Números com sinal

Com n dígitos binários, podemos representar 2^n valores diferentes de inteiros sem sinal.
Exemplo: Utilizando-se 3 dígitos:

```
000     0
001     1
010     2
011     3
100     4
101     5
110     6
111     7
```

Representações para números com sinal:

- sinal e magnitude
- complemento a um
- complemento a dois

Sinal e magnitude

O bit mais à esquerda é o sinal e os restantes fornecem a magnitude do número.

| Sinal | Magnitude |
---
0 para números positivos e 1 para números negativos
Exemplo:
00000101 = +5
10000101 = -5
Faixa de valores com n dígitos: $-(2^{n-1}-1)$ a $+(2^{n-1}-1)$.
Para n=4 teremos a faixa -7 a +7
Porque não é utilizado pelos processadores ?

1. duas representações para o zero, por exemplo, 0000 e 1000.
2. processo de adição e subtração complicado

Algoritmo de soma:

1. Verificam-se os sinais dos números e efetua-se uma comparação entre eles
2. Se ambos possuem o mesmo sinal, somam-se as magnitudes e o sinal do resultado é o mesmo das parcelas
3. Se os números possuem sinais diferentes:
    - identifica-se a maior das magnitudes e registra-se o seu sinal
    - subtrai-se a magnitude menor da maior
    - o sinal do resultado é igual ao sinal de maior magnitude

Exemplo utilizando-se 6 bits na base 2:

```
  10          001010
+ 15          001111
-----        --------
+ 25          011001


- 18          110010
-  4          100100
```

```
-----       --------
- 22         110110


+ 15         001111
-  4         100100
-----       --------
+ 11         001011


- 18         110010
+ 10         001010
-----       --------
-  8         101000
```

Algoritmo de subtração:

1. Troca-se o sinal do subtraendo
2. Executa algoritmo de soma

Exemplo: -18 - (+12)= -18 + (-12)

```
- 18         110010
- 12         101100
-----       --------
- 30         111110
```

Custo: 1 elemento para soma e outro para subtração
Velocidade: manipulação de sinal

Representação complemento a 2

Como representar números negativos no sistema decimal com 3 algarismos ?
Divide o sistema em 2, utiliza uma metade para positivos e a outra para negativos.

```
|--------------|------------|-------------|-------|--|
000            250          500           750     998 000
```

Complemento a 10

Positivos são so próprios números: 0 a +499, representação: 0 - 499
Negativos são o complemento a 10 do número positivo
Complemento a base= B^n-N onde n é o número de algarismos que temos para representar um número.
-500=10^3-500=1000-500=500
-499 = 1000 -499= 501
-1 = 1000 -1 = 999
Qual a vantagem ? Só precisamos somar números e resultado é sempre representado corretamente.
Exemplo:
-2 +3, Complemento a 10 de -2 = 1000 - 2 =998 e de +3 = 003, somando 998+003=1001, descartando o 1 à esquerda, temos o resultado 001 que é +1 em complemento a 10.
Para subtrair:
A - B = A + (-B)
-3 -5 = -3 + (-5)
-3 = 997 + 995 = 1992, retirando o 1 à esquerda temos o resultado 992 que é negativo e para sabermos

o valor fazemos 1000 - x = 992, x = 8, logo resultado = -8.

Complemento a 2

Se número positivo e existem n dígitos para representar o número, $d_{n-1}=0$.

Se o número é negativo a representação é 2^n-N onde N é a representação positiva.

```
|---------------|------------|-------------|-------|--|
000             010          100            110    111 000
```

Número 0 só tem 1 representação: 000 positivo e 2^3 - 000 = 1000, representação negativa= 000
Exemplo:
Com 3 algarismos:
-4 = 1000 - 100 = 100
-3 = 101
-2 = 110
-1 = 1000 - 001 = 111
0 = 000
+1 = 001
+2 = 010
+3 = 011

Representação de números positivos: 1 a (2^{n-1}-1)

Representação de números negativos: -1 a (-2^{n-1})

Faixa de valores:$-2^{n-1 \text{ a } (2n-1}$-1)

Como representar um número em complemento a 2 ?

Se o número é positivo, o bit $d_{n-1}=0$ e a magnitude do número é

$d_{n-2}x\ 2^{n-2}+d_{n-3}x\ 2^{n-3}+...+d_1x\ 2^1+d_0 2^0$

Exemplo: +7 utilizando-se 4 algarismos é igual a 0111.
Representação de +27 com 8 bits = 00011011
Se o número é negativo:

$C_2(-N)=2^n-N$, onde N é a representação sem sinal da magnitude do numero negativo.

$C_2(-N)=2^n-N=((2^n-1)-N)+1$

Mas 2^n-1=011111..11 e 011111..11-N é igual a inverter os bits de N. Logo para achar a representação complemento a 2 de números negativos, inverte-se o número na representação dele positiva e soma-se 1.
Exemplo: -7 utilizando-se 4 algarismos 0111, inverte 1000, soma 1 = 1001
Representação de -27 com 8 bits = inv(00011011) + 1 = 11100100 + 1 = 11100101

Dado um número representado em complemento a 2, o que ele está representando ?

