Licenciatura em Tecnologias e Sistemas de Informação — 1º Ano
Sistemas de Computação

Representação da Informação no Computador - II[1]
Números Inteiros e Reais

---

1 Introdução e Objetivos

Os Computadores guardam todos os seus dados e instruções no formato binário, usando apenas grupos de 0s e 1s. No entanto todos sabemos que existem diferentes tipos de números, nomeadamente os números fracionários, os inteiros positivos e negativos, etc. Torna-se por isso necessário arranjar formas para representar estes tipos de dados no computador.

O objetivo desta sessão de trabalho é o estudo dos vários métodos para representar números inteiros e reais no computador.

2 Números Fracionários

À semelhança do que acontece com os números inteiros do sistema decimal também os números fracionários podem ser expressos em potências de dez, neste caso potências de 10 de expoente negativo.

Por exemplo o número decimal 0,27315 pode ser escrito da seguinte forma:

$$0,2 + 0.07 + 0,003 + 0,0001 + 0,00005,$$

o que é o mesmo que:

$$2 * 0,1 + 7 * 0,01 + 3 * 0,001 + 1 * 0,0001 + 5 * 0,00001,$$

que por sua vez equivale a:

$$2 * 10^{-1} + 7 * 10^{-2} + 3 * 10^{-3} + 1 * 10^{-4} + 5 * 10^{-5}$$

ou:

$$2 * 1/10^{1} + 7 * 1/10^{2} + 3 * 1/10^{3} + 1 * 1/10^{4} + 5 * 1/10^{5}$$

Exercícios:
Escreva os seguintes números decimais em desenvolvimento de potências de 10: 0,32609, 342,3, 203,095 ;

[1] **Este documento foi escrito de acordo com o Novo Acordo Ortográfico**

2.1 Conversão de binário para decimal e de decimal para binário

A conversão de números fracionários binários continua a verificar-se de acordo com os princípios gerais que temos vindo a observar. No caso da conversão de binário para decimal, importa apenas ter conta que os números a converter serão potências de 2 com expoente negativo, ou seja:

$$2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, ...$$

também representado por

$$1/2, 1/2^2, 1/2^3, 1/2^4, ...$$

Exercícios:
Converta para o sistema decimal os seguintes números binários: $0,1101_2$ e $11001,00101_2$.

Na conversão de decimal para binário, o método de extração de potências (subtrações sucessivas) mantém-se inalterável, se tivermos em conta uma vez mais que estamos a trabalhar com potências de expoente negativo. Ou seja, trabalharemos com 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, que equivale a 1/2, 1/4, 1/8, 1/16, ou ainda,0.5, 0.25, 0.125, 0.0625.

Em relação ao segundo método de conversão para decimal estudado, existe um diferença importante. Em vez de trabalharmos com divisões sucessivas pela nova base, vamos trabalhar com multiplicações. E em vez de "aproveitarmos" os restos das divisões, vamos aproveitar as partes inteiras que resultam dessas multiplicações... Neste caso, o valor da primeira multiplicação é o dígito mais significativo e o último, o menos significativo.

É importante perceber que no caso dos números inteiros, ao dividir por uma base, estamos a chegar todo o número para a direita, sendo o resto aquele dígito que ficaria à direita do ponto numérico. No caso dos fracionários, ao multiplicar, estamos a chegar todo o número à esquerda... E aqui também vamos aproveitar o que ficou do "lado de lá" do ponto numérico, neste caso a parte inteira.

Finalmente, devemos ter em atenção que a conversão de um número fracionário pode nunca estar concluída. Nessa situação deverá interromper-se o processo de conversão quando atingidos os dígitos significativos que foram pedidos.

Exercícios:
Converta para o sistema binário os seguintes números decimais: 0,825, 0,625, 0,4 e 0,32.

2.2 Conversão de hexadecimal para decimal e de decimal para hexadecimal

Da mesma forma que acontece em qualquer outro sistema de numeração, os números fracionários hexadecimais são expressos em potências de expoente negativo. Assim, por exemplo, o número hexadecimal 0,F equivale ao número 0.9375 no sistema decimal:

$$0, F = 15 * 16^{-1} = 15 * 1/16 = 15 * 0.0625 = 0.9375$$

Exercícios:
Converta para o sistema decimal os seguintes números hexadecimais: $0,1F_{16}$ e $A,3_{16}$.

