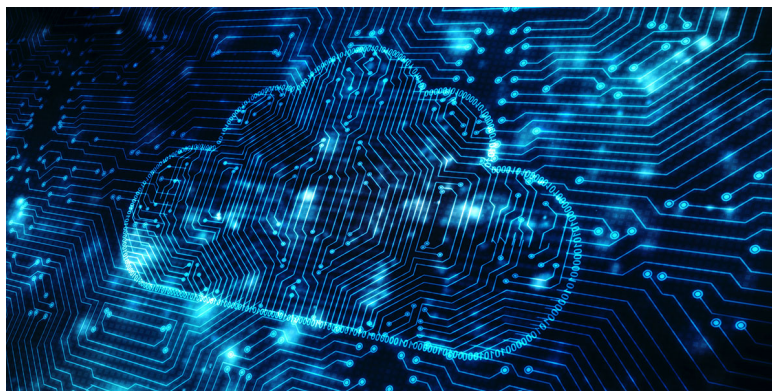


Organização de Computadores

Aula 4 - Aritmética e Representação de Sinais

INTRODUÇÃO



Já se sabe que os dispositivos de computação fazem todo seu processamento em BITS, ou seja, no sistema binário. Obviamente, toda operação aritmética ou lógica (isso será visto com mais detalhes nas próximas aulas) é feita sem efetuar nenhuma conversão, e sim em binário, para, dessa forma, ser convertido e visualizado pelo ser humano na base pretendida, seja ela decimal, hexadecimal, entre outras.

Nesta aula, serão apresentados os processos de cálculo das operações aritméticas nesses sistemas de base e as formas de representação de um número positivo ou negativo no sistema binário.

OBJETIVOS



Desenvolver cálculos aritméticos com números binários.

Desenvolver cálculos aritméticos com números hexadecimais.

Reconhecer os métodos de sinalização de números positivos e negativos em sistema binário.

INTRODUÇÃO

Toda operação aritmética ou lógica com *mais* é feita sem efetuar nenhuma conversão, mas, sim, em binário, para, dessa forma, ser convertido e visualizado pelo ser humano na base pretendida, seja ela decimal, hexadecimal ou outras.

Nesta aula, serão apresentados os processos de cálculo das operações aritméticas nestes sistemas de base. Também é muito importante entendermos como se faz a representação de números positivos e negativos na arquitetura de computadores.

Sendo assim, é fundamental que conheçamos as técnicas de: **Sinal de Magnitude** e **Complemento de 1 e de 2** para tais representações.

OPERAÇÕES ARITMÉTICAS

As operações aritméticas em outros sistemas de base devem ser feitas da mesma forma que na base decimal, com uma particularidade importante: **a quantidade de algarismos disponíveis no sistema de base**. Isso trará resultados diferentes nas somas, de acordo com a base em questão.

SOMA NO SISTEMA BINÁRIO

Para esse caso, devemos lembrar que o sistema possui somente 2 algarismos: 0 e 1.

Sendo assim, diferente da soma decimal, onde:

$$1_{10} + 1_{10} = 2_{10}$$

A soma em binário nos traz que:

$$1_2 + 1_2 = 10_2$$

Isso nos traz uma regra já conhecida no sistema decimal: **o transporte para outra coluna, “vai um”**.

Sendo assim, temos:

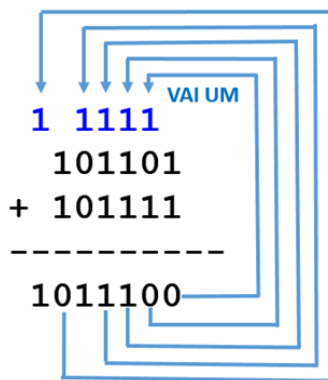
$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0, \text{ com “vai 1”, ou seja, } 10.$$

Veja o exemplo da soma dos binários 101101 + 101111:



Fonte:

Tomando como base o exemplo anterior, cada vez que ocorre a soma de 1+1, deve-se lembrar que “vai um” para o vizinho logo à esquerda, como é feito na soma de decimais.

