Transformação de Bases Numéricas e Numeros Negativos

Universidade Estadual de Mato Grosso do Sul – UEMS
Ciência da Computação
Linguagem de Montagem
Prf Dr Osvaldo Vargas Jaques
ojacques@comp.uems.br

Bits e Bytes

- Bit – Bi nary digiT, é a menor unidade de informação digital, sabemos que pode ter o valor 0 ou 1, ou polaridade positiva ou negativa.
- Byte – É um conjunto de 8 bits.
  - Curiosidade : Como, pois bite, com “i” significa mordida, diz a lenda que fizeram a seguinte a analogia: se byte (o som é o mesmo) é uma mordida, bit seria mordidinha (little byte). Assim, um conjunto de 8 mordidinhas, seria o equivalente a uma mordida.

Bits e Bytes

- Para que a informação para nós humanos ficasse coerente, precisávamos de uma sequência de bits, que seriam lidos e “pintados” na tela do computador ou impressora.
- Assim foi criado o código ASCII – American Standart Code for Information Interchange, Código Padrão Americano para Intercâmbio de Informações. Cada letra, símbolo, corresponderia a uma sequência de bytes (8 bits).
- Uma maneira do leigo entender que seu arquivo tem 8 kb, ou seja, aproximadamente 8000 bytes, diga que, grosseiramente no seu arquivo caberiam 8 mil letras. Claro que não é bem isso.

Bits e Bytes

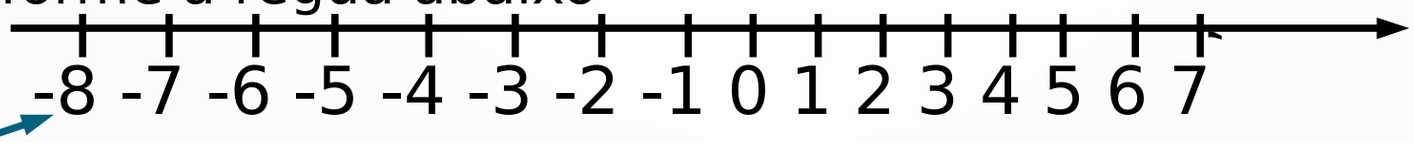
- Verifique na internet o código ASCII https://pt.wikipedia.org/wiki/ASCII
- Exemplo:
  - A letra “A” corresponde a 0100 0001_b=65, “B” a 0100 0010_b=66, “a” a 0110 0001_b = 97, e assim por diante.
- Obviamente, para representações mais complexas de arquivos de texto como Word, Write, Excel, Calc, tornou-se necessário aumentar essa sequência de bits, surgindo assim word, para 2 bytes, double word, para 4 bytes, quad words para 8 bytes, ...

Bits e Bytes

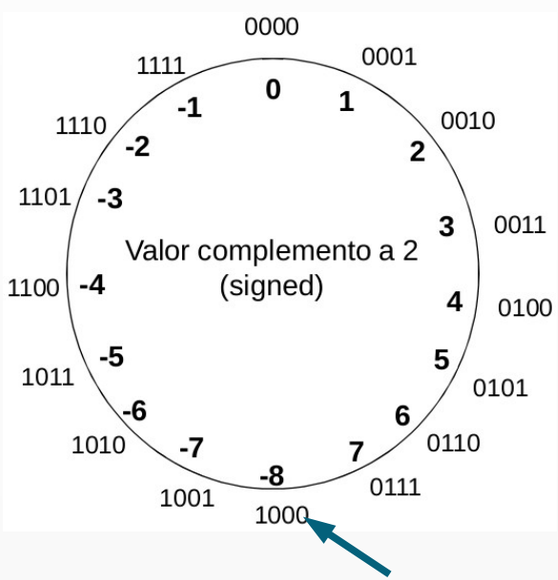
- Quando trabalhamos com tipos numéricos, temos números do tipo char, int, float, double. Esses tipos podem ser com sinal ou sem sinal. O primeiro problema foi como trabalhar com números negativos.
- Assim, com números inteiros positivos (sem sinal) :
  - 1 byte – de 0 a 255 = 2^8 valores
  - 2 bytes – de 0 a 65535=2^{16} valores
  - 4 bytes – de 0 a 4294967295 = 2^{32} valores
- Com sinal:
  - 1 byte – de -128 a 127 = 2^8 valores
  - 2 bytes – de -32768 a 32767=2^{16} valores
  - 4 bytes – de -2147483648 a 2147483647 = 2^{32} valores

