DS011

Números Binários Negativos

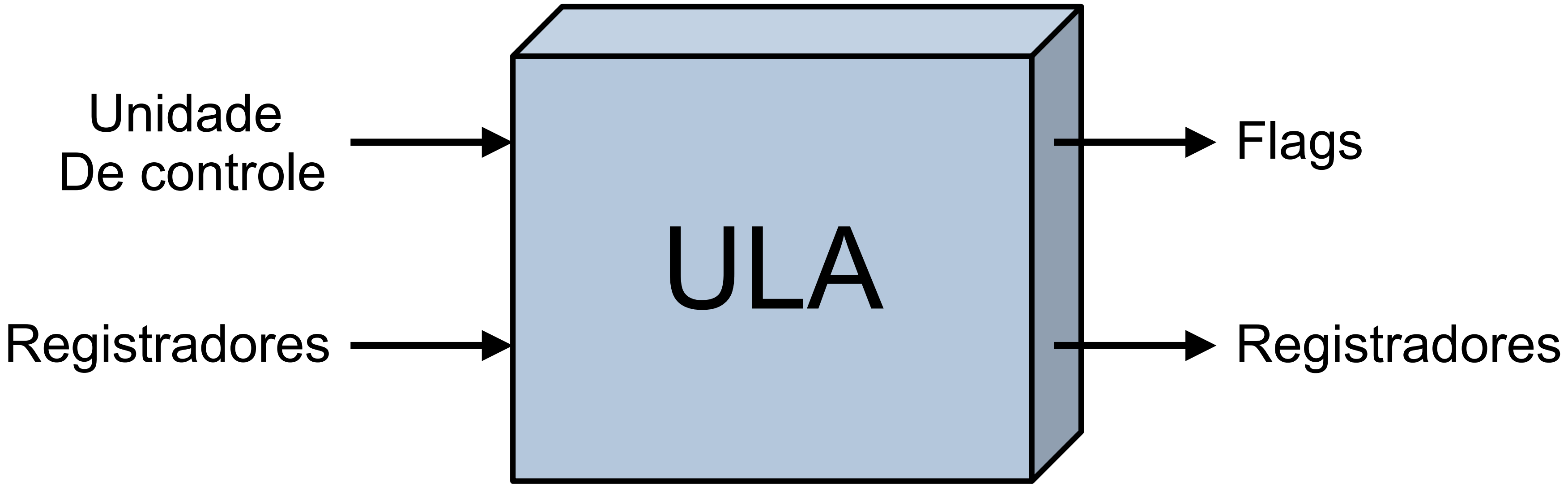
Prof. Clausius Duque Reis
clausius.reis@ufpr.br

Unidade de lógica e Aritmética (ULA)

- Faz os cálculos
- Tudo no computador existe para atender essa unidade
- Trata de números inteiros
- Pode tratar de números de ponto flutuante (Números reais), que iremos ver nos próximos conteúdos

Unidade de lógica e Aritmética (ULA)

- **Entradas e saídas da ULA (Ou ALU em inglês)**



Representação De inteiros

Representação de Inteiros

- **Computadores só tem 0 e 1 para representar tudo**
  - Números são armazenados em binário
    - Ex: $41_{10} = 00101001_2$
  - Sem sinal de positivo ou negativo
  - Sem ponto

- **Representação de números inteiros será por:**
  - Sinal-magnitude
  - Complemento a dois

Sinal Magnitude

- **Bit mais à esquerda é o bit de sinal**
  - 0 significa positivo
  - 1 significa negativo

- **Ex: $+18_{10} = 00010010_2$**
  $-18_{10} = 10010010_2$

- **Problemas com essa abordagem**
  - Precisamos considerar sinal e magnitude na aritmética
  - Temos duas representações de zero (+0 e -0)

Complemento a dois

- Números possuem intervalos diferentes entre positivos e negativos, sendo que os positivos possuem 1 a menos.

$$-2^{n-1} \text{ até } 2^{n-1}-1$$

- Uma única representação do zero (0)

