bitmap Animadas (Vídeo)

Na animação de imagens, cenas são registradas como um sucessão de quadros representados por imagens bitmap possivelmente compactadas. Estas imagens podem ser capturadas da vida real com câmeras ou criadas através do computador. A primeira técnica produz o que é chamado de **vídeo**.

Animação de imagens tem as mesmas características que as imagens: falta de uma descrição semântica e necessidade de uma grande capacidade de armazenamento.

Gráficos Animados

O termo gráfico animado ou animação gráfica é utilizado para referenciar apresentação sucessiva de objetos visuais gerados pelo computador em uma taxa suficiente para dar a sensação de movimento e onde cada atualização é comutada de uma descrição abstrata em tempo de apresentação.

A principal vantagem das animações gráficas é que elas são mais compactas: elas são descritas por um conjunto de objetos com diretivas temporais (em outras palavras um programa a ser executado em tempo de apresentação). Outra vantagem é que animações gráficas são revisáveis. Existe uma desvantagens: é necessário um poder de processamento suficiente para apresentação.

2.7.7 Especificação da Cor

Esta seção procura apresentar a forma de especificação da cor na representação de imagens e vídeos digitais.

Propriedades da Cor

A luz visível é uma forma de radiação eletromecânica que tem um espectro de comprimento de onda variando aproximadamente de 400 nm a 780 nm. Uma luz de diferente comprimento de onda produz uma sensação de cor diferente (p.e. violeta de 380 a 450 mn, azul de 450 a 490 nm, verde de 490 a 560nm). As três propriedades físicas básicas da cor são: luminância (brilho), nuança (cor) e saturação (puridade).

Sistema de especificação de cores

Para comunicar imagens e vídeos coloridos, a cor deve ser especificada usando certo método. A CIE, Comissão Internacional de Iluminação, é o órgão responsável pela padronização na área de fotometria e colorimetria. Existem vários padrões de cor estabelecidos pela CIE, entre eles o CIE-RGB e o CIE-XYZ.

2.7.8 Sistema RGB

No sistema RGB de representação de cor, uma cor é representada pela intensidade de três cores primárias (teoria Tristimulus): vermelho (Red), verde (Green) e azul (Blue), com cada valor variando de 0 a 255. Exemplos de cores familiares são apresentadas abaixo:

- Branco = 255,255,255
- Vermelho = 255,0,0
- Verde = 0,255,0
- Azul = 0,0,255
- Amarelo = 255,255,0
- Preto = 0,0,0

A representação de imagens coloridas pode ser feita através de cores por componente (*true color*), cores indexadas, ou cores fixas. Essa representação vai depender do propósito e dos dispositivos que vão ser usados para trabalhar com essas imagens.

True Color

No True Color, cada pixel da imagem é representado por um vetor de 3 componentes de cores (RGB) com um certo número de bits para representar cada componente de cor (resolução de cor). Com isso, quanto maior for a resolução de cor mais qualidade teremos para representar as cores de cada pixel. Geralmente o número de bits para cada componente RGB é igual, ou seja quando temos um pixel sendo representado por 9 bits, usamos 3 bits para cada componente (3-3-3). Mas pode ser feito uma representação com diferentes valores para as componentes, por exemplo uma representação 8 bits/pixel, pode ser usado 3 para componentes R, 3 para G e 2 para B (3-3-2), tal representação em um byte é comumente usado e tira proveito do fato que a percepção humana da componente azul é menos sensível que as outras componentes.

O número de bits para representar cada componente fornece a quantidade de cores que podem ser representados por essa componente. Ou seja, se n é a resolução de cor então a quantidade de níveis possíveis é de 2^n níveis. Por exemplo, uma imagem colorida representada por 12 bits/pixel, com 4 bits para cada componente RGB. Temos então: 2^4=16 níveis para cada componente de cor RGB, o que nos possibilita representar até 4.096 cores diferentes (16 x 16 x 16 = 4.096) , o que é equivalente à 2^{12} = 4.096.

Temos alguns padrões de cores nesse formato que são:

Bits/pixel	Padrão	Componente de cor RGB	Máximo de cores
15 bits/pixel	High Color (15 bits)	5 bits/pixel, 32 níveis por componente	32.768 cores
16 bits/pixel	High Color (16 bits)	5/6 bits/pixel, 32/64 níveis por componente	65.536 cores
24 bits/pixel	True Color, (24 bits)	8 bits/pixel, 256 níveis por componente	16.777.216 cores

O padrão com 24 bits/pixel é o mais usado para representar com fidelidade as cores, pois o número de cores que podem ser representadas com essa resolução de cores é maior do que a visão humana pode reconhecer.

Cores Indexadas

Nas cores indexadas, cada pixel é representado por um índice que aponta para uma tabela de cores (paleta) que contem as informações sobre as cores (Figura 10). Temos então um número de cores que podem ser representadas, que é o número de entradas na paleta. A paleta por sua vez, tem em geral 24 bits para representar cada cor no formato RGB. Dessa forma podemos representar n cores de um conjunto com mais de 16 milhões de cores. Nesse caso, para representar esse tipo de imagem, as informações das cores da paleta devem constar da estrutura além das dimensões e seqüência de índices.

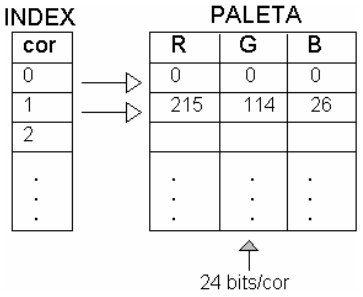


Figura 10. Índice e paleta de cores

O número de cores e a resolução de cor da paleta podem variar. Os dois padrões mais usados são apresentados na tabela abaixo.

Bits/pixel	Padrão	Resolução de cor da paleta (RGB)
4 bits/pixel	16 cores indexadas	24 bits/cor
8 bits/pixel	256 cores indexadas	24 bits/cor

Cores fixas

Nas cores fixas, cada pixel é representado por um índice que aponta para uma tabela de cores fixa. Esse sistema geralmente é usado quando o dispositivo não permite a representação de muitas cores, como no caso de placas de vídeos antigas ou padrões de cores (padrão de cores do MS Windows 3.x, 16 cores). O número de bits para representar um pixel depende do número de cores fixas, ou seja para representar por exemplo 16 cores, são necessários 4 bits/pixel.

Imagens em Tons de Cinza

A representação de imagens em tons-de-cinza é feita discretizando a informação de luminância de cada ponto da imagem. Ou seja, cada pixel contém a intensidade de luminosidade representada em um certo número de bits. Assim uma imagem com resolução de cor de 8 bits, pode representar até 256 níveis de cinza, variando do preto ao branco.

Os padrões mais usados são de 16 e 256 tons-de-cinza, 4 e 8 bits/pixel respectivamente. Representações com mais que 256 tons-de-cinza não são percebidas pela vista

humana, ou seja representar uma imagem com 256 níveis é suficiente para a maioria das aplicações.

Imagens Binárias

As imagens binárias são imagens com dois níveis, como preto e branco. São muito usadas por dispositivos de impressão e para representar imagens de documentos monocromáticos. Para representar um pixel de uma imagem binária, como o próprio nome diz, é necessário apenas 1 (um) bit. Essa informação é suficiente para representar cada pixel, ou seja temos uma representação de 1 bit/pixel. Em alguns casos, temos uma informação extra sobre a cor de cada informação, a cor para o bit com valor 0 (zero) e a cor para o bit de valor 1 (um). Essa informação de cor é geralmente é representada em 24 bits/cor no padrão RGB, podendo porém ser representada de outras formas.