Para converter a parte fracionária de um número hexadecimal usa-se também um método semelhante ao usado para converter números fracionários decimais para o sistema binário. Neste caso, multiplica-se sucessivamente a parte fracionária por 16. A parte inteira do resultado de cada multiplicação é um dígito hexadecimal do novo número, sendo o valor da primeira multiplicação o dígito mais significativo e o último, o menos significativo. O critério de paragem depende do número de dígitos significativos que pretendemos no resultado. Com os restantes sistemas de numeração aplicam-se os mesmos princípios.

Exercícios:
Converta para o sistema hexadecimal os seguintes números decimais: 0,625 e 0,32;

2.3 Conversão de hexadecimal para binário e de binário para hexadecimal

Também no caso dos números fracionários se consegue simplificar a conversão do sistema binário para o sistema hexadecimal e vice-versa. Interessa notar apenas que no caso da parte fracionária os bits agrupam-se em grupos de quatro da esquerda para a direita.

Exercícios:
Converta para o sistema hexadecimal o seguintes número binário: $10011111,101_2$;

Exercícios:
Converta para o sistema binário o seguintes número hexadecimal: $1F,E_{16}$;

2.4 Adição e subtração com números fracionários

As operações aritméticas com números fracionários binários e hexadecimais seguem basicamente as mesmas regras que as operações aritméticas com números decimais.

Exercícios:
Tendo este facto em conta, efetue as seguintes operações:

- $1001,11_2 + 100,11_2$;
- $1000,1_2 - 0,11_2$;
- $101,11_2 * 0,11_2$;
- $13,F_{16} + 0,FF_{16}$;
- $2,FF_{16} - 1E,F_{16}$;

3 Números Inteiros sem Sinal

Em aplicações em que o *output* é um *display* digital (contadores de frequências, voltímetros digitais, calculadoras, etc) são usados muitas vezes códigos de representação decimal. Um dos códigos mais conhecidos é o código BCD (*Binary Coded Decimal*).

Este código usa os primeiros 10 números do sistema binário para codificar os 10 dígitos decimais. A diferença entre o sistema binário e o código BCD só se faz sentir a partir do número decimal 9 uma vez que no código BCD os números decimais não são tratados no seu conjunto, mas sim dígito a dígito.

Exercícios:

- Converta para código BCD o número decimal 128;
- Converta para código BCD o número hexadecimal $E2_{16}$;
- Sabendo que o número 10010100_2 está em binário BCD, converta-o para hexadecimal;

4 Números Inteiros com Sinal

Até aqui temos falado sempre de números supostamente positivos. Mas do "outro lado do zero" existe a realidade dos números negativos, pelo que sempre nos habituamos à utilização do sinal + e do sinal - como formas de indicar se um número é positivo ou negativo. Ao utilizar o computador no entanto, estamos limitados pela existência de apenas dois estados, que convencionamos chamar 0 e 1, para representar qualquer tipo de informação. É com esta limitação que vamos ter de conseguir representar os sinais.

Uma das formas de o fazer é a normalmente designada por sinal e grandeza. Nesta forma de representação um dos bits - normalmente o mais significativo - é reservado para nos indicar se um número é positivo - o bit de sinal a 0 - ou se esse número é negativo - o bit de sinal a 1.

Por exemplo quando falávamos apenas de inteiros positivos, conseguíamos com um byte (8 bits) representar 256 números positivos diferentes, ou seja um total de 2^8 valores, ou seja ainda, os números de 0 a 255. Ao reservarmos o bit mais significativo para sinal, ficamos apenas com 7 bits para representar a grandeza do número, o que faz com que os 256 valores representados se dividam por 128 positivos e 128 negativos. Um dos inconvenientes desta representação está associada à existência de duas representações distintas para a mesma quantidade (+0 e -0) e a uma complexidade significativa na realização de operações, já que os algoritmos a utilizar variam em função do sinal dos operandos.

A possibilidade de representar números positivos e negativos sem recorrer ao uso do sinal deu origem a outras formas de representação. Por exemplo a designada por complemento para 9, usada para representar números inteiros negativos e positivos em base 10.

Peguemos em 3 dígitos. As representações podem ir de 000 a 999, um total de mil números, sendo 500 utilizados para representar inteiros positivos e 500 para representar inteiros negativos. As primeiras 500 representações correspondem a números inteiros de 0 a 499 - a representação 0 representa o número 0, a representação 499 representa o número 499 e, por exemplo a representação 369 representa o número 369.

Ficamos agora com as representações 500 a 999 para representar números negativos. Nesta situação, para números negativos, dizemos que a representação de um número negativo é obtido pela subtração do seu valor a 999. Ou seja "obtemos o seu complemento para 9", quase como se fosse um espelho.