- Negação:
  - Acompanhe o complemento boleano de cada bit do número positivo correspondente, depois some 1 ao padrão de bits resultante visto como um **inteiro sem sinal**

- Expansão de bits é feita adicionando bits à esquerda com o valor do bit de sinal original

Complemento a dois

- Overflow:
  - Se dois números com o mesmo sinal (positivo ou negativo) são somados, então o estouro ocorre se e somante se o resultado tem sinal oposto

- Regra da subtração:
  - Para **subtrair B de A**, pegue o complemento a 2 de **B** e some-o a **A**

Complemento a dois

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Complemento a dois

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Complemento a dois

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Complemento a dois

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Complemento a dois

- Benefícios do complemento a dois
  - Uma (única) representação de zero
  - Atirmética funciona com facilidade
    - Veremos mais adiante
  - Negação é muito fácil
    - +3 = 00000011
    - Complemento boleano gera 11111100
    - Some 1 ao LSB 11111101
    - -3 = 11111101

Complemento a dois

- Negação especial – Caso 1
  - 0 = 00000000
  - Complemento boleano gera `11111111`
  - Some 1 ao LSB `+00000001`
  - Resultado `100000000`
  - Estouro é ignorado, portanto:
    - 0 = 0 → Correto!

Complemento a dois

- Negação especial – Caso 2
  - 128 = 10000000
  - Complemento boleano gera `01111111`
  - Some 1 ao LSB `+00000001`
  - Resultado `100000000`
  - Portanto
    - -(-128) = -128 → Errado!
    - Monitore o MSB (bit de sinal)
      - Ele deve mudar durante a negação, se não mudou está errado

Intervalo de números

- Complemento a 2 com 8 bits:
  - `+127 = 01111111 =` 2^7-1
  - `-128 = 10000000 =` -2^7

- Complemento a 2 com 16 bits:
  - `+32767 = 01111111 11111111 =` $2^{15}-1$
  - `-32768 = 10000000 00000000 =` $2^{15}-1$

Intervalo de números

- Complemento a 2 com 8 bits:
  - `+127 = 01111111 =` 2^7-1
  - `-128 = 10000000 =` -2^7

- Complemento a 2 com 16 bits:
  - `+32767 = 01111111 11111111 =` $2^{15}-1$
  - `-32768 = 10000000 00000000 =` $2^{15}-1$

- Conversão entre tamanhos:
  - `+18          00010010  (8 bits)`
  - `+18 00000000 00010010  (16 bits)`
  - `-18          11101110  (8 bits – complemento à 2)`
  - `-18 11111111 11101110  (16 bits – complemento à 2)`

Intervalo de números

- Complemento a 2 com 8 bits:
  - +127 = 01111111 = 2^7-1
  - -128 = 10000000 = -2^7

- Complemento a 2 com 16 bits:
  - +32767 = 01111111 11111111 = 2^{15}-1
  - -32768 = 10000000 00000000 = 2^{15}-1

- Conversão entre tamanhos:
  - +18 00010010 (8 bits)
  - +18 00000000 00010010 (16 bits)

  Números positivos

  - -18 11101110 (8 bits – complemento à 2)
  - -18 11111111 11101110 (16 bits – complemento à 2)

Intervalo de números

- Complemento a 2 com 8 bits:
  - `+127 = 01111111 =` 2^7-1
  - `-128 = 10000000 =` -2^7

- Complemento a 2 com 16 bits:
  - `+32767 = 01111111 11111111 =` $2^{15}-1$
  - `-32768 = 10000000 00000000 =` $2^{15}-1$

- Conversão entre tamanhos:
  - `+18          00010010  (8 bits)`
  - `+18 00000000 00010010  (16 bits)`

  Números positivos

  - `-18          11101110  (8 bits)`
  - `-18 11111111 11101110  (16 bits)`

  Números negativos

Adição e Subtração

- Adição binária normal

- Monitore o estouro do bit de sinal

- Pegue o complemento a dois do subtraendo e some ao minuendo.