Bits e Bytes

- Para entendermos isso, vamos imaginar que fosse possível trabalhar só com sequência de 4 bits, por comodidade.
- Teriámos assim $2^4=16$ números para representar indo então de -8 até 7. Ou, conforme a régua abaixo

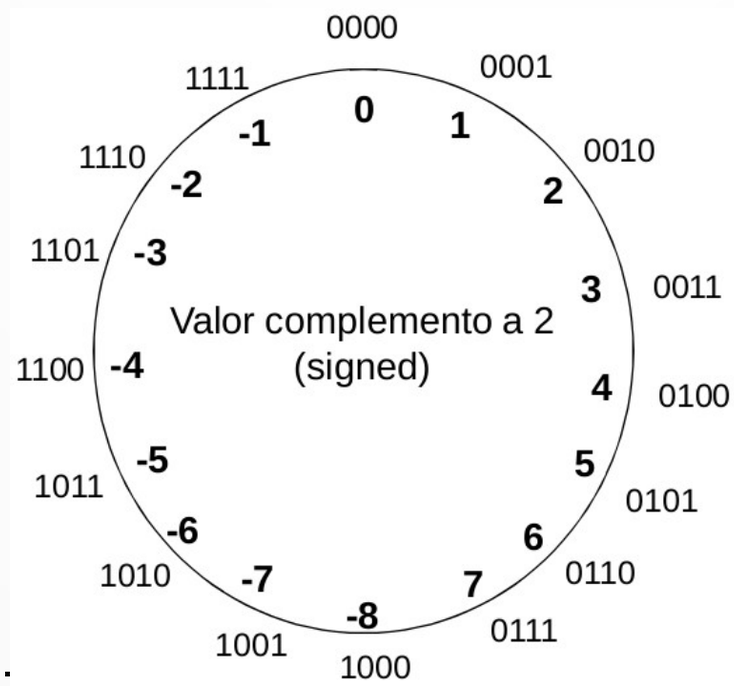


- Vamos “dobrar” essa régua em um relógio de 16 bits.



Bits e Bytes

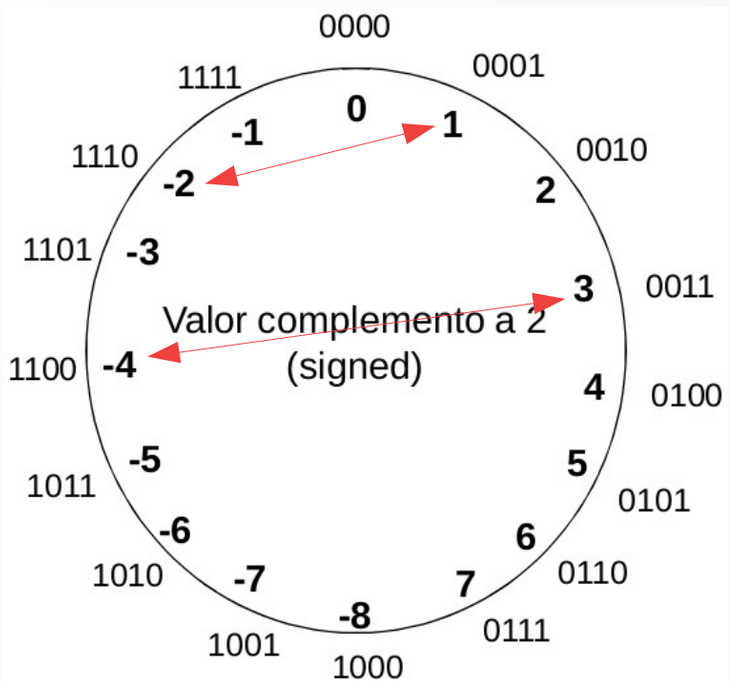
- O relógio começa na “meia noite” ou zero hora e vai até -1. Ou tem a sequencia de valores binários 0000, 0001, ...1110, 1111.



- Verifique que o valor · começando com 1.
- Ou seja, com números com sinal, os números negativos começam com 1.