O bit mais significativo é o bit de sinal. Se ele é zero o número é positivo dado por :
$N=(d_{n-1}\ d_{n-2}\ ...\ d_1 d_0)$

$N=d_{n-2}x\ 2^{n-2}+d_{n-3}x\ 2^{n-3}+...+d_1x\ 2^1+d_02^0$
Exemplo:
n= 4, N=0111, $N=7_{10}$, N=0100, $N=4_{10}$
Se ele é 1 o número que ele representa é negativo. Como achar que número ele representa ?
$C_2(C_2(-N))=2^n-(2^n-N)=N$
Logo para descobrir a magnitude de um número negativo representado em complemento a 2, inverte-se o número na representação dele sem sinal e soma-se 1.
Exemplo:
O que representa 00011011 ? +27
O que representa 11100101 ? - inv(11100101)+1= -00011011 = -27

Outra maneira de ver:
$2^n+(-N)=d_{n-1}x2^{n-1}+d_{n-2}x2^{n-2}+...+d_12^1+d_02^0$
Mas $d_{n-1}=1$, logo $(-N)=-2^n+2^{n-1}+d_{n-2}x2^{n-2}+...+d_12^1+d_02^0=2^{n-1}(-2^1+1)+d_{n-2}x2^{n-2}+...+d_12^1+d_02^0=2^{n-1}(-1)+d_{n-2}x2^{n-2}+...+d_12^1+d_02^0$
Exemplo: 1001 = -8 +1 = -7

Soma e subtração em Complemento a 2

Soma: N1 + N2
Subtração: N1-N2 = N1+(-N2) e joga fora o vai um
Exemplo: Somar +11 com +21

```
+11          00001011
+21          00010101
----------------------
+32          00100000

+21                            00010101
-11         00001011 11110100+1=11110101
-----------------------------------------
+10                           100001010
                             joga fora o ultimo 1
```

Subtrair +21 de +11

```
+11 - (+21)=+11 + (-21)
+11                                00001011
-21      00010101 = 11101010 +1 = 11101011
-----------------------------------------
-10                                11110110
```

Overflow

Overflow ocorre quando o número não pode ser representado com o número de bits disponível.
Só ocorrerá overflow quando estivermos somando dois números de mesmo sinal.

Se temos 3 bits para representar números em complemento a 2, podemos representar os seguintes inteiros com sinal:

```
011  +3
010  +2
001  +1
000   0
```

```
111  -1
110  -2
101  -3
100  -4
```

Menor número negativo = $-2^{3-1}=-4$

Maior número positivo = $2^{3-1}-1=+3$

Soma: N1+N2

Primeiro caso: N1>0, N2>0:

```
        00
+2       010
+1       001
-----------------
+3       011  OK

        01
+3      0011
+1       001
-----------------
-4       100  Estouro (overflow)
```

Segundo caso: N1<0, N2>0:

```
       11
-1      111
+2      010
-----------------
+1      001  OK

        00
-2       110
+1       001
-----------------
-1       111 OK
```

Terceiro caso: N1<0, N2<0:

```
       11
-1      111
-2      110
-----------------
-3      101  OK

        10
-1       111
-4       100
-----------------
+3       011 Estouro (overflow)
```

$A=(a_{n-1}a_{n-2}...a_0)$

$B=(b_{n-1}b_{n-2}...b_0)$

```
cn  cn-1
    an-1 ..... a0
```

+

b_{n-1} ..... b_0

a_{n-1}	b_{n-1}	c_{n-1}	Overflow ?	c_n # c_{n-1} ?
0	0	0	Não	Não
0	0	1	Sim	Sim
0	1	0	Não	Não
0	1	1	Não	Não
1	0	0	Não	Não
1	0	1	Não	Não
1	1	0	Sim	Sim
1	1	1	Não	Não

Então se c_n # c_{n-1}, temos overflow !

Representação de caracteres

Um caracter deve ser representado utilizando-se um padrão de bits.
Vários dispositivos de entrada e saída trabalham com 8 bits.
Um código padrão denominado ASCII (American Standard for Computer Information Interchange> define um padrão de seqüência diferente para cada caracter.
Exemplo:
0100 0001 é 41_{hex} ou 65_{10} e representa o caracter 'A'

0100 0010 é 42_{hex} ou 66_{10} e representa o caracter 'B'

Uma propriedade deste código é que se padrões de bits são comparados, então 'A' <'B', o que ajuda a ordenar elementos em ordem alfabética.
Note que 'a'(61_{hex}) é diferente de 'A' (41_{hex})

'8' (38_{hex}) é diferente do inteiro 8

representação do inteiro 8 em complemento a 2: 0000 0000 0000 0000 0000 0000 0000 1000
Representação ASCII do caracter '8': 0011 1000
Representação ASCII do caracter '0' é 48 (decimal) ou 30 (hex)
Representação ASCII do caracter '9' é 57 (decimal) ou 39 (hex)
Como traduzir um string de caracteres para um inteiro ?
Exemplo: '354'
Lê '3'
Traduz '3' para 3 (subtrai de 48)
inteiro= 0 *10 + 3 = 3
Lê '5'
Traduz '5' para 5 (subtrai de 48)
inteiro= 3 *10 + 5 = 35
Lê '4'
Traduz '4' para 4 (subtrai de 48)
inteiro= 35 *10 + 4 = 354
Como traduzir um inteiro para string de caracteres ?
Exemplo: 354
Verifica quantos caracteres existem na base desejada (no caso 3)
Inicia com base$^{(\text{no. caracteres - 1})}$, ($10^{3-1}$)=100
354 div 100, dá 3
Traduz 3 para '3' (soma 48) e imprime
354 mod 100, é 54
100/10=10

54 div 10, dá 5
Traduz 5 para '5' (soma 48) e imprime
54 mod 10, é 4
10/10=1
4 div 1, dá 4
Traduz 4 para '4' (soma 48) e imprime
4 mod 1, é 0
1/10=0, então acabou