Desta forma, o número -136 tem a representação 863 (999-136). Do mesmo modo, o numero -499 tem a representação 500 em complemento para 9 e o número -0, tem a representação 999 em complemento para 9.

Quando trabalhamos em binário, os princípios são equivalentes, surgindo esta representação designada por complemento para 1. Se trabalharmos com 8 bits, temos a representação desde 0000000 a 01111111 para os positivos, e as representações 1000000 a 1111111 para os negativos.

Da mesma forma, em complemento para 1, os números positivos têm uma representação idêntica a eles próprios, e os negativos, são representados pelo resultado da subtração do seu valor pelo número binário em que todos os dígitos são 1s.

Exemplo:
Converter -95_{10} em complemento para 1
-95_{10} em binário: -01011111_2
Complemento para 1: 11111111-01011111=10100000

Se repararmos com atenção, notamos que efetuar esta operação equivale a inverter todos os dígitos, um a um, ou seja onde está 0 passa a 1 e onde está 1 passa a 0.

Uma das limitações que se mantêm ao utilizarmos uma representação de complemento para 9 ou de complemento para 1 é a de que o zero continua a ser representado por dois valores, ou seja continuamos a ter os valores +0 e -0 representados.

A representação em complemento para 10 e complemento para 2 resolve este problema. A representação em complemento para 10 e complemento para 2 é obtida pela adição de 1 unidade no caso das representações de números negativos.

Assim, em decimal, o número -108 tem as seguintes representações:

- em complemento para 9: 801 (999-108) .
- em complemento para 10: 802 (1000-108) ou seja, a representação em complemento para 9 mais uma unidade.

Em binário a situação é idêntica, pelo que a representação em complemento para 2 de um número negativo pode ser obtida pela inversão do seu valor (complemento para 1) acrescido de uma unidade.

Exemplo:
Converter -86_{10} em complemento para 2

-86_{10} em binário: -01010110_2
Em complemento para 1: 10101001
Em complemento para 2: 10101010 (10101001+1)

Exercícios:
Converta para complemento para dois os seguintes números decimais usando a seguinte representação: 1 bit para o sinal e 7 bits para a parte inteira: +7, +46, +105, -12, -54, -117;

Exercícios:
Converta para decimal os seguintes números binários, sabendo que são números binários com sinal representados em complemento para dois: 11010010_2 e 11101011_2;

4.0.1 Operações em complemento para 2

Uma das vantagens da utilização da representação em complemento para 2 é que o algoritmo utilizado para a adição pode ser sempre o mesmo, independentemente do sinal das parcelas.

A soma é feita de forma normal, sendo que a ocorrência de um "carry" final é ignorado. No entanto, é importante notar que, quando numa soma em que os bits mais significativos de cada uma das parcelas forem idênticos, se obtiver um resultado onde o bit mais significativo for de valor diferente do das parcelas temos a ocorrência de uma situação de "overflow", ou seja, ultrapassámos a capacidade de representação de valores daquele número de bits...

Exercícios:
Efetue a adição dos seguintes números binários com sinal:

- 00001101 (+13) e 11110111 (-9)
- 11110011 (-13) e 00001001 (+9)
- 11110011 (-13) e 11110111 (-9)

Desta forma descobrimos um novo método para subtrair números binários. Para subtrair X por Y basta-nos somar X com o complemento para 2 de Y.

Exercícios:
Efetue a subtração do seguintes números binários sem sinal usando o novo método aprendido.

- 01011011 (91) e 00101110 (46)
- 01001101 (77) e 01011000 (88)

5 Números em Vírgula Flutuante

Consideremos o número decimal 12345. Sem alterarmos o seu valor podemos representá-lo de diferentes formas:

$$12345 * 10^0$$

$$0.12345 * 10^5$$

$$12345 * 10^4$$

$$0.0012345 * 10^7$$

Estas representações não implicaram perda de precisão.

No entanto, a representação $0,00123 * 10^7$ traduz-se num sacrifício de dois dígitos de precisão, em detrimento de 2 zeros iniciais que não acrescentaram nenhuma informação...)

Esta representação é designada por notação exponencial ou notação científica.

Dela fazem parte:

- O sinal do número (+, no nosso exemplo, embora omitido)
- A grandeza do número, designada por mantissa (12345)
- O sinal do expoente
- A grandeza do expoente
- A base do expoente

5.1 Representações de números em Vírgula Flutuante

Como no caso dos inteiros, os números em vírgula flutuante são armazenados e manipulados de acordo com um " standard", ou seja , de acordo com um sistema pré-definido. Por razões práticas, um múltiplo de 8 bits é normalmente usado como "palavra".

