Conceitos Básicos

BITs e BYTEs

- **Bit = BInary digiT = vale sempre 0 ou elemento básico de informação**
- **Byte = 8 bits processados em paralelo (ao mesmo tempo)**
- **Word = n bytes (depende do processador em questão)**
- **Double word = 2 words**
- **Nibble = 4 bits (utilidade para BCD)**

- **Posição de bits:**

Para 1 byte:

7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1

Para 1 word (de 16 bits):

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

byte alto (high byte) | byte baixo (low byte)

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Litle Endian X Big Endian

Words são armazenados em bytes consecutivos, em memórias de largura de 8 bits.

Exemplo:

1025_{10} = 00000000 00000000 00000100 00000001_2

Endereço	Representação Big-Endian (MOTOROLA)	Representação Little-Endian (INTEL)
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Memória

- **Memória: local do computador (hardware) onde se armazenam temporária ou definitivamente dados (números, caracteres e instruções)**
- **Posição de memória ou endereço: localidade física da memória onde se encontra o dado.**
- **Organização da memória:**

Endereço	Conteúdo
...	...
4MB	10110101
...	...
1048576	01001010
...	...
1765	01001101
...	...
4	01010000
3	11111111
2	11101001
1	11011010
0	01100100

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Representação binária de números não sinalizados

Qualquer número em qualquer base ✍ $N = \sum_{i=0}^{n-1} d_i \times base_i$

a) 1 byte

$00100111_2 = 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 0 + 0 + 32 + 0 + 0 + 4 + 2 + 1 = 39_{10}$

$= 27_{16}$

b) 1 word

$0101011101101110_2 = 0 \times 2^{15} + 1 \times 2^{14} + ... + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 22382_{10}$

$= 576E_{16}$ (mais fácil de representar!)

high byte $= 0101\ 0111b = 57_{16}$

low byte $= 0110\ 1110b = 6E_{16}$

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Conversão entre bases numéricas

Tipo de conversão	Procedimento
Decimal ✍ Binário	Divisões sucessivas por 2 até se obter zero no quociente. Leitura dos dígitos binários de baixo para cima.
Binário ✍ Decimal	Soma de potências de 2 cujo expoente é a posição do bit e cujo coeficiente é o próprio bit.
Hexadecimal ✍ Binário	Expandir cada dígito hexa em quatro dígitos binários segundo seu valor.
Binário ✍ Hexadecimal	Compactar cada quatro dígitos binários em um único dígito hexa segundo seu valor.
Decimal ✍ Hexadecimal	Divisões sucessivas por 16 até se obter zero no quociente; leitura dos dígitos de baixo para cima.
Hexadecimal ✍ Decimal	Soma de potências de 16 cujo expoente é a posição do dígito e cujo coeficiente é o valor do próprio dígito hexa.

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Representação binária de números sinalizados

- **Representação com sinal e magnitude**
  - **O bit mais significativo é o sinal do número ✍ se for 1 o número é negativo se for 0 o número é positivo**

Exemplo 1: 01110001_2

valor não sinalizado = $0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 +$
$+ 0 \times 2^1 + 1 \times 2^0 =$
$= 64 + 32 + 16 + 1 = 113_{10}$

valor sinalizado **bit de sinal = 0 => " + " (positivo)**
$= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 +$
$= 0 \times 2^1 + 1 \times 2^0 =$
$= 64 + 32 + 16 + 1 = 113_{10}$ ✍ logo= $+113_{10}$

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplo 2: 10110001_2

valor não sinalizado **$= 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 +$**
$+ 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$
$= 128 + 32 + 16 + 1 = 177_{10}$

valor sinalizado **bit de sinal = 1 => " - " (negativo)**
$= 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 +$
$+ 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
$= 32 + 16 + 1 = 49_{10}$ ✍ logo = $- 49_{10}$

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplo 3:
$70FF_{16}$ = 0111000011111111_2

valor não sinalizado = $0 \times 2^{15} + 1 \times 2^{14} + \ldots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

valor sinalizado ✍ bit de sinal = 0 => " + " (positivo)
= + ($0 \times 2^{15} + 1 \times 2^{14} + \ldots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$)

Exemplo 4:
$C777_{16}$ = 1100011101110111_2

valor não sinalizado = $1 \times 2^{15} + 1 \times 2^{14} + \ldots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

valor sinalizado ✍ bit de sinal = 1 => " - " (negativo)
= - ($1 \times 2^{14} + \ldots + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$)

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

- **Representações possíveis de números sinalizados**

Sinal e Magnitude	Complemento de 1	Complemento de 2
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = -0	100 = -3	100 = -4
101 = -1	101 = -2	101 = -3
110 = -2	110 = -1	110 = -2
111 = -3	111 = -0	111 = -1

- **Representação em Complemento de 2 ✍ utilizada pois temos apenas uma representação para o zero e podemos fazer a soma e subtração com apenas um circuito.**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

- **Números sinalizados de 32 bits, em Complemento de 2:**

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = +1_{10}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = +2_{10}$
...
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = +2{,}147{,}483{,}646_{10}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = +2{,}147{,}483{,}647_{10}$ ← ***maxint***
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2{,}147{,}483{,}648_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -2{,}147{,}483{,}647_{10}$ ← ***minint***
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -2{,}147{,}483{,}646_{10}$
...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = -3_{10}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2_{10}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Representação em Complemento de 2 de um número:

