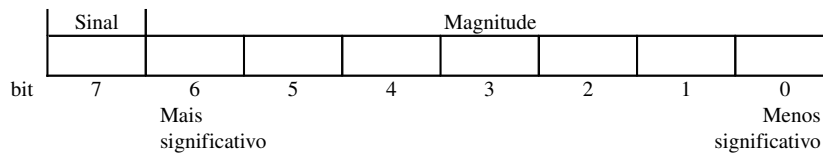


Apoio 1

Complemento de 1, Complemento de 2 e Aritmética Binária

Dado um número binário A, composto de n bits, tem-se que 1 bit é utilizado para representar o sinal, negativo ou positivo e n-1 bits são utilizados para representar o valor numérico (magnitude) associada ao número em questão. Desta forma, a utilização de n bits permite que 2^n valores diferentes sejam representados, no entanto, além do número 0, uma faixa de valores será considerada positiva, outra faixa será considerada negativa. Normalmente o bit mais significativo representa o sinal, sendo 1 utilizado para representar o sinal negativo. Nesta representação, o bit menos significativo corresponde ao bit 0, apresentado normalmente como o mais a direita, e o bit mais significativo o bit n-1, apresentado como o bit imediatamente a direita do bit de sinal.



Representação de um número binário com 8 bits.

Complemento de 1

A representação em complemento de 1 de um determinado número binário é obtida pela **negação** de todos seus bits. Ou seja, todo bit 0 passa a ser bit 1 e todo bit originalmente 1 passa a ser 0. Todos os n bits do número devem ser negados, mesmo o bit de sinal. Desta forma, ressalta-se que o número 0 tem duas representações: 0000 0000 (+0) e 1111 1111 (-0). A Tabela 1 apresenta os valores do complemento de 1 para números representados com 4 bits de precisão.

Tabela 1. Complemento de 1 com valores inteiros de 4 bits

| Decimal | Complemento de 1 |
|---------|------------------|
| 7 | 0111 |
| 6 | 0110 |
| 5 | 0101 |
| 4 | 0100 |
| 3 | 0011 |
| 2 | 0010 |
| 1 | 0001 |
| 0 | 0000 |
| -1 | 1110 |
| -2 | 1101 |
| -3 | 1100 |
| -4 | 1011 |
| -5 | 1010 |
| -6 | 1001 |
| -7 | 1000 |
| -0 | 1111 |

Complemento de 2

A representação em complemento de 2 tem o atrativo de propor uma única representação para o número 0. Por exemplo, considerando 4 bits, o valor 0 é dado por 0000. O valor 1111 corresponde a -128. Uma maneira de obter o complemento de 2 de um número é tomar seu complemento de 1 e somar 1.

As tabelas 2 e 3 abaixo apresentam exemplos de valores em complemento de 2. A Tabela 2 apresenta números com precisão dada por n = 4. Neste caso podem ser representados $2^4 = 16$ valores, além do número 0, os valores positivos

entre 1 e 7 e os valores negativos entre -1 e -8. Observe nesta representação que o valor 0 é assumido *positivo*. Exemplos de representação com 8 bits são dados abaixo.

Tabela 2. Complemento de 2 com valores inteiros de 4 bits

| Decimal | Complemento de 2 |
|---------|------------------|
| 7 | 0111 |
| 6 | 0110 |
| 5 | 0101 |
| 4 | 0100 |
| 3 | 0011 |
| 2 | 0010 |
| 1 | 0001 |
| 0 | 0000 |
| -1 | 1111 |
| -2 | 1110 |
| -3 | 1101 |
| -4 | 1100 |
| -5 | 1011 |
| -6 | 1010 |
| -7 | 1001 |
| -8 | 1000 |

Tabela 3. Alguns valores em complemento de 2 para inteiros de 8 bits

| Decimal | Complemento de 2 |
|---------|------------------|
| 127 | 0111 1111 |
| 64 | 0100 0000 |
| 1 | 0000 0001 |
| 0 | 0000 0000 |
| -1 | 1111 1111 |
| -64 | 1100 0000 |
| -127 | 1000 0001 |
| -128 | 1000 0000 |

Uma vantagem do uso do complemento de 2 é permitir a construção de circuitos que não necessitem conferir os sinais dos números operados para realizar as operações de soma e de subtração. Segue os algoritmos clássicos para realizar as operações aritméticas com números binários. Nos exemplos apresentados os números possuem 6 bits, é possível representar 2^6 valores, ou seja, 64 números, de -32 a +31. Resultados obtidos fora desta faixa não podem ser representados com 6 bits.

Algoritmo 1: Aritmética em complemento de UM

Descrição: Soma dois números representados em binário com n bits em complemento de 2: 1 bit de sinal e n-1 bits de valor

Entrada: A, B;

Saída: R

- $R = A + B$, considerando soma bit a bit, inclusive bit de sinal
- Resultado correto:
 - o (A) Se não ocorreu nenhum caso de "vai-um" para fora do número
 - o Senão:
 - (B) Se ocorreu "vai-um" para o bit de sinal (B)
 - O resultado está incorreto (overflow), pois excede a representação do número
 - Se ocorreu "vai-um" para fora dos n bits de representação
 - Soma-se 1 ao valor final e despreza-se o "vai-um" gerado
 - o Neste caso, se o número de "vai-um" ocorrido para o bit de sinal, para fora do número for:
 - (C) par (ou seja, trocou de sinal duas vezes) o resultado esta correto.
 - (D) ímpar: o resultado esta incorreto - ocorreu overflow