Peguemos no exemplo de um número decimal representado por 7 dígitos decimais e um para sinal: SMMMMMMM. Este formato permite representar qualquer inteiro na seguinte gama:

$$-9999999 < numero < +9999999$$

No caso da vírgula flutuante, uma representação pode ser: SEEMMMMM . Sendo S o sinal da mantissa, EE os 2 dígitos para o expoente e MMMMM o valor da mantissa.

Neste formato, ao utilizarmos dois dígitos para o expoente ficamos com falta de um digito para representar o sinal do expoente, a menos que abdicássemos de um deles.

Uma forma comum de ultrapassar esta questão consiste em utilizar uma representação que surge designada como excesso para 50. Nesta representação, 50 corresponde ao valor 0 de expoente, para baixo de 50 estão os negativos e para cima de 50 os positivos. Assim 46 representa -4 como valor de expoente e 53 representa 3 como valor do expoente.

A escolha do 50 é uma opção da especificação do formato e surge naturalmente porque surge a meio dos valores que podem ser representados por 2 dígitos: 0 a 99.

Com a notação SEEMMMMM e utilizando excesso para 50, como representação do expoente, a gama de valores representados será:

$$-0.99999x10^{-50} < numero < +0.99999x10^{49}$$

5.1.1 Normalização

A normalização tem por objetivo maximizar a precisão possível, procurando-se por isso armazenar os números com o menor número possível de zeros "à esquerda".

Pegando no nosso exemplo de representação, SEEMMMMM, a normalização pode fazer-se através dos seguintes passos:

1. Se o número não tem um expoente, dotar o número de um expoente zero (i.e. X 10^0)
2. Deslocar a parte decimal para a esquerda ou direita, aumentando ou diminuindo o expoente, por forma a colocar o ponto decimal na sua posição correta
3. Corrigir a precisão, através da eliminação de dígitos ou pela adição de zeros no fim
4. Mudar a notação do expoente para excesso para 50

Exemplo:
Normalizar 246.8035
Passo 1: 246.8035 x 10^0 (Acrescentou-se x 10^0)
Passo 2: 0.2468035 x 10^3 (Deslocou-se 3 casas e aumentou-se o expoente)
Passo 3: 0.24680 x 10^3 (Acertou-se a precisão a 5 dígitos)
Passo 4: O expoente 3 em notação excesso 50 representa-se por 53
Resultado da normalização: 05324680

Exercícios:

- Sabendo que os seguintes números estão representados na notação SEEMMMMM, onde S corresponde ao sinal (0 para positivo e 5 para negativo) e estando o expoente representado em excesso para 50, diga quais são os valores a que correspondem os seguintes números: 05324657, 54810000 e 04925000.
- Usando o mesmo formato represente os seguintes números decimais: 19557, -63,24 e -0.0234567.

5.2 Representação de números em virgula flutuante no computador

As técnicas discutidas anteriormente podem ser convertidas para armazenamento no computador simplesmente pela substituição da representação decimal de cada um dos campos por uma representação em binário.

Números em virgula flutuante podem ser representados de diferentes maneiras, existindo diferentes formatos.

Um dos formatos mais comuns está associado à norma IEEE Standard 754. IEEE (normalmente, lê-se I 3 És) está associado à *International Electric and Electronic Engineering Society*, que é uma associação profissional bastante ativa e conhecida pelas suas atividades de normalização.

A IEEE desenvolveu uma norma para representação em virgula flutuante para 32 e 64 bits.

A primeira representação, designada de precisão simples, divide os 32 bits da seguinte forma: 1 para sinal, 8 bits para expoente e 23 para mantissa. A normalização IEEE 754 para 32 bits (precisão simples) apresenta as seguintes especificidades:

- O expoente é representado em excesso de 127
- Utilizando os 23 bits de mantissa posso representar 24 bits: como os números normalizados começam sempre por 1 esse bit não é representado mas está implícito !!!

Exemplo:
Representar 25375 em IEEE 754 precisão simples
Passo 1: 110001100011111 (Conversão para binário)
Passo 2: 1.10001100011111 x 2^{14} (Desloco até obter '1.')
Passo 3: 14 equivale a 141 (excesso de 127 em base 10) que equivale a 10001101 em binário
Passo 4: Ignoro o 1 à esquerda do ponto, porque é sempre 1
Resultado: 0 10001101 10001100011111000000000

Exercícios:
Represente os seguintes números usando o standard 754 da IEEE (32 bits): 123 e -6799