- **Partindo-se da representação do negativo do valor a ser achado, nega-se este número (negar ✍ inverter e somar 1)**

Exemplo 1:

-5 em Complemento de 2 (com 1 bit de sinal de 4 para a magnitude)

Partindo-se da representação do $5_{10} = 00101_2$ ✍ (invertendo os bits) = 11010 ✍ (somando 1) = 11011_2 = - 5 em Complemento de 2

Exemplo 2:

+5 em Complemento de 2 (com 1 bit de sinal de 4 para a magnitude)

Partindo-se da representação do $-5_{10} = 11011_2$ ✍ (invertendo os bits) = 00100_2 ✍ (somando 1) = 00101_2 = +5 em Complemento de 2

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

- **Conversão de números com n bits em números com mais que n bits:**
  - **copiar o bit mais significativo (bit de sinal) nos outros bits (extensão do sinal):**

Exemplo:

0010 ✍ 0000 0010

1010 ✍ 1111 1010

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Operações de soma e adição binárias

- **Como aprenderam no primeiro grau: (vai-um/vem-um)**

```
  0111 (7)      0111 (7)      0110 (6)
+ 0110 (6)    - 0110 (6)    - 0101 (5)
  1101 (13)     0001 (1)      0001 (1)
```

- **Adição e subtração em complemento de 2 é feito como se fosse uma soma:**
  - **subtração usando adição de números negativos**

```
   0111  (=+7)
 + 1010  (=-6)
1|  0001  (=1)
```

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Overflow

- **Overflow (resultado maior (menor) que a palavra do computador pode representar):**

Exemplo:

- **Quando na operação abaixo ocorre e quando não ocorre overflow ???**

```
  0111 (7) ou (+7)
+ 0001 (1) ou (+1)
  1000
```


ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Detecção de Overflow

- **Não existe overflow quando adicionamos um número positivo e um negativo**
- **Não existe overflow quando os sinais dos números são os mesmos na subtração**
- **Ocorre overflow quando os valores afetam o sinal:**
  - **Somando dois números positivos dá um número negativo**
  - **Somando dois números negativos dá um número positivo**
  - **Subtrai um número negativo de um positivo e dá negativo**
  - **Subtrai um número positivo de um negativo e dá positivo**

Exercício

- **Considere as operaçôes A + B e A – B**
  - **Pode ocorrer overflow se B = 0 ?**
  - **Pode ocorrer overflow se A = 0 ?**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Multiplicação Binária

- **Exemplo:**

1010 X 101

```
   1 0 1 0
  X 1 0 1
  _______
   1 0 1 0
 0 0 0 0
1 0 1 0
_____________
1 1 0 0 1 0
```


ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Divisão Binária

- **Exemplo:**

1 1 0 0 1 0 / 101

```
  1 1 0 0 1 0   | 1 0 1
- 1 0 1         |_______
  _____          1 0 1 0
  0 0 1 0 1
-     1 0 1
  _________
      0 0 0 0
```

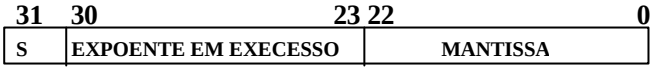
ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Representação de Números em Ponto Flutuante

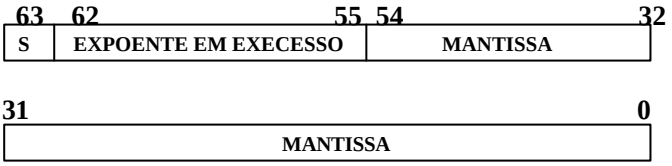
? **Padrão IEEE 754 – normalizado, expoente em excesso 127**

$$N = (-1)^S \times 1.M \times 2^E$$

? **precisão simples**



? **precisão dupla**

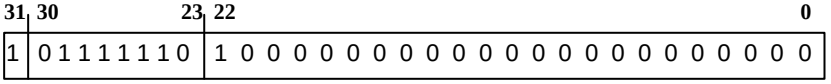


ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

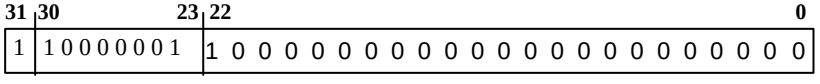
Exemplo

$-0{,}75_{10} = -0{,}11_2$

Normalizando $1{,}1_2 \times 2^{-1}$



Exemplo: Qual o decimal correspondente ?



$$N = -(1+0.25) \times 2^{(129-127)} = -1{,}25 \times 4 = -5{,}0$$

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Representação de Caracteres Alfanuméricos

- **Tabela ASCII (American Standard Code Interchange Information)**

Exemplo:

64	@		96	´
65	**A**		97	**a**
66	**B**		98	**b**
67	**C**		99	**c**
68	**D**		100	**d**
69	**E**		101	**e**
70	**F**		102	**f**
71	**G**		103	**g**
72	**H**		104	**h**
73	**I**		105	**i**

48	**0**
49	**1**
50	**2**
51	**3**
52	**4**
53	**5**
54	**6**
55	**7**
56	**8**
57	**9**