Operações lógicas

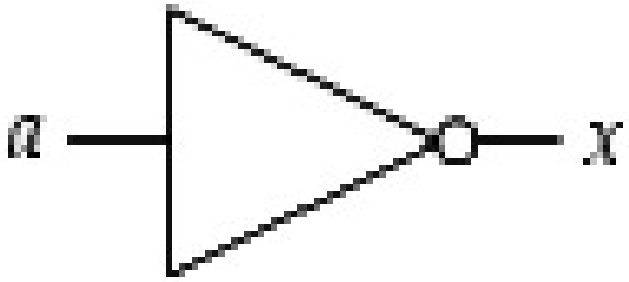
- As operações lógicas são estudadas pela álgebra de boole (George Boole)
- A álgebra de Boole trabalha com apenas duas grandezas: **falso** ou **verdadeiro**.
- As duas grandezas são representadas por **0** (falso) e **1** (verdadeiro).
- Nos circuitos lógicos do computador, os sinais binários são representados por níveis de tensão.

Portas lógicas

- As portas lógicas são os elementos mais básicos e elementares de um sistema de computação.
- Elas são responsáveis por realizar as operações lógicas sobre os bits.
- Os valores de entrada e saída são números binários.
- Cada porta lógica realiza uma tarefa trivial.

Portas lógicas

- **NOT**: inverte a entrada.

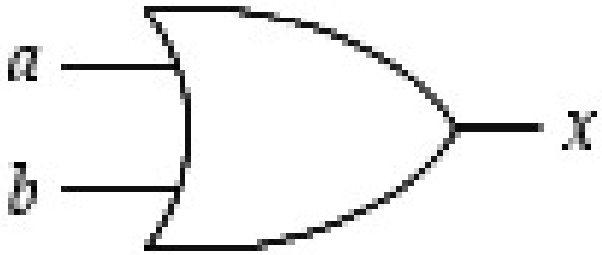


NOT	
a	x
0	1
1	0

Expressão: x = a' ou $x = \bar{a}$

Portas lógicas

- **OR**: retorna 1 se uma das entradas é 1.

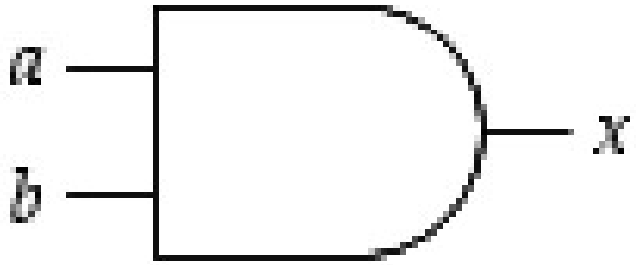


OR		
a	b	x
0	0	0
1	0	1
0	1	1
1	1	1

Expressão: x = a + b

Portas lógicas

- **AND**: retorna 1 se ambas as entradas são 1.

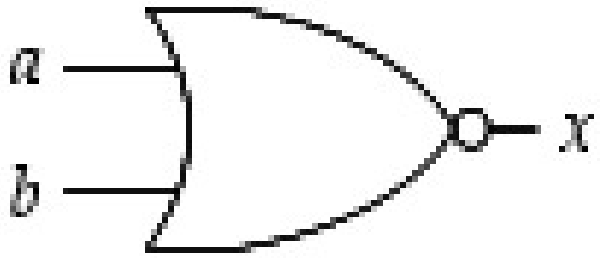


AND		
a	b	x
0	0	0
1	0	0
0	1	0
1	1	1

Expressão: $x = a \times b$

Portas lógicas

- **NOR**: é uma porta OR e uma porta NOT combinadas. O resultado é exatamente o inverso da porta OR.

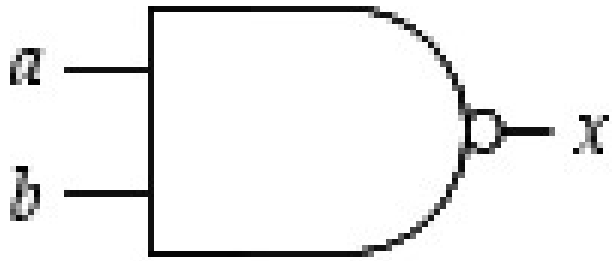


NOR		
a	b	x
0	0	1
1	0	0
0	1	0
1	1	0

Expressão: x = (a + b)'

Portas lógicas

- **NAND**: é uma porta AND e uma porta NOT combinadas. O resultado é exatamente o inverso da porta AND.

