CIRCUITOS DIGITAIS
ARITMÉTICA: MULTIPLICAÇÃO

Marco A. Zanata Alves

Slides baseados nos slides de Rodrigo Hausen - CMCC – UFABC (2013)
http://compscinet.org/hausen/courses/circuitos/

MULTIPLICAÇÃO BINÁRIA

Algoritmo da multiplicação: mesma ideia usada na base decimal.

```
      11011
x       101
  ---------
```

Note que a tabuada da multiplicação na base 2 é muito mais fácil.

×	0	1
0	0	0
1	0	1

MULTIPLICAÇÃO BINÁRIA

Algoritmo da multiplicação: mesma ideia usada na base decimal.

```
      11011
  x     101
  ---------
      11011
     00000
 +  11011
  ---------
    10000111
```

Note que a tabuada da multiplicação na base 2 é muito mais fácil.

×	0	1
0	0	0
1	0	1

MULTIPLICAÇÃO BINÁRIA

Algoritmo da multiplicação: mesma ideia usada na base decimal.

```
     11011
x      101
----------
     11011
    00000
+  11011
----------
  10000111
```

Note que a tabuada da multiplicação na base 2 é muito mais fácil.

Se A tem n algarismos e B tem m algarismos, então o produto $A * B$ terá, no máximo, $\boldsymbol{n + m}$ **algarismos.**

×	0	1
0	0	0
1	0	1

MULTIPLICAÇÃO BINÁRIA

Note que não é necessário armazenar todas as parcelas da soma ao mesmo tempo.

$$\begin{array}{r} 11011 \\ \times\ \underline{00101} \end{array} \qquad \begin{array}{r} 11011 \\ \times\ \underline{00101} \\ 11011 \end{array} \qquad \begin{array}{r} 11011 \\ \times\ \underline{\ 00101} \\ +\ 11011 \\ 00000\color{red}{0} \end{array} \leftarrow \text{desloca 1}$$

MULTIPLICAÇÃO BINÁRIA

Note que não é necessário armazenar todas as parcelas da soma ao mesmo tempo.

$$\begin{array}{r} 11011 \\ \times \underline{00101} \end{array} \qquad \begin{array}{r} 11011 \\ \times \underline{00101} \\ 11011 \end{array} \qquad \begin{array}{rr} & 11011 \\ \times & \underline{\;00101} \\ + & 11011 \\ & 000000 \end{array} \leftarrow \text{desloca 1}$$

$$\begin{array}{rr} & 11011 \\ \times & \underline{\;00101} \\ + & 011011 \end{array}$$

MULTIPLICAÇÃO BINÁRIA

Note que não é necessário armazenar todas as parcelas da soma ao mesmo tempo.

$$\begin{array}{r} 11011 \\ \times \underline{00101} \end{array} \qquad \begin{array}{r} 11011 \\ \times \underline{00101} \\ 11011 \end{array} \qquad \begin{array}{rr} & 11011 \\ \times & \underline{\ \ 00101} \\ + & 11011 \\ & 000000 \end{array} \leftarrow \text{desloca 1}$$

$$\begin{array}{rr} & 11011 \\ \times & \underline{\ \ 00101} \\ + & 011011 \end{array} \qquad \begin{array}{rr} & 11011 \\ \times & \underline{\ \ \ 00101} \\ + & 011011 \\ & 11011\color{red}{00} \end{array} \leftarrow \text{desloca 2} \qquad \begin{array}{rr} & 11011 \\ \times & \underline{\ \ \ 00101} \\ + & 10000111 \end{array}$$

NÚMEROS REAIS

NÚMEROS REAIS

O que acontece com os algoritmos da soma, subtração, multiplicação e divisão quando os números sendo operados não são inteiros?

NÚMEROS REAIS

O que acontece com os algoritmos da soma, subtração, multiplicação e divisão quando os números sendo operados não são inteiros?

Sem perder a generalidade, iremos supor que A e B possuem k algarismos depois da vírgula.

E se eles não tiverem a mesma quantidade de algarismos após a vírgula?

$$A = a_{n-1}\, a_{n-2}\, \dots a_2\, a_1\, a_0\,, a_{-1} \dots a_{-k}$$
$$B = b_{m-1}\, b_{m-2}\, \dots b_2\, b_1\, b_0\,, b_{-1} \dots b_{-k}$$

NÚMEROS REAIS

Para a **soma** e a **subtração**: como os algoritmos são “copiados” da versão para números na base 10, a solução é simples: ignore, inicialmente a vírgula. Após a soma, recoloque a vírgula no seu lugar (conte $\boldsymbol{k}$ **algarismos à direita**).

NÚMEROS REAIS

Para a **soma** e a **subtração**: como os algoritmos são “copiados” da versão para números na base 10, a solução é simples: ignore, inicialmente a vírgula. Após a soma, recoloque a vírgula no seu lugar (conte k **algarismos à direita**).

Para a **multiplicação**: de novo, a inspiração vem da base decimal. Ignore, inicialmente a vírgula e, após a multiplicação, recoloque a vírgula no seu lugar (conte $2 * k$ **algarismos à direita**).

MULTIPLICAÇÃO BINÁRIA DE NÚMEROS REAIS

Após a multiplicação, recoloque a vírgula no seu lugar (conte $2 * k$ algarismos à direita).

$$\begin{array}{r} 110{,}1 \\ \times\ \underline{010{,}0} \end{array} \qquad\qquad \begin{array}{r} 1101 \\ \times\ \underline{0100} \end{array}$$

MULTIPLICAÇÃO BINÁRIA DE NÚMEROS REAIS

Após a multiplicação, recoloque a vírgula no seu lugar (conte $2 * k$ algarismos à direita).

$$\begin{array}{r} 110{,}1 \\ \times\ \underline{010{,}0} \end{array}$$

$$\begin{array}{r} 1101 \\ \times\ \underline{0100} \end{array}$$

$$\begin{array}{r} 1101 \\ \times\ \underline{010\color{red}{0}} \\ 0000 \end{array}$$

MULTIPLICAÇÃO BINÁRIA DE NÚMEROS REAIS

Após a multiplicação, recoloque a vírgula no seu lugar (conte $2 * k$ algarismos à direita).

$$\begin{array}{r} 110{,}1 \\ \times\ \underline{010{,}0} \end{array} \qquad \begin{array}{r} 1101 \\ \times\ \underline{0100} \end{array} \qquad \begin{array}{r} 1101 \\ \times\ \underline{0100} \\ 0000 \end{array}$$

$$\begin{array}{r} 1101 \\ \times\ \underline{\ \ 0100} \\ +\ \ 0000 \\ 0000\color{red}{0} \end{array} \qquad \begin{array}{r} 1101 \\ \times\ \underline{\ \ \ \ 0100} \\ +\ \ \ \ 0000 \\ 0000\color{red}{0} \\ \color{red}{110100} \end{array}$$