$$A–B = A+(-B)$$

- Assim, só precisamos implementar no hardware do computador os circuitos de adição e complemento a dois

Adição

```
 1001
+0101
```

(a) (-7)+(+5)=-2

Adição

```
 1001
+0101
 ----
 1110
(a) (-7)+(+5)=-2
```

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 (b) (-4)+(+4)=0

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 (c) (+3)+(+4)=+7	

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 0111 (c) (+3)+(+4)=+7	

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 0111 (c) (+3)+(+4)=+7	1100 +1111 (d) (-4)+(-1)=-5

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 0111 (c) (+3)+(+4)=+7	1100 +1111 11011 (overflow) (d) (-4)+(-1)=-5

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 0111 (c) (+3)+(+4)=+7	1100 +1111 11011 (overflow) (d) (-4)+(-1)=-5
0101 +0100 (e) (+5)+(+4)=+9	

Adição

1001 +0101 1110 (a) (-7)+(+5)=-2	1100 +0100 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 0111 (c) (+3)+(+4)=+7	1100 +1111 11011 (overflow) (d) (-4)+(-1)=-5
0101 +0100 (overflow) 1001 (e) (+5)+(+4)=+9	

Adição

1001 +0101 ——— 1110 (a) (-7)+(+5)=-2	1100 +0100 ——— 10000 (overflow) (b) (-4)+(+4)=0
0011 +0100 ——— 0111 (c) (+3)+(+4)=+7	1100 +1111 ——— 11011 (overflow) (d) (-4)+(-1)=-5
0101 +0100 ——— (overflow) 1001 (e) (+5)+(+4)=+9	1001 +1010 ——— (f) (-7)+(-6)=-13

Adição

```
  1001
+0101
 -----
  1110
(a) (-7)+(+5)=-2
```

```
  1100
+0100
 -----
 10000        (overflow)
(b) (-4)+(+4)=0
```

```
  0011
+0100
 -----
  0111
(c) (+3)+(+4)=+7
```

```
  1100
+1111
 -----
 11011        (overflow)
(d) (-4)+(-1)=-5
```

```
              0101
            +0100
             -----
(overflow)    1001
(e) (+5)+(+4)=+9
```

```
  1001
+1010
 -----
 10011        (overflow)
(f) (-7)+(-6)=-13
```

Subtração

```
(a)     0010
       +1001
       _____

 M =   2 = 0010
 S =   7 = 0111
-S =  -7 = 1001
(+2) + (-7)=-5
```

Subtração

```
(a)     0010
       +1001
       -----
        1011
 M =  2 = 0010
 S =  7 = 0111
-S = -7 = 1001
(+2) + (-7)=-5
```

Subtração

(a) 0010 +1001 ——— 1011 M = 2 = 0010 S = 7 = 0111 -S = -7 = 1001 (+2) + (-7)=-5	(b) 1011 +1110 ——— M = -5 = 1011 S = 2 = 0010 -S = -2 = 1110 (-5) + (-2)=-7

Subtração

(a)	(b)
0010	1011
+1001	+1110
1011	(overflow) 11001
M = 2 = 0010	M = -5 = 1011
S = 7 = 0111	S = 2 = 0010
-S = -7 = 1001	-S = -2 = 1110
(+2) + (-7)=-5	(-5) + (-2)=-7

Subtração

(a)	(b)	(c)
0010 +1001 1011	1011 +1110 (overflow) 11001	0111 +0111
M = 2 = 0010 S = 7 = 0111 -S = -7 = 1001 (+2) + (-7)=-5	M = -5 = 1011 S = 2 = 0010 -S = -2 = 1110 (-5) + (-2)=-7	M = 7 = 0111 S = -7 = 1001 -S = 7 = 0111 (+7) + (+7)=+14

Subtração

(a) 0010	(b) 1011	(c) 0111
+1001	+1110	+0111
1011	(overflow) 11001	1110
M = 2 = 0010	M = -5 = 1011	M = 7 = 0111
S = 7 = 0111	S = 2 = 0010	S = -7 = 1001
-S = -7 = 1001	-S = -2 = 1110	-S = 7 = 0111
(+2) + (-7)=-5	(-5) + (-2)=-7	(+7) + (+7)=+14 (overflow)