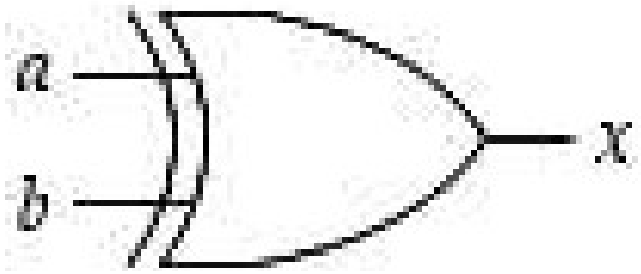


NAND		
a	b	x
0	0	1
1	0	1
0	1	1
1	1	0

Expressão: x = (a x b)'

Portas lógicas

- **XOR**: retorna 1 somente se uma das entradas é 1.

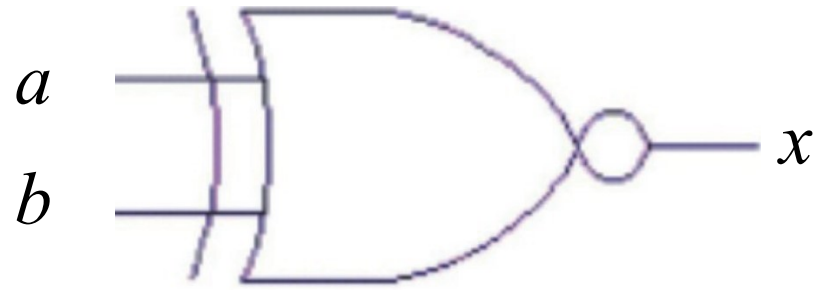


XOR		
a	b	x
0	0	0
1	0	1
0	1	1
1	1	0

Expressão: $x = a \oplus b$

Portas lógicas

- **NXOR**: é uma porta XOR e uma porta NOT combinadas. O resultado é exatamente o inverso da porta XOR.