Desloca 1 casa

Desloca 2,3 casas

MULTIPLICAÇÃO BINÁRIA DE NÚMEROS REAIS

Após a multiplicação, recoloque a vírgula no seu lugar (conte $2 * k$ algarismos à direita).

$$\begin{array}{r} 110{,}1 \\ \times\ \underline{010{,}0} \end{array} \qquad \begin{array}{r} 1101 \\ \times\ \underline{0100} \end{array} \qquad \begin{array}{r} 1101 \\ \times\ \underline{0100} \\ 0000 \end{array}$$

$$\begin{array}{r} 1101 \\ \times\ \underline{0100} \\ +\ 0000 \\ 00000 \end{array} \qquad \begin{array}{r} 1101 \\ \times\ \underline{0100} \\ +\ 0000 \\ 00000 \\ 110100 \end{array} \qquad \begin{array}{r} 110{,}1 \\ \times\ \underline{010{,}0} \\ 1101{,}00 \end{array}$$

Desloca 1 casa

Desloca 2,3 casas

Re-colocamos a vírgula

REPRESENTAÇÃO NUMÉRICA

REPRESENTAÇÃO NUMÉRICA

Representação de números no papel: usamos tantos dígitos forem necessários.

Limitado apenas pela quantidade de papel, tempo disponível para escrever os dígitos, paciência. . .

REPRESENTAÇÃO NUMÉRICA

Representação de números no papel: usamos tantos dígitos forem necessários.

Limitado apenas pela quantidade de papel, tempo disponível para escrever os dígitos, paciência. . .

Número π:

```
3.1415926535897932384626433832795028841971693993751058
209749445923078164062862089986280348253421170679821480
865132823066470938446095505822317253594081284811174502
841027019385211055596446229489549303819644288109756659
334461284756482337867831652712019091456485669234603486
104543266482133936072602491412737245870066063155881748
815209209628292540917153643678925903600113305305488204
665213841469519415116094330572703657595919530921861173
81932611793105118548074462379...
```

REPRESENTAÇÃO NUMÉRICA NUM COMPUTADOR DIGITAL

Recordando: em um computador digital **qualquer informação**, em última instância, é **representada por um número**.

Atualmente, os números são representados **internamente em binário** (por vários motivos, entre eles facilidade de fazer contas na **base 2**).

Um computador digital possui **espaço finito** para guardar informações.

Por questões de eficiência, geralmente o processamento de dados (ou seja, números) não é feito algarismo binário por algarismo binário, e sim por **grupos de algarismos binários** de uma só vez.

BITS E PALAVRAS

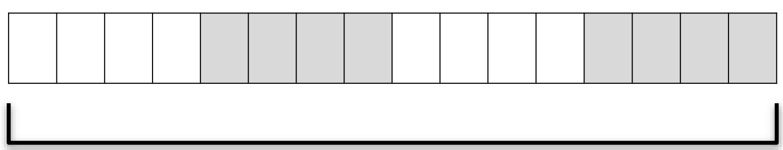
Abreviação: algarismo binário = bit (do inglês ***b****inary dig****it***)

BITS E PALAVRAS

Abreviação: algarismo binário = bit (do inglês ***b***_inary dig_***it***)

A unidade natural de processamento de um determinado sistema é chamada **palavra de dado**

Trata-se de uma sequência de bits com tamanho fixo que é processada em conjunto.



Tamanho w = 16 bits

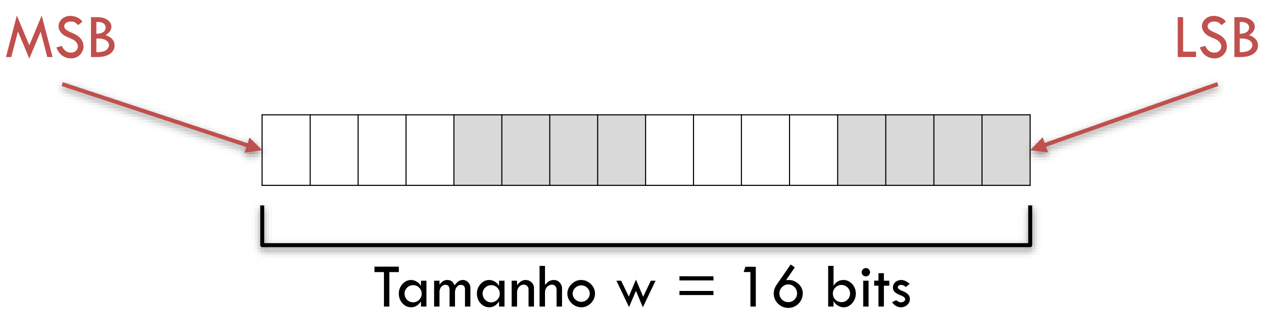
Essa palavra tem 16 bits

BITS E PALAVRAS

Abreviação: algarismo binário = bit (do inglês ***bi**nary digi**t***)

A unidade natural de processamento de um determinado sistema é chamada **palavra de dado**

Trata-se de uma sequência de bits com tamanho fixo que é processada em conjunto.



MSB = Most Significant Bit = bit mais significativo
LSB = Least Significant Bit = bit menos significativo

BITS E PALAVRAS

Nomes comuns para conjunto de bits:

… **8 bits = byte** (binary term) ou octeto

… 4 bits = nibble

(nibble, em inglês, significa “mordidinha” = “small bite”)

Atenção: $10Mb/s \neq 10MB/s$

BITS E PALAVRAS

Um sistema digital pode padronizar o tamanho de seus operandos.

Por exemplo, podemos ter um processador de 32 bits ou 64 bits.

Nesse caso dizemos que a **palavra** tem 32 ou 64 bits, respectivamente.

Note que a palavra pode mudar de tamanho em cada sistema.

Mas as unidades de medida (bytes, nibble, MB, Mb, etc.) não.

REPRESENTAÇÃO BINÁRIA

REPRESENTANDO NÚMEROS EM PALAVRAS BINÁRIAS

Qual é o **maior inteiro** sem sinal que podemos representar?