Bits e Bytes

- Qual foi a percepção que tiveram com isso?
  - Vemos que o complementar de 0001 é 1110, de 0011 é 1100, ou seja, o complementar 1 em decimal é -2 , de 2 é -3, de 3 é -4, e assim por diante até o complementar de 7 que é -8. Obviamente não temos o complementar de 0.
  - Sabemos que -2 + 1 = -1, -3 +1 = -2, ..., -8 + 1 = -7. Ora, para termos o valor negativo de um número positivo temos que ter o seu complemento e somar 1.
- Daí surge a analogia de que para encontrar um número negativo pegamos o seu valor positivo:
  1) Achamos o seu complemento
  2) Somamos 1
- Como são “dois” passos chamamos de complemento de 2.



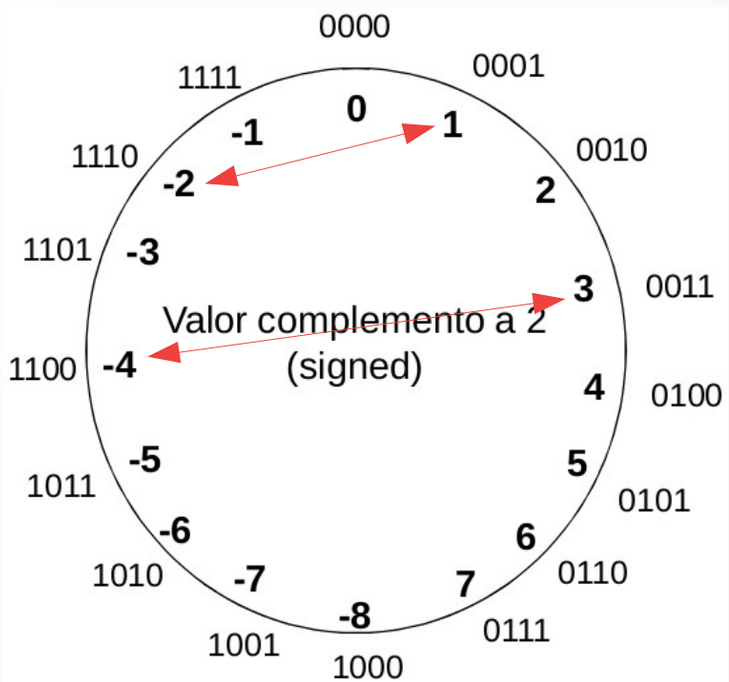
Bits e Bytes

- Assim, um dos modos de verificar qual seria o valor de um número negativo seria
  1) Diminuir 1
  2) Pegar o seu complemento
- Contudo, tem um jeito muito mais elegante, e para isso basta verificar que todo número negativo tem o **bit mais significativo** tendo o **valor 1**. E todo positivo tem 0 como bit mais significativo.
- Assim, como 7 com 4 bits **sem sinal** é

  $0111 = 0x2^3+1x2^2+1x2^1+1x2^0=7$
- Logo -7 com 4 bits será

  $1001 = 1x-2^3+0x2^2+0x2^1+1x2^0=-8+1=-7$
- Supondo agora, que usássemos 7 com sinal

  $0111 = 0x-2^3+1x2^2+1x2^1+1x2^0=0x-8+7=7$, pois o 0 multiplicando o valor negativo anularia o poder do -8, neste caso.