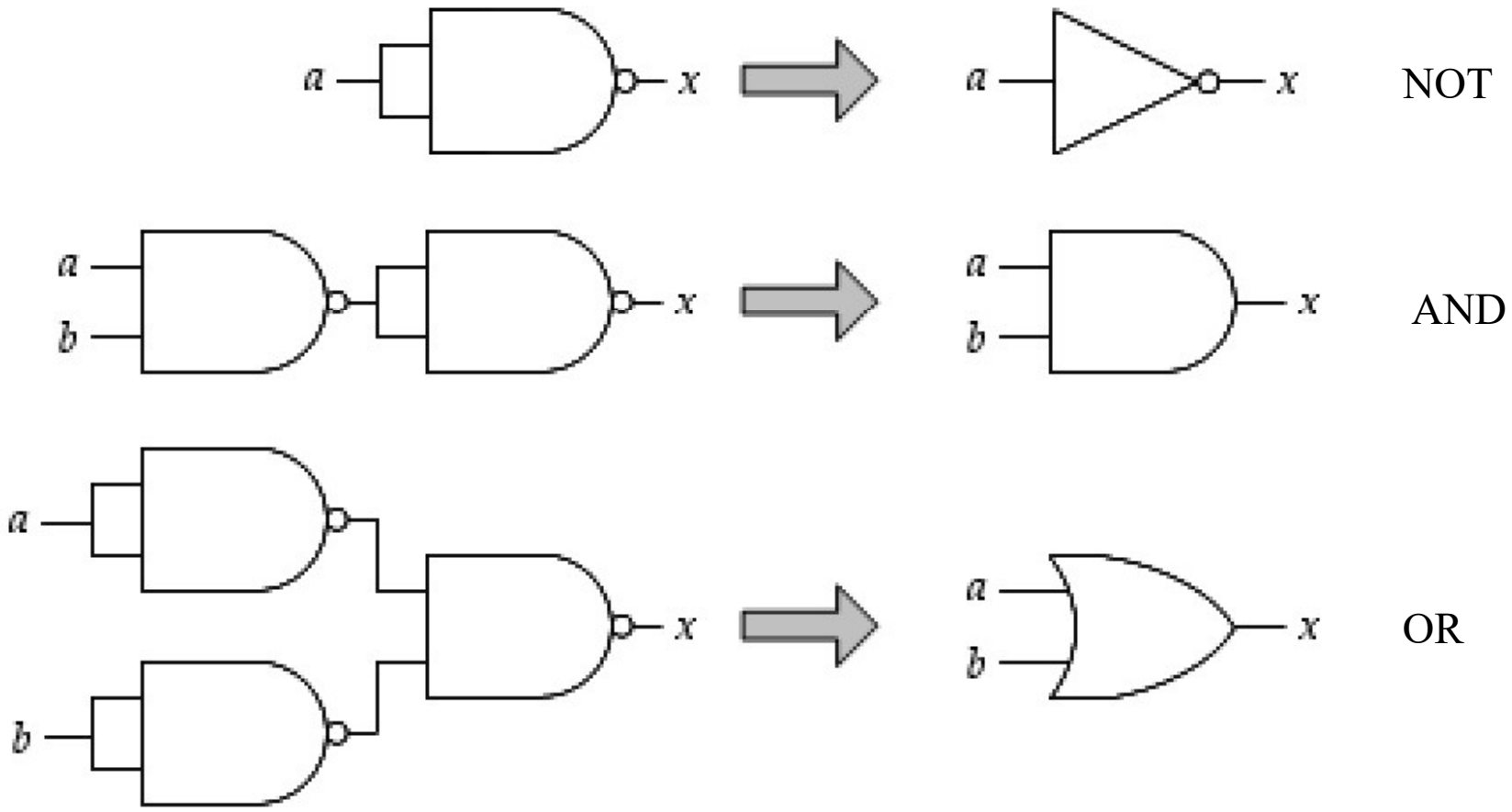
NXOR		
a	b	x
0	0	1
1	0	0
0	1	0
1	1	1

Expressão: $x = a \otimes b$

Portas lógicas

- Combinações de portas NAND podem ser usadas para simular todas as outras.
- Por este motivo, a porta NAND é considerada uma **porta universal**.
- Isso significa que qualquer circuito pode ser expresso pela combinação de portas NAND.

Portas lógicas



Exercícios: Faça as operações lógicas AND, OR e XOR para a seguinte tabela

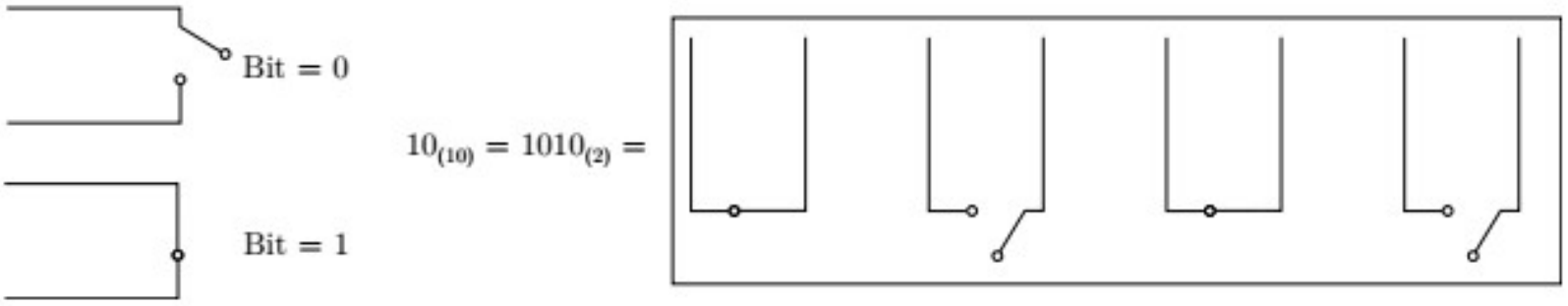
Op1	Op2	AND	OR	XOR
00011100	11110000			
11100011	10101010			
10101010	00110111			
11110000	10001000			
00001111	01010101			
11001100	11111111			
11101110	00100100			
10001000	00010001			

Computadores e Números

- Computadores modernos são, em sua enorme maioria, dispositivos digitais
  - Representam sinais como conjuntos de símbolos discretos
- Mais que isso, computadores são normalmente binários
  - Dois estados: tensão baixa ou tensão alta.
- Nas últimas aulas, vimos como números podem ser representados em base 2
  - Base numérica composta por dois símbolos (algarismos), os **bits**.

Computadores e Números (II)

- Com uma sequência ordenada de bits, podemos representar números quaisquer
  - Desde que o **número de bits** disponível seja **grande o suficiente**.