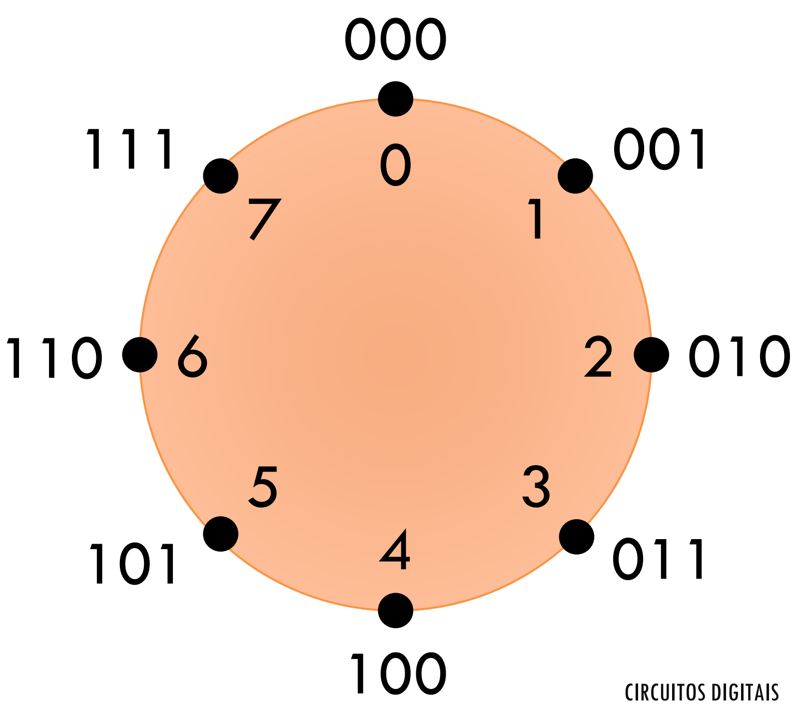
Exemplo: quais inteiros sem sinal podemos representar com 3 bits?

REPRESENTANDO NÚMEROS EM PALAVRAS BINÁRIAS

Qual é o **maior inteiro** sem sinal que podemos representar?

Exemplo: quais inteiros sem sinal podemos representar com 3 bits?

De 0 até $7 = 2^3 - 1$



REPRESENTANDO NÚMEROS: INTEIROS SEM SINAL

Inteiros sem sinal em palavras binárias com w bits.

Palavra		**Decimal**
00...000	=	0
00...001	=	1
00...010	=	2
	...	
11...110	=	?
11...111	=	? maior inteiro sem sinal com w bits

O próximo número na sequência, que não cabe em *w* bits, é…

REPRESENTANDO NÚMEROS: INTEIROS SEM SINAL

Inteiros sem sinal em palavras binárias com w bits.

Palavra		**Decimal**
00...000	=	0
00...001	=	1
00...010	=	2
	...	
11...110	=	$2^w - 2$
11...111	=	$2^w - 1$ = maior inteiro sem sinal com w bits
100...000	=	2^w

Quantas combinações diferentes temos com N bits?

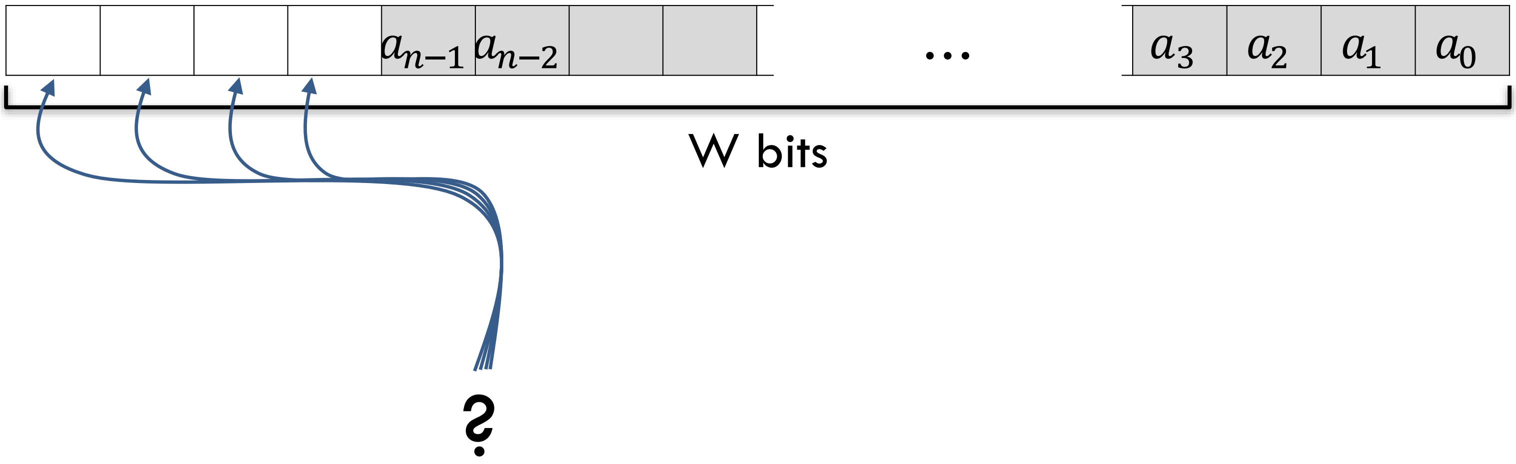
O próximo número na sequência, que não cabe em w bits, é

$$(100\ldots000)_2 = 2^w$$

EXTENSÃO DE NÚMEROS SEM SINAL

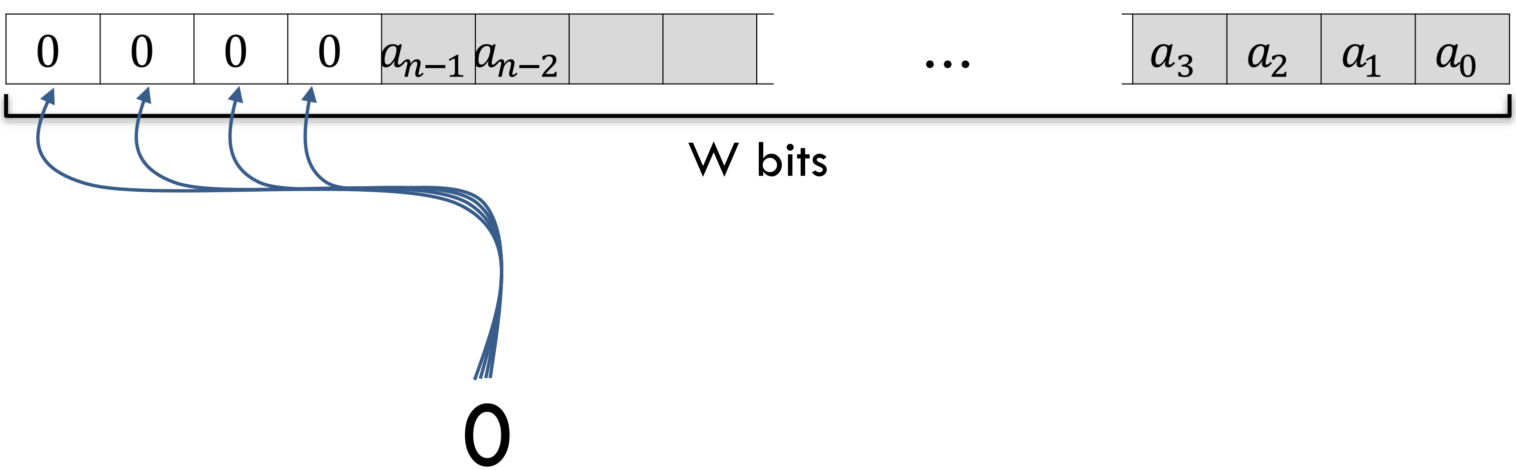
EXTENSÃO DE NÚMEROS SEM SINAL

Como representar um número **inteiro sem sinal**
$A = (a_{n-1}\, a_{n-2}\, \ldots a_2\, a_1\, a_0)_2$ numa palavra de comprimento $W \geq n$?