Exemplos:

| 15 + 10 = 25 | 15 + 22 = 37 | -15 -10 = -25 | -15 -22 = -37 |
|---|---|--|--|
| <pre> _111 001111 (+) + 001010 (+) 011001 (+) </pre> | <pre> 1111 001111 (+) + 010110 (+) 100101 (-) </pre> | <pre> 1 1 110000 (-) + 110101 (-) 100101 + 1 100110 (-) </pre> | <pre> 1 110000 (-) + 101001 (-) 011001 + 1 011010 (+) </pre> |
| <p>Caso (A) CORRETO Não ocorreu "vai-um".</p> | <p>Caso (B) OVERFLOW Só ocorreu "vai-um" para o bit de sinal. A soma de dois números positivos não pode gerar resultado negativo.</p> | <p>Caso (C) CORRETO Ocorreu "vai-um" p/ bit de sinal e p/ fora do número, mas não na soma final. O número de "vai-um" é par.</p> | <p>Caso (D): OVERFLOW Ocorreu "vai-um" só p/ fora do número, mas não na soma final. O número de "vai-um" é ímpar. Somar dois números negativos não pode gerar número positivo.</p> |



Algoritmo 2: Aritmética em complemento de DOIS

Descrição: Soma dois números representados em binário com n bits em complemento de 2: 1 bit de sinal e n-1 bits de valor

Entrada: A, B;

Saída: R

- $R = A + B$, considerando soma bit a bit, inclusive bit de sinal
- Resultado correto:
 - o (A) Se ocorreu "vai-um" para o bit de sinal e também para fora do número
 - o (B) Se não ocorreu "vai-um" nem para o bit de sinal nem para fora do número
- Resultado incorreto:
 - o (C) Se ocorreu "vai-um" para o bit de sinal, mas não para fora do número
 - o (D) Se não ocorreu "vai-um" para o bit de sinal, mas ocorreu para fora do número

Exemplos:

| | | | | |
|--|--|--|---|---|
| 15+10 = 25 | 15 + 17 = 32 | -15 -10 = -25 | -15 -17 = -32 | -15 -27 = -42 |
| <pre> _111 001111 (+) + 001010 (+) ----- 011001 (+) </pre> | <pre> 11111 001111 (+) + 010001 (+) ----- 100000 (-) </pre> | <pre> 1 110001 (-) + 110110 (-) ----- 100111 (-) </pre> | <pre> 1 11111 110001 (-) + 101111 (-) ----- 100000 (-) </pre> | <pre> 1 1 110001 (-) + 100101 (-) ----- 010110 (+) </pre> |
| <p>Caso (B) Correto Não ocorreu "vai-um".</p> | <p>Caso (C) Overflow Ocorre "vai-um" para o bit de sinal. A soma de dois positivos não pode gerar negativo.</p> | <p>Caso (A) Correto Ocorreu "vai-um" para o bit de sinal e para o número.</p> | | <p>Caso (D) Overflow Ocorre "vai-um" apenas para fora do número.</p> |

Exercícios:

1. Realize as seguintes operações:

- a) $15 + (-5) =$ b) $15 - (-5) =$ c) $5 * (-6) =$

2. Complete a representação binária da tabela abaixo. Algumas representações não podem ser dadas.

| Decimal | Sem sinal | Complemento de 1 | Complemento de 2 |
|---------|-----------|------------------|------------------|
| +8 | | | |
| +7 | | | |
| +6 | | | |
| +5 | | | |
| +4 | | | |
| +3 | | | |
| +2 | | | |
| +1 | | | |
| (+) 0 | | | |
| (-) 0 | | | |
| -1 | | | |
| -2 | | | |
| -3 | | | |
| -4 | | | |
| -5 | | | |
| -6 | | | |
| -7 | | | |
| -8 | | | |



Respostas:

a)

```

11111 111 (carry)
  0000 1111 (15)
+ 1111 1011 (-5)
=====
0000 1010 (10)
  
```

b)

```

11110 000 (borrow)
  0000 1111 (15)
- 1111 1011 (-5)
=====
0001 0100 (20)
  
```

c)

```

00000101 (5)
x 11111010 (-6)
=====
      0
     101
      0
    101
   101
  101
 x01
xx1
=====
xx11100010 (-30)
  
```

2. Complete a representação binária da tabela abaixo. Algumas representações não podem ser dadas. .

| Decimal | Sem sinal | Complemento de 1 | Complemento de 2 |
|---------|-----------|------------------|------------------|
| +8 | 1000 | N/A | N/A |
| +7 | 0111 | 0111 | 0111 |
| +6 | 0110 | 0110 | 0110 |
| +5 | 0101 | 0101 | 0101 |
| +4 | 0100 | 0100 | 0100 |
| +3 | 0011 | 0011 | 0011 |
| +2 | 0010 | 0010 | 0010 |
| +1 | 0001 | 0001 | 0001 |
| (+)0 | 0000 | 0000 | 0000 |
| (-)0 | N/A | 1111 | N/A |
| -1 | N/A | 1110 | 1111 |
| -2 | N/A | 1101 | 1110 |
| -3 | N/A | 1100 | 1101 |
| -4 | N/A | 1011 | 1100 |
| -5 | N/A | 1010 | 1011 |
| -6 | N/A | 1001 | 1010 |
| -7 | N/A | 1000 | 1001 |
| -8 | N/A | N/A | 1000 |