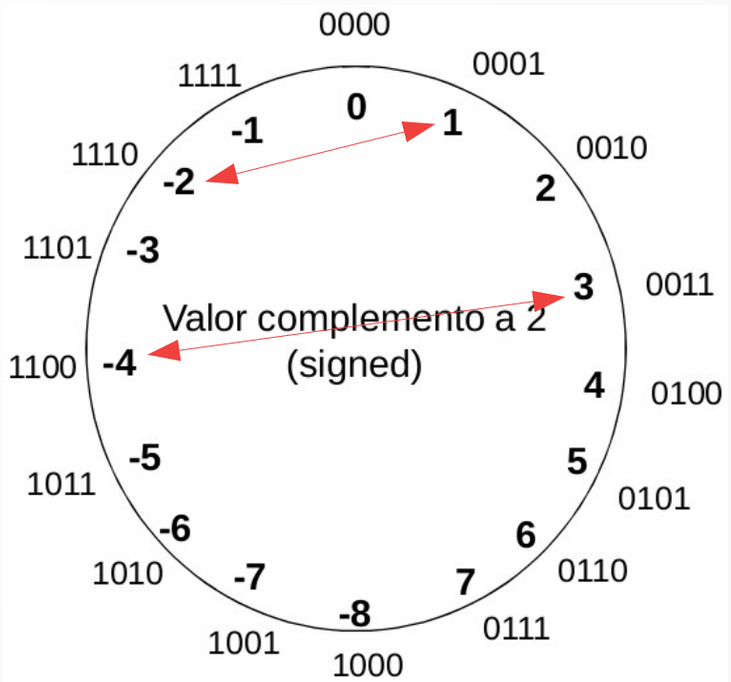


Bits e Bytes

- Um erro comum é esquecer com **quantos bits** estamos fazendo a nossa representação. Tenha sempre em mente que se for um com 4 bits, temos que ter 4 bits, se for representação com 8 bits temos que ter 8 bits. Se positivo, o bit mais significativo é **zero**.
- Como seria 7 em 8 bits? Estamos em numeros com sinal
  - 0000 0011 = **0**x-2^7+ 0x ...+ 0x...+1x2^1+1x2^0 = 0 + 7 = 7
- Como seria -7?
  - Temos que pensar que -2^7=-128
  - Vejamos que -128+**x** = -7 **x** = 128 – 7=121
  - Basta encontra o valor de **x**=121 e adicionar nas 7 casas restantes, já que a 8ª casa terá valor 1.
  - Vemos que $121_d = 0111\ 1001_b$
  - Logo -7_d =**1**111 1001_b =

    =1x-2^7 + 1x2^6 + 1x2^5 +1x2^4 +1x2^3 + 0x2^2 + 0x2^1 + 1x2^0

    = -128 + 121 = -7

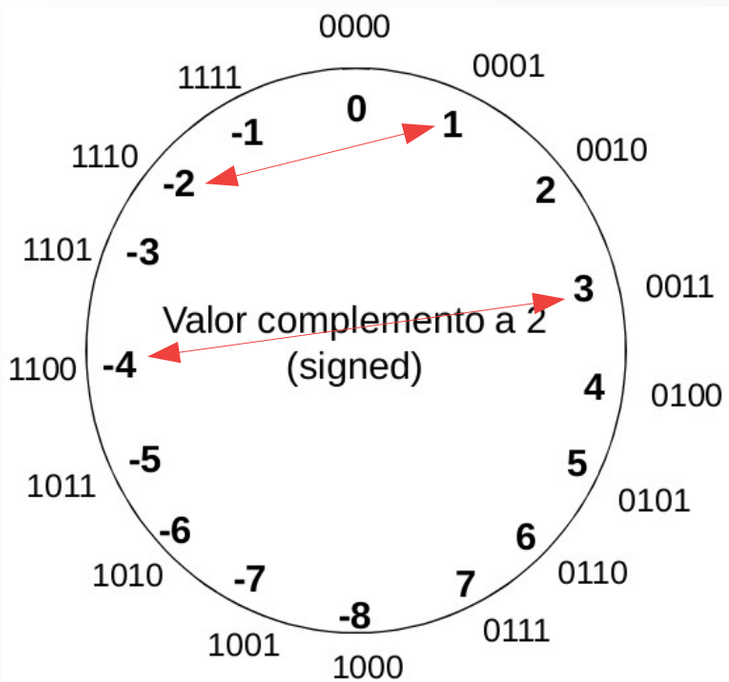


Bits e Bytes

- Para representar um número com n bits com sinal
  - O bit mais significativo ou primeiro bit será multiplicado por -2^{n-1}. No arquivo 2.1.Complemento2.pdf, ele chama o número de bits de w. E usa a seguinte fórmula:

$$B2T(X) = -x_{w-1} \cdot 2^{w-1} + \sum_{i=0}^{w-2} x_i \cdot 2^i$$

  - Se temos w=8, ou seja 8 bits, isso significa que essa fórmula teria a seguinte sequência para **1**001 1001:
  - $\mathbf{-1}.2^7 + 0.2^6 + 0.2^5 + 1.2^4 + 1.2^3 + 0.2^2 + 0.2^1 + 1.2^0 = -128 + 16 + 8 + 1 = -103$



Exercícios

- Enviar os exercícios por email

1) Converta para o sistema decimal os números binários abaixo. Veja que estão em diferentes quantidades de bits, todos com sinal. Lembre que quando o bit mais significativo é 1 o valor será negativo.

a) 100110_2

b) 011110_2

c) 111011_2

d) 1010000_2

e) 11000101_2

f) 11010110_2

g) $0110011001101 01_2$

1) Converta os valores decimais apurados para 16 bits binários