EXTENSÃO DE NÚMEROS SEM SINAL

Como representar um número **inteiro sem sinal**
$A = (a_{n-1}\, a_{n-2}\, \ldots a_2\, a_1\, a_0)_2$ numa palavra de comprimento $W \geq n$?



REPRESENTANDO NÚMEROS NEGATIVOS

REPRESENTANDO NÚMEROS: INTEIROS COM SINAL

Precisamos reservar espaço na palavra para representar, além dos algarismos do número, alguma informação sobre o sinal.

Existem várias possibilidades para o sinal:

REPRESENTANDO NÚMEROS: INTEIROS COM SINAL

Precisamos reservar espaço na palavra para representar, além dos algarismos do número, alguma informação sobre o sinal.

Existem várias possibilidades para o sinal:

- Podemos usar um dos bits para representar o sinal.
- Podemos usar complemento de 1.
- Podemos usar complemento de 2.

SINAL MAGNITUDE

REPRESENTAÇÃO SINAL-MAGNITUDE

Esta representação é conhecida como **sinal-magnitude.**

Sinal +: bit de sinal 0

Sinal −: bit de sinal 1

s	a_{w-2}	a_{w-3}	…	a_2	a_1	a_0

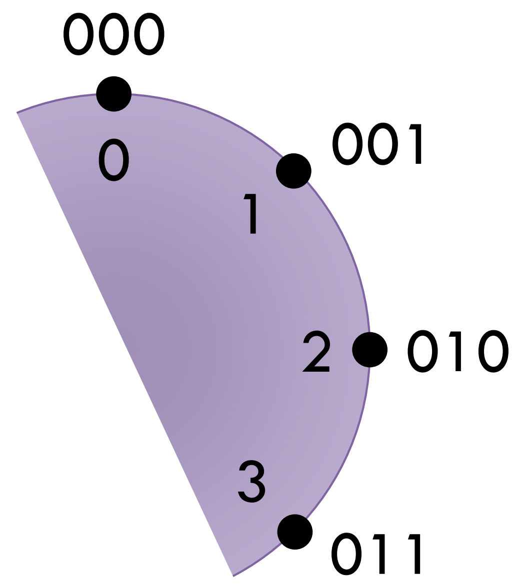
W bits

REPRESENTAÇÃO SINAL-MAGNITUDE

Esta representação é conhecida como **sinal-magnitude**.

Sinal +: bit de sinal 0

Sinal –: bit de sinal 1

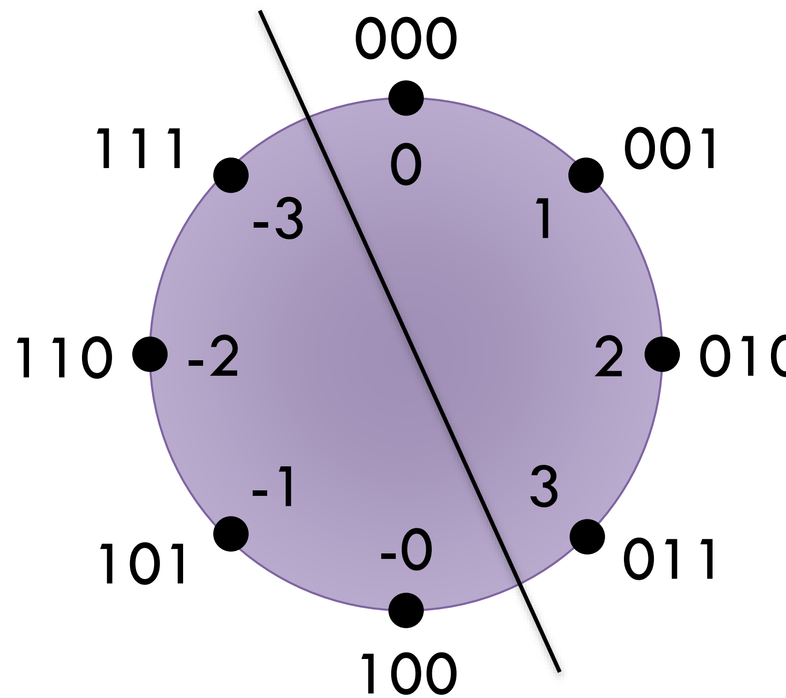


REPRESENTAÇÃO SINAL-MAGNITUDE

Esta representação é conhecida como **sinal-magnitude.**

Sinal +: bit de sinal 0

Sinal –: bit de sinal 1



REPRESENTAÇÃO SINAL-MAGNITUDE

Menor número:

$$111\ldots111 = -(\underbrace{11\ldots111}_{\text{W-1 uns}})_2 = -(100\ldots000 - 1)_2 = -2^{w-1} + 1$$

$$\underbrace{\begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 1 & 1 & \ldots & 1 & 1 & 1 \\ \hline\end{array}}_{\text{W bits}}$$

REPRESENTAÇÃO SINAL-MAGNITUDE

Menor número:

$$111\ldots111 = -(11\ldots111)_2 = -(100\ldots000 - 1)_2 = -2^{w-1} + 1$$

Maior número:

$$011\ldots111 = +(\underbrace{11\ldots111}_{\text{W-1uns}})_2 = +(100\ldots000 - 1)_2 = 2^{w-1} - 1$$

0	1	1	...	1	1	1

W bits

REPRESENTAÇÃO SINAL-MAGNITUDE

Vantagens:

- Simples de entender
- Simples de implementar

Desvantagens:

- Zero tem duas representações:
- $00\ldots000 = +0$
- $10\ldots000 = -0$
- Complica a aritmética: é necessário tratar o sinal separadamente na hora de fazer as contas de soma e subtração.

EXTENSÃO DE NÚMEROS EM SINAL-MAGNITUDE

EXTENSÃO DE NÚMEROS EM SINAL-MAGNITUDE

Como representar um número inteiro $A = (a_{n-1}\, a_{n-2}\, \ldots a_2\, a_1\, a_0)_2$

numa palavra de comprimento $W \geq n$?

Se for positivo, e se for negativo?

?	?	?	?	a_{n-1}	a_{n-2}			$\ldots$	a_3	a_2	a_1	a_0

$\underbrace{\hspace{10cm}}_{\text{W bits}}$

EXTENSÃO DE NÚMEROS EM SINAL-MAGNITUDE

Como representar um número inteiro $A = (a_{n-1}\, a_{n-2}\ \dots a_2\, a_1\, a_0)_2$ numa palavra de comprimento $W \geq n$?