Computadores e Números (III)

- Note que por se tratar de circuitos eletrônicos, computadores usam um número fixo de bits para representar números:
  - 8 bits, 16 bits ou 32 bits.
- Se um número tem menos algarismos binários, ele e complementado com zeros à esquerda.
- Exemplos com 8 bits:
  - $200_{(10)} = 11001000_{(2)}$.
  - $100_{(10)} = 01100100_{(2)}$.
  - $50_{(10)} = 00110010_{(2)}$.
- Por outro lado, isso **limita o conjunto de números que podem ser representados**.

Computadores e Números (IV)

- Há necessidade de representação não somente de números inteiros positivos
  - O que fazer com os números **negativos**?
  - E com os números com parte **fracionária** não-nula?

- Devem ser definidos os **esquemas de representação**

Esquemas de Representação

- Maneira padronizada de codificar informações usando apenas bits
  - Valor que assume dois estados
- Esquemas:
  - Sinal e Magnitude
  - Representação em Excesso
  - Complemento a Um
  - **Complemento a Dois**
  - Representação em Ponto Fixo
  - **Representação em Ponto Flutuante**

Sinal e Magnitude

- A representação por Sinal e Magnitude é a mais intuitiva capaz de representar apenas números inteiros:
  - Mas tanto positivos, quanto negativos.

- Números são separados em duas componentes:
  - Sinal
    - ⋆ Como o sinal assume dois estados (negativo e positivo) – um bit para representá-lo
  - Magnitude (ou módulo, ou valor absoluto)
    - ⋆ Como se o número fosse positivo

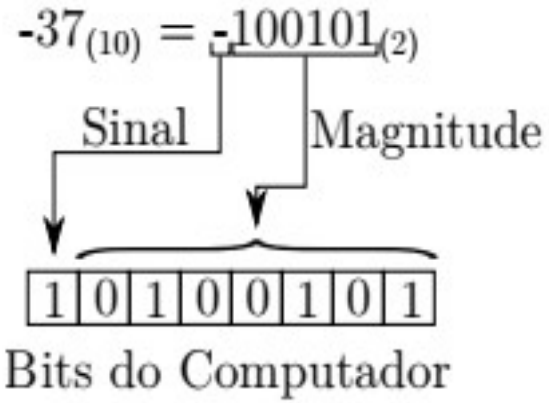
Convenções em Sinal e Magnitude

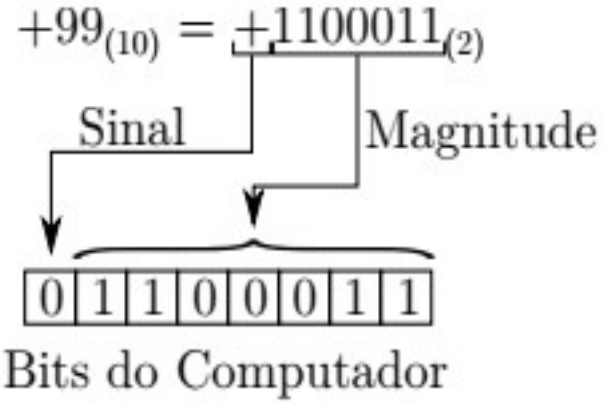
- Bit reservado para o sinal é sempre o mais significativo
  - *i.e.*, o mais à esquerda
- Em números positivos, esse bit é 0
- Em números negativos, esse bit é 1

- A magnitude é simplesmente representada em base 2 com $n - 1$ bits
  - Onde n é o número de bits usado pela máquina para representar números.

Exemplos de Representação

- Exemplos considerando 8 bits.