SUBTRAÇÃO NO SISTEMA BINÁRIO

Para este caso, devemos também lembrar que o sistema possui somente 2 algarismos: 0 e 1.

Apesar de usar o mesmo método de subtração da base decimal, a subtração em binário pode ser um pouco mais complexa, no caso de se ter a ocorrência 0 – 1, onde deve ser feito um empréstimo superior do primeiro algarismo diferente de zero, existente à esquerda.

Isso traz a seguinte regra:

$$0 - 0 = 0$$

$$0 - 1 = 11 \text{ (“1 e empresta 1”)}$$

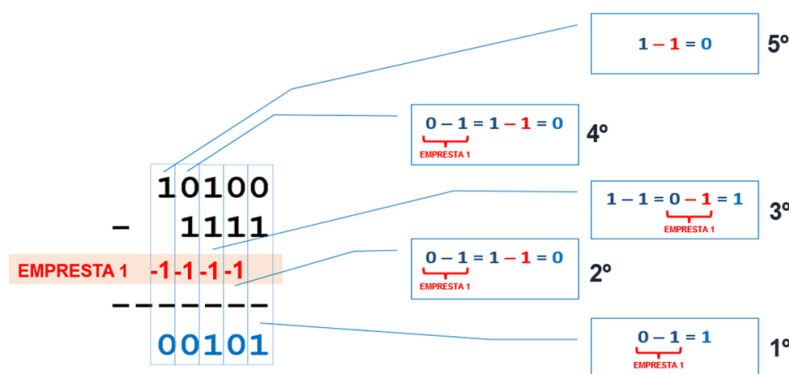
$$1 - 0 = 1$$

$$1 - 1 = 0$$

Veja, no exemplo da subtração dos binários 10100 – 1111, o passo a passo de como se fazer a subtração.

Atenção

, O empréstimo pode ser feito em qualquer local da operação. Porém, recomenda-se que seja feito no meio ou no final. Caso você coloque acima, pode se confundir com o “vai um” utilizado na soma de binários.



Perceba a importância de executar o passo a passo, verificando quando é necessário “emprestar” do vizinho a fim de completar a subtração necessária.

Essa atenção é necessária para que o cálculo seja feito com exatidão. Por isso, para não confundir, é recomendável inserir a representação do empréstimo com o sinal negativo e abaixo dos cálculos, conforme diagrama acima.

ATIVIDADE

Nada melhor do que praticar as operações aritméticas para verificar se os conceitos e procedimentos foram assimilados!

Dessa forma, tente efetuar os cálculos abaixo, conforme regras estudadas:

- 1) $101010_2 + 11011_2$
- 2) $110011_2 + 101101_2$
- 3) $100011_2 - 1111_2$
- 4) $1000_2 - 111_2$

Resposta Correta

SOMA NO SISTEMA HEXADECIMAL

Para realizar operações aritméticas no sistema hexadecimal, é importante relembrar, assim como na base binária, que existe um número diferenciado de algarismos que representam esta base.

Dessa forma, antes de abordar as regras para cálculos com números hexadecimais, é importante relembrar a **equivalência dos números decimais e seus respectivos hexadecimais**:

Decimais	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----------------	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

Hexadecimais	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
---------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

No caso da soma de hexadecimais, todas as regras continuam as mesmas, com um detalhe a ser observado: o **“vai 1”** será **utilizado sempre que a soma de dois algarismos exceder o valor de 15** (ou seja, F), que é o maior algarismo da base hexadecimal, e irá com o valor 1 para o vizinho, e não como 15 ou F.