Se positivo

a_{n-1}	0	0	0	0	a_{n-2}			…	a_3	a_2	a_1	a_0

W bits

Se negativo

a_{n-1}	0	0	0	0	a_{n-2}			…	a_3	a_2	a_1	a_0

W bits

COMPLEMENTO DE 1

COMPLEMENTO DE 1

O processo de conversão ocorre apenas se o número for negativo

Trata-se de uma forma simples de converter uma representação binária sem sinal para uma representação com sinal

COMPLEMENTO DE 1

O processo de conversão ocorre apenas se o número for negativo

Trata-se de uma forma simples de converter uma representação binária sem sinal para uma representação com sinal

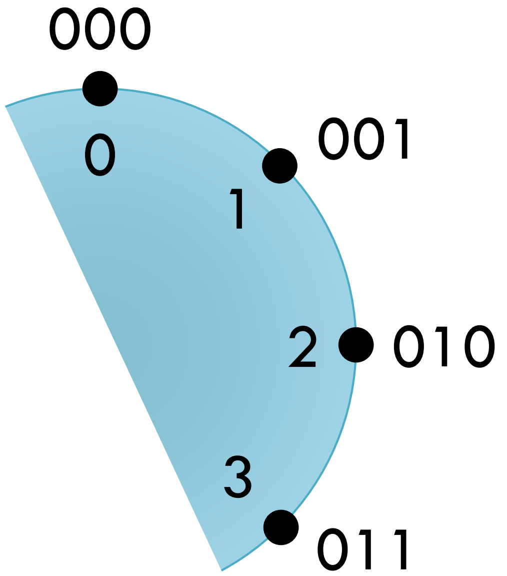
Matematicamente fazemos: $111111 - bbbbb = (\quad)_{c1}$

Para realizar a conversão basta inverter os níveis lógicos do número

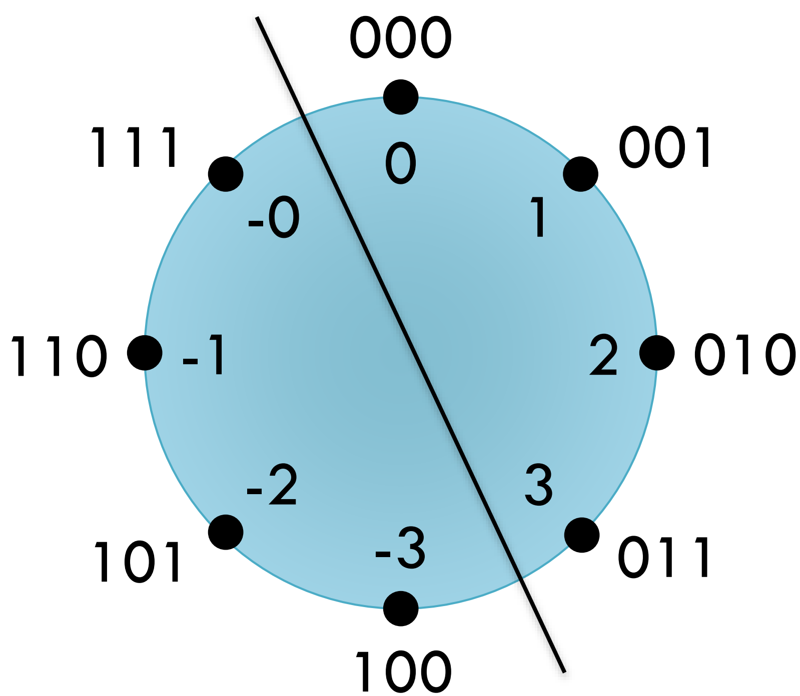
O bit mais significativo irá indicar o sinal:

- 0 se positivo
- 1 se negativo

REPRESENTAÇÃO EM COMPLEMENTO DE 1



REPRESENTAÇÃO EM COMPLEMENTO DE 1



REPRESENTAÇÃO EM COMPLEMENTO DE 1

Vantagens:

- Conversão simples
- Somas e subtrações são feitas da mesma forma que para números sem sinal

Desvantagens:

- Duas representações para o zero
- Comparações não são tão simples. Ex.: $(-1)_{10} = (\mathbf{110})_\mathbf{2} > (\mathbf{100})_\mathbf{2} = (-3)_{10}$

SUBTRAINDO COM COMPLEMENTO DE 1

Vamos considerar a subtração de $3 - 2$.

Iniciamos com o complemento de 1, do número 2.

$$010 \rightarrow 101_{c1}$$

SUBTRAINDO COM COMPLEMENTO DE 1

Vamos considerar a subtração de $3-2$.

Iniciamos com o complemento de 1, do número 2.

$$010 \rightarrow 101_{c1}$$

Depois fazemos a soma $A+(-B)$

$$\begin{array}{r} 11 \\ 011 \\ 101 \\ \hline 1\underline{000} \end{array}$$

$$3-2=0\,???$$

Cuidando para adicionar 1 ao final.

SUBTRAINDO COM COMPLEMENTO DE 1

Vamos considerar a subtração de $3 - 2$.

Iniciamos com o complemento de 1, do número 2.

$$010 \rightarrow 101_{c1}$$

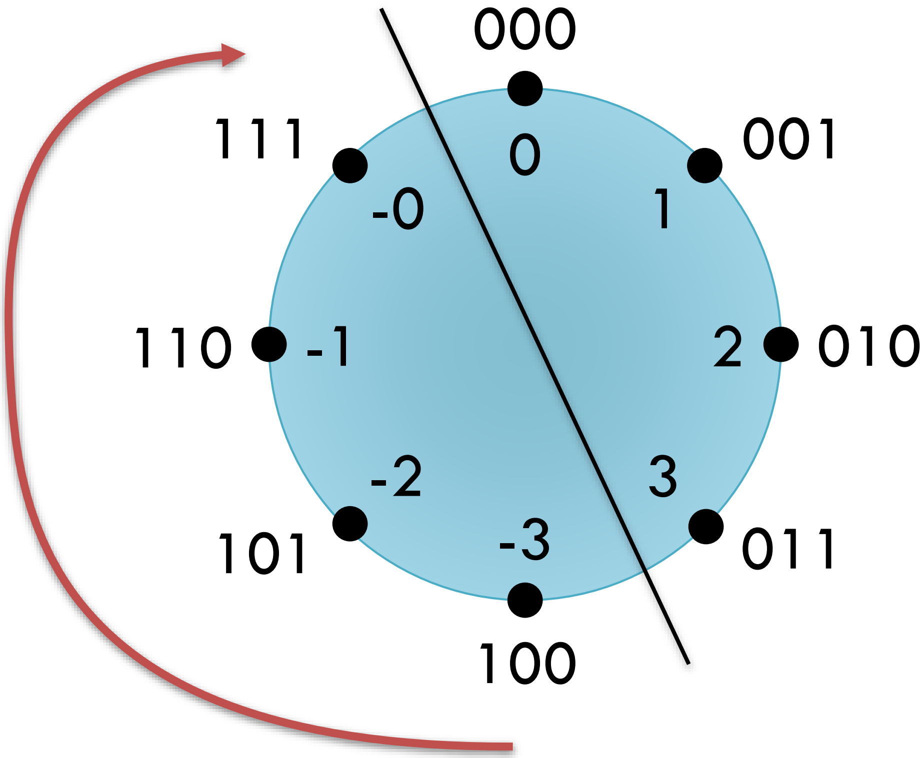
Depois fazemos a soma $A + (-B)$

$$\begin{array}{r} 11 \\ 011 \\ 101 \\ \hline 1000 \\ +\ \ 1 \\ \hline \boxed{001} \end{array}$$