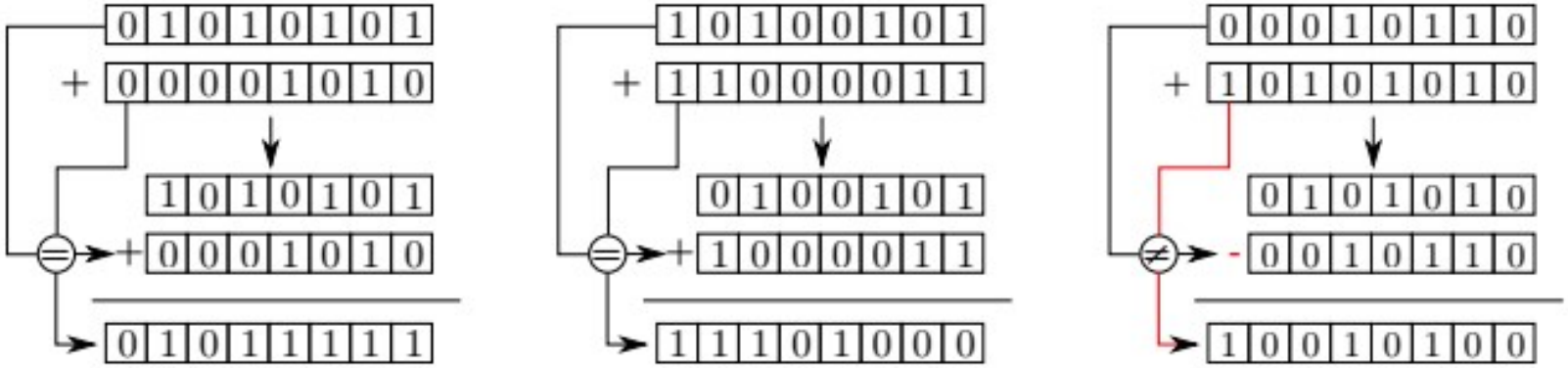
Fazendo Contas com Sinal e Magnitude

- Suponha que um computador deseje operar sobre dois números em Sinal e Magnitude.
  - Somar
  - Subtrair
  - Comparar
  - ...
- Como isso deve ser feito?
- Considere uma soma, por exemplo
  - Não podemos simplesmente somar os dois números algarismo a algarismo – um dos "algarismos" é o sinal

Fazendo Somas com Sinal e Magnitude

- Se os bits de sinal dos dois números são iguais
  - separe as magnitudes e as some
  - Bit de sinal do resultado é igual ao bit de sinal dos operandos.

- Caso contrário
  - isole as magnitudes e **subtraia** a menor da maior.
  - Bit de sinal do resultado é igual ao bit de sinal do operando de maior magnitude.



Fazendo Contas com Sinal e Magnitude

- Outras operações podem ser realizadas de forma similar
  - Para subtração, pode-se trocar o sinal do subtraendo (inverter bit de sinal) e executar mesmos passos da soma
  - Para divisão e multiplicação, opera-se apenas sobre a magnitude
    - ★ Se operandos têm mesmo sinal, resultado terá bit de sinal 0.
    - ★ Se operandos têm sinal diferente, resultado terá bit de sinal 1.

Sinal e Magnitude: Limites

- Suponha 8 bits para representar um número em Sinal e Magnitude
- Qual o maior número (maior positivo) que pode ser representado?
  - Em Sinal e Magnitude, primeiro bit 0 e todos os outros iguais a 1.
  - Para 8 bits: $0\ 1111111_{(2)} = 127_{(10)}$.
- Qual o menor número (negativo) que pode ser representado?
  - Em Sinal e Magnitude, o primeiro bit é 1 e todos os outros iguais a 1.
  - Para 8 bits: $-1111111_{(2)} = -127_{(10)}$.

Sinal e Magnitude: Limites (II)

- De forma mais genérica, com uma quantidade n qualquer de bits:
  - **Maior número:**

$$2^{(n-2)} + 2^{(n-3)} + \cdots + 2^0 = \boxed{2^{(n-1)} - 1}$$

  - **Menor número:**

$$-\left(2^{(n-2)} + 2^{(n-3)} + \cdots + 2^0\right) = \boxed{-\left(2^{(n-1)} - 1\right)}$$

- Exemplos para alguns valores de n:
  - Para $n = 4$: de -7 a 7.
  - Para $n = 16$: de -32767 a 32767.
  - Para $n = 32$: de -2147483647 a 2147483647

Sinal e Magnitude: Duplicidade do Valor 0

- Considere as seguintes sequências de bits:
  - 00000000 e 10000000.
- Assumindo que ambas são representações em Sinal e Magnitude com 8 bits, quais os valores representados?
- Vamos fazer a interpretação:
  - Primeiro número tem sinal positivo e magnitude 0.
  - Segundo número tem sinal negativo e magnitude 0.

- Conclusão: **ambos são 0**

Sinal e Magnitude: Usos

- Pela sua similaridade com a notação escrita, a representação em Sinal e Magnitude foi usada em alguns computadores antigos.
  - *e.g.*, IBM 7090 em 1959.
- Mas a duplicidade do valor 0 e a lógica "complicada" para certas operações matemáticas resultou em pouco popularidade.
  - Outras representações são mais simples.
  - **Note que a duplicidade do zero desperdiça bits.**
- Hoje, a maior importância desta representação é ser a base para a Representação em Ponto Flutuante.