Veja o exemplo a seguir, relativo à soma $12ABCD_{16} + 4B1F91_{16}$:

5	D	C	B	5
4	B	1	F	9
1	2	A	1	C
4	5	1	8 e ⁿ val 1"	D
		2+11=13=D	11+15+1=27 / 27-16=11=8 e ⁿ val 1"	E
		10+1+1=12=C	13+9+2=21 21-16=5=e ⁿ val 1"	
			13+1=14=E	

Repare que, ao efetuar a soma, trocamos a letra pelo valor decimal correspondente. Caso seja superior, subtraímos o valor 16, a fim de verificar o valor que ficará nessa unidade, e o restante será transportado como “1” para o vizinho da esquerda.

Logo, o procedimento de soma em hexadecimal será concluído com sucesso.

SUBTRAÇÃO NO SISTEMA HEXADECIMAL

Para realizar subtrações no sistema hexadecimal, é importante relembrar as regras do sistema da base hexadecimal, que também possui caracteres que representam alguns de seus algarismos.

A regra se assemelha à soma no sistema hexadecimal. Porém, ao contrário do transporte, conhecido como “vai um”, temos o empréstimo do vizinho. Assim como já é feito no sistema decimal. Porém, ao emprestar 1 unidade do vizinho, estará emprestando o equivalente a 16.

Vamos ao exemplo abaixo, da subtração dos hexadecimais **D8A93C - 23E4A1**:

B	2	D	$13 - 2 = 11 = B$
4	3	8	$7 - 3 = 4$
C	E	9	$10 + 16 (\text{1 emprastado}) - 14 = 12 = C$
4	4	8	$8 - 4 = 4$
9	A	3	$3 + 16 (\text{1 emprastado}) - 10 = 9$
B	1	C	$12 - 1 = 11 = B$

ATIVIDADE

Nada melhor do que praticar as operações aritméticas para verificar se os conceitos e procedimentos foram assimilados!

Efetue os cálculos abaixo, conforme as regras estudadas:

- 1) $AA_{16} + BB_{16}$
- 2) $ABC_{16} + 2DE_{16}$
- 3) $BB_{16} - AA_{16}$
- 4) $CAFE_{16} - ABCD_{16}$

Resposta Correta

REPRESENTAÇÃO DE NÚMEROS POSITIVOS E NEGATIVOS EM BINÁRIO

É possível que você esteja se perguntando...Se os sistemas de computação processam tudo em binário, como podem fazer com números negativos?

Pois bem, é isto que será visto agora, ou seja, o uso de sinal + (positivo) ou – (negativo) em números binários.

Existem várias maneiras. Porém, falaremos dos três principais tipos.

SINAL E MAGNITUDE

Nesta representação, caso tenhamos um número com n algarismos binários (n bits), seu sinal é obtido inserindo-se um bit adicional mais à esquerda, para indicar o valor do sinal, e sua magnitude, ou seja, o seu valor, continua mantendo os bits restantes deste número.

Os valores dos bits de magnitude (ou seja, o valor binário do número) sempre permanecem os mesmos, sendo que a única alteração é o bit mais à esquerda, onde 0 indica POSITIVO e 1 indica NEGATIVO.

Veja um exemplo, da representação binária em 7 bits do número decimal 39:

Número em binário:	0100111 (7 Bits)	39
Sinal + (Bit adicional)	00100111 (8 Bits)	+39
Sinal - (Bit adicional)	10100111 (8 Bits)	-39

Essa solução implementada gerou alguns problemas, pois iniciou uma dupla interpretação para o zero, que poderia ser um sinal positivo, ou somente uma indicação de um número, gerando ambiguidades de significado.

COMPLEMENTO DE 1

Mais um método de representação de números positivos e negativos em binário. Um pouco mais complexo, porém mais confiável do que o método de sinal e magnitude.