COMPLEMENTO DE 2

REPRESENTAÇÃO EM COMPLEMENTO DE 1

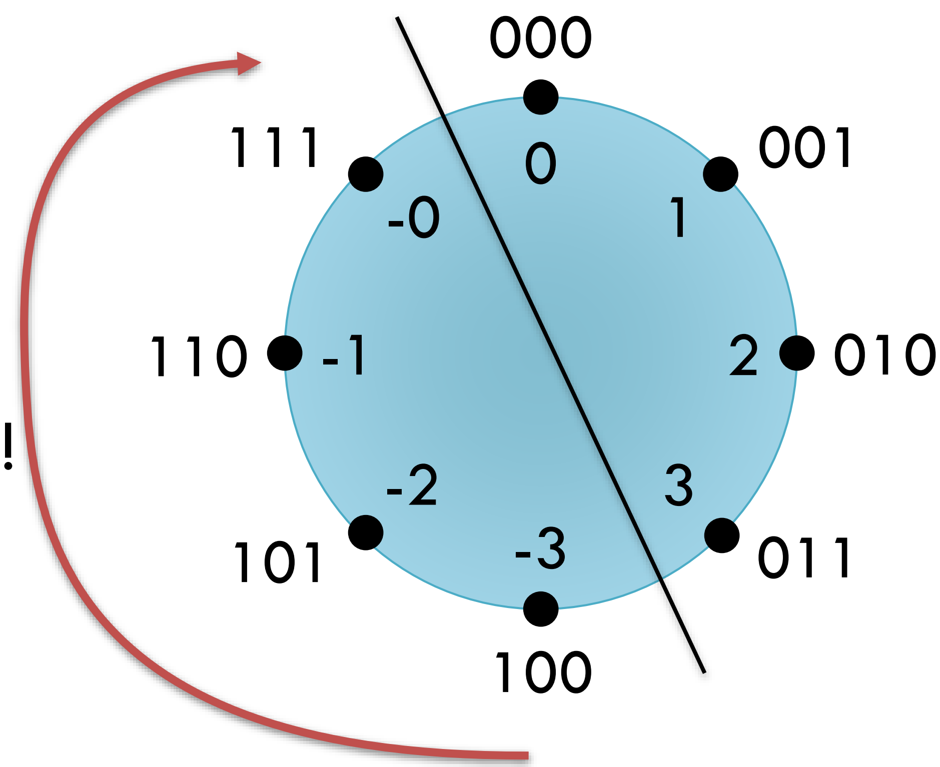
Que tal deslocar 1 posição?!



REPRESENTAÇÃO EM COMPLEMENTO DE 1

- Podemos ter apenas 1 representação para zero!
- Podemos representar um número negativo a mais!

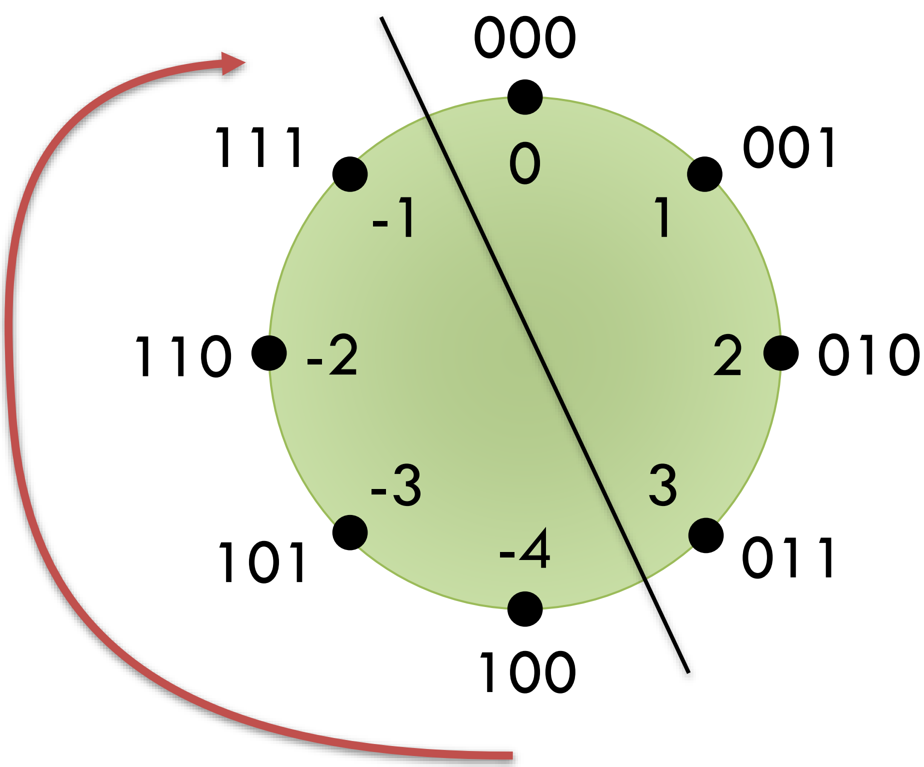
Que tal deslocar 1 posição?!



REPRESENTAÇÃO EM COMPLEMENTO DE 2

- Podemos ter apenas 1 representação para zero!
- Podemos representar um número negativo a mais!

Que tal deslocar 1 posição?!



