Introdução a Computação

Prof.

Luís Fernando GARCIA

luis@Garcia.pro.br
www.Garcia.pro.br

Aula 7

Aritmética Computacional

Números Binários

Em Geral

Representação de números

- Números reais: infinitos.
- No computador: finitos.
- Maioria: grande quantidade de zeros à esquerda.
- Computador: pode lidar com números até um certo tamanho.
- *Overflow*: tratado pelo sistema operacional.
- No computador: é preciso representar números com sinal.
  - Solução: usar 1 bit (sinal magnitude).
- Primeira tentativa: o bit mais significativos (MSB) é usado para sinal.
  - Problema: duas representações para o zero
  - Solução mais usada: complemento a 2

Adição

Adição binária

Regra:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10 \quad \text{(0 e vai 1)}$$
$$1 + 1 + 1 = 11 \quad \text{(1 e vai 1)}$$

Exemplos:

```
   10100  (20)
+  01001  (9)
   -----
   11101  (29)
```

```
  1101        11001        111     100111
+    1     +   1011   + 01100   +   1110
  ----     ------     -----     ------
  1110     100100     10011     110101
```

Aritmética Binária

Como a Unidade Lógica e Aritmética do processador realiza operações aritméticas ?

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

Exercício de Adição binária

101	100100	11001	101110
+ 1101	+ 10010	+ 10011	+ 1110

Overflow

Aritmética Computacional

- Overflow
  - Acontece quando o resultado de uma soma não poder ser armazenada no número de bits disponíveis.
  - Somar 10_{10} e 13_{10} (4 bits).
  - Carry Out (Vai Um) = Overflow

Números Binários

Com e Sem SINAL

Números com Sinal e Números sem Sinal

- **Base 10:**
  - $2543_{(10)}=2\times10^3+5\times10^2+4\times10^1+3\times10^0{}_{(10)}$
- **Base 2:**
  - $1011_{(2)}=1\times2^3+0\times2^2+1\times2^1+1\times2^0=11_{(10)}$
- **Representação (8 bits)**

0 0 0 0 1 0 1 1

Bit Mais Significativo (MSB)

Bit Menos Significativo (LSB)

Representação de Números Binários

- Dois mét
  - Sinal
    - Sime
    - Dific méticas
  - Comp
    - Repr os negativos
    - Facil tméticas (ao menos para o hardware)

Representação de Números Binários com Sinal

- Dois métodos principais:
  - **Sinal Magnitude**
    - Simetria de Positivos e Negativos
    - Dificuldade em fazer operações aritméticas
  - **Complemento de 2**
    - Representação não trivial de números negativos
    - Facilidade em fazer operações aritméticas (ao menos para o hardware)

- Técnica **Sinal-Magnitude**
  - Ex: 1 0001010 → valor = 10

    sinal negativo

    -10
  - Sinal: 0 = “+”, 1 = “-”,
- O bit mais significativo, o da esquerda, representa o sinal. Os demais o valor.
- Apesar de ser muito familiar ao ser humano esta notação é pouco utilizada na prática

Sinal-Magnitude

- Ex.:
  0011 = +3
  1011 = -3

Exemplo para 4 bits:

- **0**000 : 0
- **0**001 : 1
- **0**010 : 2
- **0**011 : 3
- **0**100 : 4
- **0**101 : 5
- **0**110 : 6
- **0**111 : 7
- **1**000 : -0 (?)
- **1**001 : -1
- **1**010 : -2
- **1**011 : -3
- **1**100 : -4
- **1**101 : -5
- **1**110 : -6
- **1**111 : -7

- Técnica de **Complemento de 2**

  - Ex: $\boxed{1\,|\,1110110} \rightarrow$ valor = 10 $\Big\}$ -10

    sinal negativo

  - Sinal: 0 = "+", 1 = "-",

- O bit mais significativo, o da esquerda, representa o sinal. Os demais o valor, da seguinte forma:
  - os valores positivos normalmente; os negativos invertemos os bits do valor positivo e somamos 1
- Apesar de parecer complexa, esta notação apresenta propriedades interessantes.

- Ex.:
  0011 = +3
  1101 = -3

Exemplo para 4 bits:

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

Complemento a 2: regra prática

- Como representar 8 e -8 ?
- 8 = 0000 1000_2
- -8 →

$$
\begin{array}{rr}
 & 8 \rightarrow 0000\ 1000 \\
 & -\quad 1111\ 0111 \\
+ & 1 \\
\hline
 & 1111\ 1000
\end{array}
$$

Inverte bits

Soma 1

- -8 = 1111 1000

Represente os valores 5 e -5 (em 5 bits) ?

- 5_{10} em binário é 00101
  - Invertendo bits: 00101 → 11010
  - Somando 1: 11010 + 1 = 11011
- -5_{10} em complemento de 2 é **11011**

Porque Complemento de 2 é útil ?

```
  00000110 (+6)
+ 00001101 (+13)
  ---------------
  00010011 (+19) - ok

  00000110 (+6)
+ 11110011 (-13 em C2)
  ---------------------
  11111001 (-7 em C2) - ok

  11111010 (-6 em C2)
+ 11110011 (-13 em C2)
  ---------------------
  11101101 (-19 em C2) - ok
```

Porque Complemento de 2 é útil ?

```
   00000110 (+6)
+
```

Porque dá direto a resposta !!!

Não precisa ficar convertendo ...

Só faz a soma bit a bit ...

```
+  11110011 (-13 em C2)
   11101101 (-19 em C2) - ok
```

Porque Complemento de 2 é útil ?

```
  00000110 (+6)
+ 00001101 (+13)
  ---------------
  00010011 (+19) - ok

  00000110 (+6)
+ 11110011 (-13 em C2)
  ---------------------
  11111001 (-7 em C2) - ok

  11111010 (-6 em C2)
+ 11110011 (-13 em C2)
  ---------------------
  11101101 (-19 em C2) - ok
```

Exercícios em Complemento de 2

- Represente em binário os números abaixo utilizando 5 bits (1 bit para sinal e 4 bits para o valor)
  - Ex.: 11_{10} = 00011 e -11_{10} = 11101

- 5_{10} = 00101 e -5_{10} = 11011
- 1_{10} = 00001 e -1_{10} = 11111
- 0_{10} = 00000 e -0_{10} = 00000
- 15_{10} = 01111 e -15_{10} = 10001
- 16_{10} = ????? e -16_{10} = 10000

Exercícios

1). Converta os decimais para binário e faça as operações:

a) 35 + 18 b) 64 + 17 c) 155 + 32

3) efetue as operações binárias, considerando um registrador de 8 bits.

a) 0 0 0 1 0 1 0 0 + 0 0 1 0 0 0 1 1

b) 0 0 1 1 0 0 0 0 + 1 1 1 0 1 1 1 0

c) 1 0 1 0 0 0 0 0 + 0 0 1 0 1 0 0 0

4) considerando um registrador de 8 bits e representação em complemento de 2, converta os decimais para binário e faça as operações.

a) 96 + 13 b) 112 – 70 c) -65 + 30 d) -52 – 43

112 + (-70) ...
112 para binário
70 para binário
para complemento de 2 (para ser negativo)
soma