Conceito de Complemento

- Uma das utilidades do conceito de complemento é a simplificação do processo de subtração:
  - Calculamos o complemento (ums subtração *fácil*)
  - Somamos com o complemento
  - Incrementamos de 1 (outra soma)
  - No final, ignoramos o algarismo mais significativo

Conceito de Complemento

- Exemplo: calcular $65_{(10)} - 37_{(10)}$
  - Complemento a 9 (*i.e.*, na base 10) de $37_{(10)}$ é $62_{(10)}$

$$99_{(10)} - 37_{(10)} = 62_{(10)}$$

  - Somando minuendo e complemento do subtraendo:

$$65_{(10)} + 62_{(10)} = 127_{(10)}$$

  - Incrementando de um e ignorando o algarismo mais significativo, chegamos à resposta:

$$28_{(10)}$$

Representação de Números Inteiros Negativos

- Idealmente, precisamos de uma representação que permita tratarmos **uniformemente** números positivos e negativos.
  - números em um representação uniforme não receberão **tratamento especial** para serem operados
  - principalmente **para somas**.
    - ⋆ Outras operações são mais complicadas

- Por exemplo, para somar em Sinal e Magnitude, precisamos **olhar para o primeiro bit**:
  - Se for 0, efetuamos a soma.
  - Se for 1, temos que efetuar uma subtração

Complemento e os Números Negativos

- Voltando ao conceito de complemento, note que ele possui uma relação com números negativos:
- Isto é, para calcular $a_1 - a_2$ em uma base b qualquer podemos usar o método do complemento
  - Calculamos o complemento a $b-1$ de a_2 e **transformamos a operação em uma soma**
    - ⋆ Embora ainda haja os detalhes do incremento e de ignorar o algarismo mais significativo do resultado

Complemento e os Números Negativos

- Para calcular $a - a$, por exemplo, somamos a com seu próprio complemento

- Calcular $63_{(10)} - 63_{(10)}$ usando o método de complemento a 9:
  - Complemento a 9 de $63_{(10)}$ é $36_{(10)}$
  - Fazendo a soma, obtemos $63_{(10)} + 36_{(10)} = 99_{(10)}$
  - Incrementando e ignorando o algarismo mais significativo, obtemos $0_{(10)}$

- De certa forma, podemos dizer que o complemento de um número é *similar* ao seu negativo.
  - Ao menos, exerce um papel **parecido**.

Representação por Complemento a Um: Ideia Básica

- Aproveitando esta proximidade dos números negativos com o complemento da magnitude, vejamos o seguinte um esquema de representação
- Na representação por complemento a 1, números negativos são representados pelo complemento a 1 da sua magnitude
  - Números positivos são representados normalmente em base 2.

- Exemplos
  - $5_{(10)}$ é representado com 5 bits como 00101
  - $-5_{(10)}$ é representado com 5 bits como 11010
  - $37_{(10)}$ é representado com 8 bits como 00100101
  - $-37_{(10)}$ é representado com 8 bits como 11011010

Representação por Complemento a Um: Detalhes

- Note que calcular o complemento a 1 de um número binário é muito simples.
  - Basta "inverter" os bits.
    - ⋆ 0 vira 1.
    - ⋆ 1 vira 0.

- Exemplos:
  - Se a tem representação 01101100, $-a$ tem representação 10010011.
  - Se a tem representação 1010, $-a$ tem representação 0101.

Representação por Complemento a Um: Unicidade

- Uma propriedade importante de um esquema de representação é a **unicidade**.
  - *i.e.*, uma dada sequência de bits deve representar **um único valor**
  - Exemplo: em Sinal e Magnitude, a sequência 01101 representa **apenas** o número $13_{(10)}$.

- No entanto, vejamos representação por Complemento a Um
  - Vamos analisar alguns exemplos, considerando 5 bits:
    $\rightarrow +19_{(10)}$ tem representação 10011
    $\rightarrow -19_{(10)}$ tem representação 01100
    $\rightarrow +12_{(10)}$ tem representação 01100
    { Mesmo Valor
  - Conclusão: encontramos uma sequência de bits (01100) que representa **dois valores simultaneamente**.