REPRESENTAÇÃO EM COMPLEMENTO DE 2

Inteiros representados em complemento de dois em palavras de 3 bits:

$011 = +3_{10}$

$010 = +2_{10}$

$001 = +1_{10}$

$000 = 0_{10}$

$111 = ?_{10}$

$110 = ?_{10}$

$101 = ?_{10}$

$100 = ?_{10}$

REPRESENTAÇÃO EM COMPLEMENTO DE 2

Inteiros representados em complemento de dois em palavras de 3 bits:

$011 = +3_{10}$

$010 = +2_{10}$

$001 = +1_{10}$

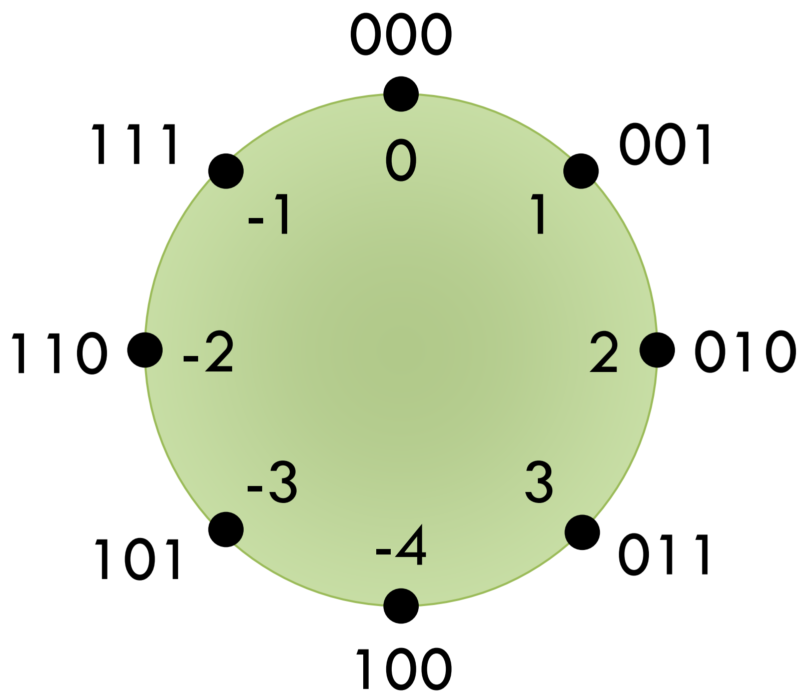
$000 = 0_{10}$

$111 = -\left(\overline{11} + 1\right)_2 = -(01)_2 = -1_{10}$

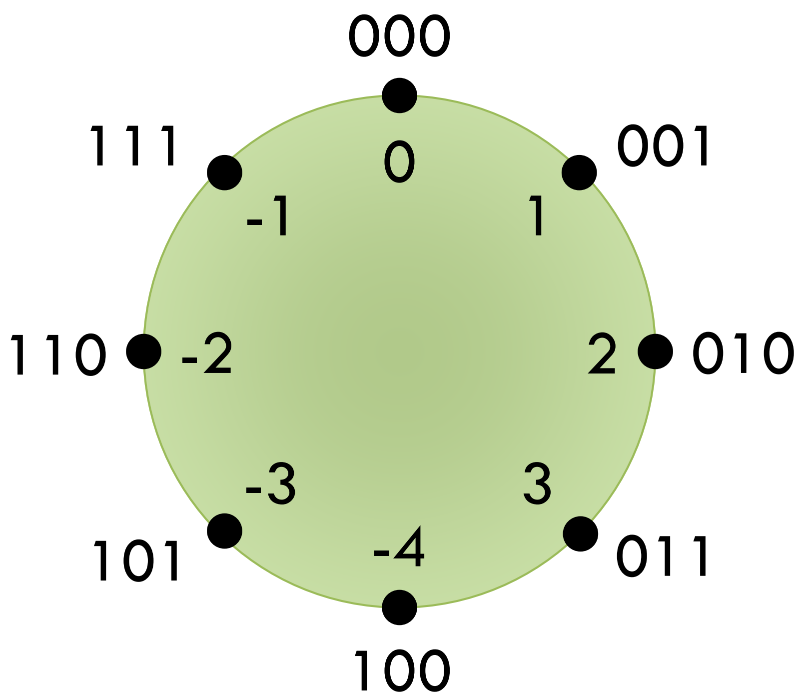
$110 = -\left(\overline{10} + 1\right)_2 = -(11)_2 = -2_{10}$

$101 = -\left(\overline{01} + 1\right)_2 = -(10)_2 = -3_{10}$

$100 = -\left(\overline{00} + 1\right)_2 = -(100)_2 = -4_{10}$



REPRESENTAÇÃO EM COMPLEMENTO DE 2



Note que o intervalo de representação não é simétrico

Como só há uma representação para 0, é possível representar um inteiro negativo a mais

somas/subtrações com esta representação são simples!

$A - B = A + (-B)$

REPRESENTAÇÃO EM COMPLEMENTO DE 2

Logo, podemos notar que caso o número esteja em representação de complemento de 2, devemos ignorar o primeiro digito mais significativo, pois este indica o sinal.

Caso o número seja negativo, a conversão entre bases deve acontecer após a conversão para complemento de 2.

REPRESENTAÇÃO EM COMPLEMENTO DE 2

Menor número:

$$100\ldots000 = -(0\underbrace{1\ldots111}_{\text{W-1 uns}} + 1)_2 = -(\mathbf{1}00\ldots000)_2 = -2^{w-1}$$

1	1	1	...	1	1	1

W bits

REPRESENTAÇÃO EM COMPLEMENTO DE 2

Menor número:

$$100\ldots000 = -(01\ldots111 + 1)_2 = -(\mathbf{1}00\ldots000)_2 = -2^{w-1}$$

Maior número:

$$011\ldots111 = +(0\underbrace{11\ldots111}_{\text{W-1 uns}})_2 = 2^{w-1} - 1$$

Assim como sinal magnitude!

0	1	1	...	1	1	1

W bits

REPRESENTAÇÃO EM COMPLEMENTO DE 2

Vantagens:

- Representação única para o zero
- Somas e subtrações são feitas da mesma forma que para números sem sinal

Desvantagens:

- Não é tão intuitivo para nós (indiferente para computador)
- Comparações não são tão simples. Ex.: $(1)_{10} = (\mathbf{001})_{\mathbf{2}} > (\mathbf{101})_{\mathbf{2}} = (-3)_{10}$

EXTENSÃO DE NÚMEROS EM COMPLEMENTO DE 2

EXTENSÃO DE NÚMEROS EM COMPLEMENTO DE 2

Como representar um número inteiro em **complemento de 2**
$A = (a_{n-1}\, a_{n-2}\, \ldots a_2\, a_1\, a_0)_2$ numa palavra de comprimento $W \geq n$?
Se for positivo, e se for negativo?

EXTENSÃO DE NÚMEROS EM COMPLEMENTO DE 2

Como representar um número inteiro em **complemento de 2** $A = (a_{n-1}\, a_{n-2}\, \ldots a_2\, a_1\, a_0)_2$ numa palavra de comprimento $W \geq n$?
Se for positivo, e se for negativo?