Para executá-lo, devem ser feitos os seguintes procedimentos:

1. Com o número binário a ser inserido o sinal, acrescenta-se um bit significativo 0 (zero) mais à esquerda do número, identificando o sinal POSITIVO do mesmo. Procedimento já executado anteriormente em sinal e magnitude;


2. Para identificação do número como negativo, é feita também a inclusão de um bit significativo à esquerda, porém agora invertido, ou seja, ao invés de 0 (zero), 1 (um), assim como a regra de sinal e magnitude. Porém, *também são invertidos todos os Bits relativos à magnitude (ou identificação do valor) do número.*

Veja um exemplo da mesma representação binária em 7 bits do número decimal 39 no Complemento de 1:

Número em binário:	0100111 (7 Bits)	39
Sinal + (Bit adicional)	00100111 (8 Bits)	+39
Sinal - (Bit adicional)	11011000 (8 Bits)	-39

Explicando a tabela, temos:

Número em binário:	0100111 (7 Bits)	39
Sinal + (Bit adicional)	00100111 (8 Bits)	+39
Sinal - (Bit adicional)	11011000 (8 Bits)	-39



Repare que todos os binários que representam o algarismo foram invertidos

COMPLEMENTO DE 2

Método muito utilizado em computação, por não dar ambiguidade no valor e significado do zero em seu significado. **Este método é tido como o mais confiável da atualidade.**

Para entendê-lo, devemos rever o processo do Complemento de 1, com um procedimento adicional:

1. Com o número binário a ser inserido o sinal, acrescenta-se um bit significativo 0 (zero) mais à esquerda do número, identificando o sinal POSITIVO do mesmo.

Procedimento já executado anteriormente em sinal e magnitude;

2. Para identificação do número como negativo, é feita também a inclusão de um bit significativo à esquerda. Porém, agora invertido, ou seja, ao invés de 0 (zero), 1 (um), assim como a regra de sinal e magnitude. Porém, *também são invertidos todos os Bits relativos à magnitude (ou, identificação do valor) do número*;

3. Realizada a inversão dos Bits, é adicionado 1 ao binário resultante;

4. Feito isso, é encontrado o Complemento de 2, relativo ao negativo de um número em binário.

Veja um exemplo, da mesma representação binária em 7 bits do número decimal 39 no Complemento de 2:

Número em binário:	0100111 (7 Bits)	39
Sinal + (Bit adicional)	00100111 (8 Bits)	+39
Sinal - (Bit adicional)	10100111 (8 Bits) 11011000 + 1 11011001	-39

Explicando a
tabela, temos:

Número em binário:	0100111 (7 Bits)	39
Sinal + (Bit adicional)	00100111 (8 Bits)	+39
Sinal - (Bit adicional)	11011000 (8 Bits) 11011000 + 1 11011001	-39

Veja que todos os binários que representam o algarismo foram invertidos.
Para finalizar o Complemento de 2, é somado 1 ao resultado obtido.

ATIVIDADE

1 - Vimos na atividade da aula passada a calculadora do sistema operacional e sua funcionalidade para conversão de sistemas de bases diferentes. Agora vamos fazer um novo teste.

Será que esta mesma calculadora também faz cálculos aritméticos com números de outras bases?

Encontre os resultados dos cálculos abaixo:

$$AAA_{16} + BBB_{16} = ?$$

$$101101_2 + 101111_2 = ?$$

$$10100_2 - 1111_2 = ?$$

Lembre-se: Abra a Calculadora e selecione o Menu. Depois, clique em Programador. A calculadora será alterada, podendo fazer conversões entre as bases.

Clique na base que você deseja fazer os cálculos. Ela alterará a cor e você poderá efetuar os cálculos naquela base específica.

Resposta Correta

2 - Efetue a operação aritmética $10101_2 - 1111_2$:

- ☐ 110
- ☐ 111
- ☐ 011
- ☐ 1000
- ☐ 1001

Justificativa

3 - Efetue operação aritmética $111000111_2 + 11001_2$:

- ☐ 1000000001
- ☐ 111100000
- ☐ 10000
- ☐ 1110011111
- ☐ 1111111111

Justificativa

4 - Efetue operação aritmética $FACE_{16} - BA1A_{16}$:

- ☐ CAC4
- ☐ FAB4

☐ 40B4

☐ 44BB

☐ FBFA

Justificativa

Glossário