Representação por Complemento a Um: Unicidade (II)

- Esse problema pode ser evitado:
  - Determinação dos **limites** para o uso da representação

- O problema ocorreu porque tentamos representar um número **positivo** que necessita do **bit mais significativo igual a 1**.
  - seu complemento (negativo) tenha o primeiro bit igual a 0
  - Mas esta representação já é utilizada por outro número positivo

- Então, seja a regra adicional:
  - O bit mais significativo determina o sinal do número:
    - **0, para positivos**
    - **1 para negativos**

  - Mesma lógica da representação por sinal e magnitude
  - problema da falta de unicidade resolvido

Representação por Complemento a Um: Realizando Contas

- A grande motivação para a exploração de outros esquemas de representação é a necessidade de simplificar operações aritméticas

- Até que ponto o Complemento a Um é bem sucedido?

Complemento a Um: Realizando Contas

- Considere o exemplo da soma:
  - Soma de dois números positivos – trivial
    - Suas representações são simplesmente as conversões para base 2
    - Exemplo com 5 bits: $01010 + 00101 = 01111$.
  - Soma de um número positivo com outro negativo – mais complicada
    - Se tentarmos a soma direta como para os positivos, teremos problemas:
    - Exemplo com 5 bits: $01010 + 11010 = 100100$.
    - O resultado esperado seria 00101.
  - Somas entre dois números negativos– também problemática

Complemento a Um: Realizando Contas

- Algoritmo simples de soma que funciona sempre:

---

1. Seja n o número de bits usados para a representação
2. Some os números normalmente, como se fossem positivos
3. Se a soma resultar no $n+1$-*ésimo* bit igual a 1
   - ⋆ ignore-o e incremente o resultado em uma unidade

- Exemplos com 5 bits (números representados em Complemento a Um):

$$\begin{array}{r} 01010 \\ +\ 00101 \\ \hline 01111 \end{array} \qquad \begin{array}{r} 00101 \\ +\ 10101 \\ \hline 11010 \end{array} \qquad \begin{array}{r} 01010 \\ +\ 11010 \\ \hline \boxed{1}00100 \\ +\ \cdots\cdots\!\!\rightarrow 1 \\ \hline 00101 \end{array} \qquad \begin{array}{r} 10101 \\ +\ 11010 \\ \hline \boxed{1}01111 \\ +\ \cdots\cdots\!\!\rightarrow 1 \\ \hline 10000 \end{array}$$

Complemento a Um: Realizando Contas

- Se a soma é fácil, a subtração também é
- Basta lembrar que

$$a_1 - a_2 = a_1 + (-a_2)$$

- Trocar o sinal de um número é trivial:
  - Basta inverter todos os bits

- Algoritmo para subtração:
  1. Inverta os bits do subtraendo.
  2. Aplique o algoritmo de soma entre o resultado e o minuendo

---

Complemento a Um: Realizando Contas

- Exemplos com 5 bits (números representados em Complemento a Um):

$$\begin{array}{rr} & 01010 \\ - & 00101 \\ \hline & 01010 \\ + & 11010 \\ \hline & \boxed{1}00100 \\ + & \dashrightarrow 1 \\ \hline & 00101 \end{array} \qquad \begin{array}{rr} & 01010 \\ - & 11010 \\ \hline & 01010 \\ + & 00101 \\ \hline & 01111 \end{array}$$

Complemento a Um: Problemas

- O principal problema do Complemento a Um é o mesmo da representação por Sinal e Magnitude.
  - Existem duas representações para o 0.
- Podemos verificar isso com um exemplo:
  - Em uma representação com 5 bits, qual valor é representado pela sequência 11111?
    - ★ O valor deve ser negativo, já que o primeiro bit é 1.
    - ★ Para saber sua magnitude, invetemos todos os bits, obtendo 00000.
- Embora o exemplo tenha sido dado com 5 bits, o mesmo ocorre com qualquer número de bits
  - A sequência $11\ldots1$ é às vezes chamada de **zero negativo**.
  - Ocorre, por exemplo, quando somamos dois números de mesmo módulo, mas sinais opostos.
    - ★ *e.g.*, com 5 bits, $01010 + 10101 = 11111$.

Complemento a Um: Uso

- A existência de duas representações para o 0 é inconveniente
- Além disso, reduz o intervalo de números que podem ser representados com um dado número de bits
- Por conta disso, o Complemento a Um não é muito utilizado hoje

- Mas é a base para um esquema de representação melhor
  - **Complemento a Dois**