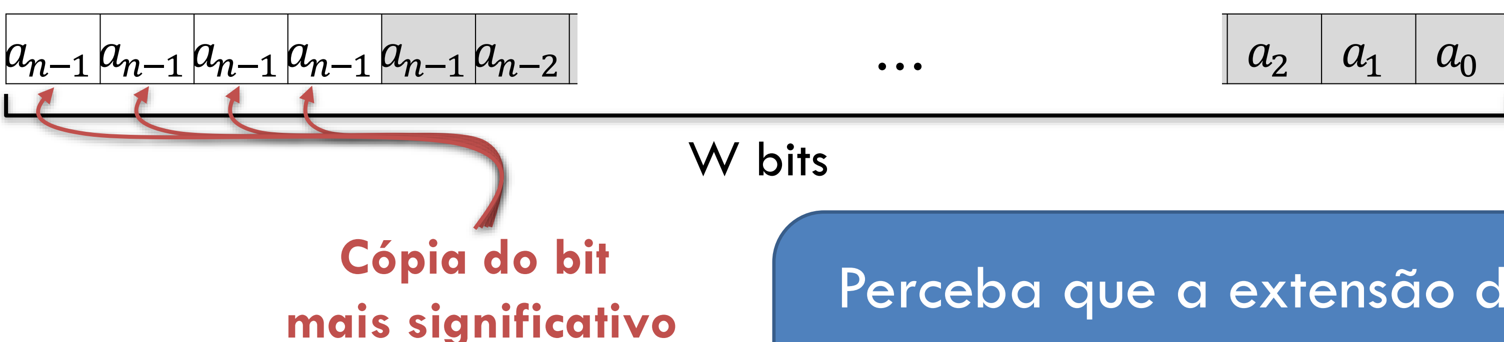
Se positivo

Se negativo

EXTENSÃO DE NÚMEROS EM COMPLEMENTO DE 2

Como representar um número inteiro em complemento de 2
$A = (a_{n-1}\, a_{n-2}\, \ldots a_2\, a_1\, a_0)_2$ numa palavra de comprimento $W \geq n$?
Se for positivo, e se for negativo?

Se **positivo ou negativo**

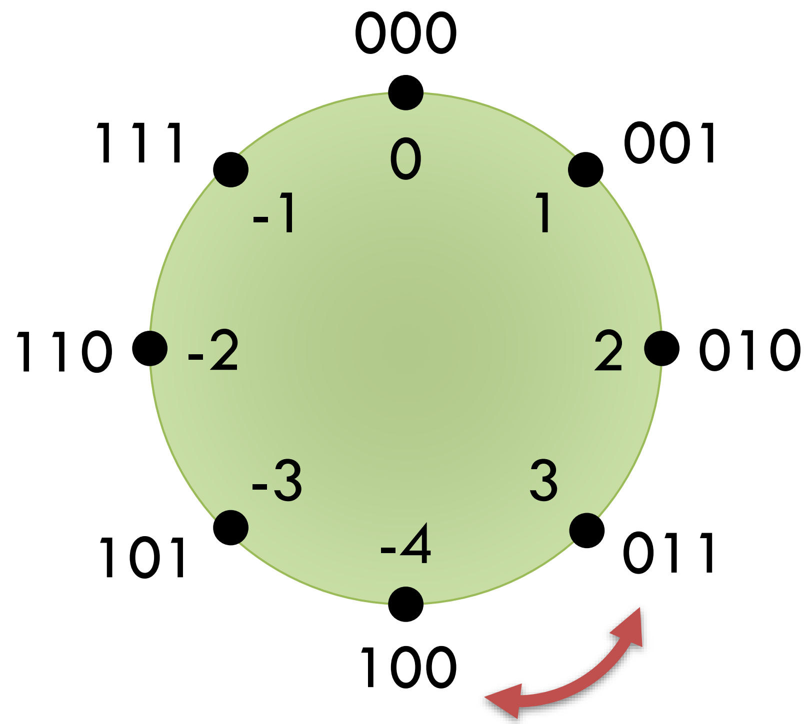


Perceba que a extensão de sinal será diferente do que fazíamos para números sem sinal!

LIMITAÇÕES EM ADIÇÕES

LIMITAÇÕES NA REPRESENTAÇÃO

Toda vez que uma operação precisar de mais bits do que os disponíveis na representação que estamos utilizando um erro (overflow/underflow) ocorre.



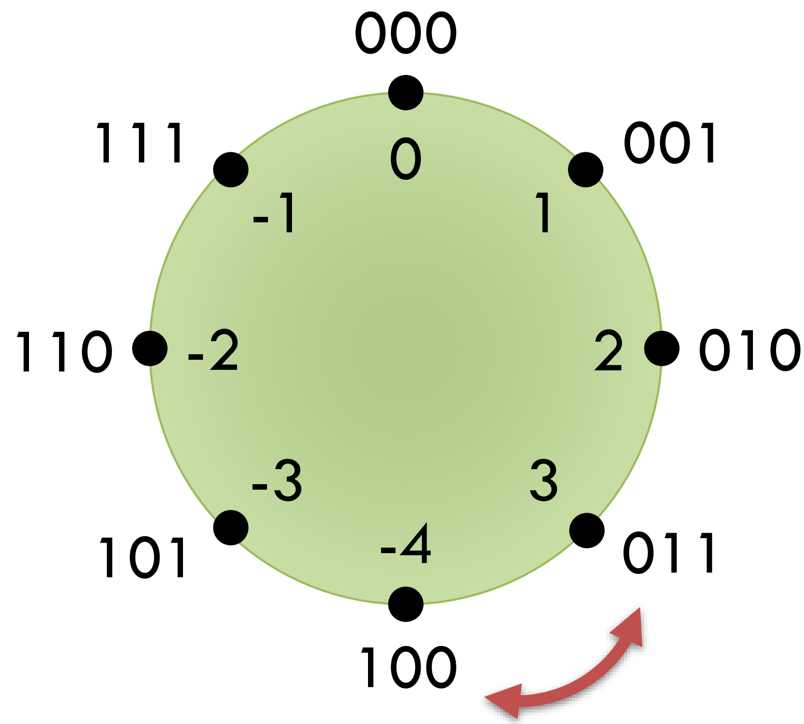
LIMITAÇÕES NA REPRESENTAÇÃO

Toda vez que uma operação precisar de mais bits do que os disponíveis na representação que estamos utilizando um erro (overflow/underflow) ocorre.

Considere que estamos utilizando complemento de dois, com representação de 3 bits.

Ao somar +1 no número 011 (3_{10}) causamos um erro **(overflow)**, devido à representação com 3 bits.

$$011 + 1 = 100 = (-4_{10})$$



DETECÇÃO DE OVERFLOW/UNDERFLOW

A ocorrência de overflow pode ser detectada examinando-se o bit de sinal do resultado e comparando-o com os bits de sinal dos números que estão sendo adicionados.

Nos computadores, um circuito especial é usado para detectar qualquer condição de overflow para indicar que a resposta está errada.

Overflow/Underflow só ocorre quando somando dois números de mesmo sinal (positivos ou negativos).

Quando temos dois números de mesmo sinal, devemos verificar se o resultado da soma tem mesmo sinal dos operadores.

Caso negativo temos um overflow/underflow.

PROBLEMAS DE CONVERSÃO

Sempre que queremos adicionar o sinal, na representação do número iremos precisar de um bit livre para isso (representar o sinal).

Toda vez que fizermos uma conversão de inteiro sem sinal para inteiro com sinal, podemos ter um overflow.

Isso ocorre caso o bit mais significativo do inteiro sem sinal esteja sendo usado.