Subtração

```
(a)     0010
       +1001
       -----
        1011
 M =   2 = 0010
 S =   7 = 0111
-S =  -7 = 1001
(+2) + (-7)=-5
```

```
(b)     1011
       +1110
       -----
(overflow) 11001
 M = -5 = 1011
 S =  2 = 0010
-S = -2 = 1110
(-5) + (-2)=-7
```

```
(c)     0111
       +0111
       -----
        1110
 M =  7 = 0111
 S = -7 = 1001
-S =  7 = 0111
(+7) + (+7)=+14 (overflow)
```

```
(d)     0101
       +1110
       -----

 M =   5 = 0101
 S =   2 = 0010
-S =  -2 = 1110
(+5) + (-2)=+3
```

Subtração

(a)	(b)	(c)
0010 +1001 1011 M = 2 = 0010 S = 7 = 0111 -S = -7 = 1001 (+2) + (-7)=-5	1011 +1110 (overflow) 11001 M = -5 = 1011 S = 2 = 0010 -S = -2 = 1110 (-5) + (-2)=-7	0111 +0111 1110 M = 7 = 0111 S = -7 = 1001 -S = 7 = 0111 (+7) + (+7)=+14 (overflow)

(d)
0101 +1110 10011 (overflow) M = 5 = 0101 S = 2 = 0010 -S = -2 = 1110 (+5) + (-2)=+3

Subtração

(a) 0010 +1001 ――― 1011 M = 2 = 0010 S = 7 = 0111 -S = -7 = 1001 (+2) + (-7)=-5	(b) 1011 +1110 ――― (overflow) 11001 M = -5 = 1011 S = 2 = 0010 -S = -2 = 1110 (-5) + (-2)=-7	(c) 0111 +0111 ――― 1110 M = 7 = 0111 S = -7 = 1001 -S = 7 = 0111 (+7) + (+7)=+14 (overflow)
(d) 0101 +1110 ――― 10011 (overflow) M = 5 = 0101 S = 2 = 0010 -S = -2 = 1110 (+5) + (-2)=+3	(e) 0101 +0010 ――― M = 5 = 0101 S = -2 = 1110 -S = 2 = 0010 (+5) + (+2)=+7	

Subtração

```
(a)     0010
       +1001
       -----
        1011
 M =   2 = 0010
 S =   7 = 0111
-S =  -7 = 1001
(+2) + (-7)=-5
```

```
(b)     1011
       +1110
       -----
(overflow) 11001
 M =  -5 = 1011
 S =   2 = 0010
-S =  -2 = 1110
(-5) + (-2)=-7
```

```
(c)     0111
       +0111
       -----
        1110
 M =   7 = 0111
 S =  -7 = 1001
-S =   7 = 0111
(+7) + (+7)=+14 (overflow)
```

```
(d)     0101
       +1110
       -----
       10011 (overflow)
 M =   5 = 0101
 S =   2 = 0010
-S =  -2 = 1110
(+5) + (-2)=+3
```

```
(e)     0101
       +0010
       -----
        0111
 M =   5 = 0101
 S =  -2 = 1110
-S =   2 = 0010
(+5) + (+2)=+7
```

Subtração

```
(a)     0010
       +1001
       -----
        1011
 M =  2 = 0010
 S =  7 = 0111
-S = -7 = 1001
(+2) + (-7)=-5
```

```
(b)     1011
       +1110
       -----
(overflow) 11001
 M = -5 = 1011
 S =  2 = 0010
-S = -2 = 1110
(-5) + (-2)=-7
```

```
(c)     0111
       +0111
       -----
        1110
 M =  7 = 0111
 S = -7 = 1001
-S =  7 = 0111
(+7) + (+7)=+14 (overflow)
```

```
(d)     0101
       +1110
       -----
       10011 (overflow)
 M =  5 = 0101
 S =  2 = 0010
-S = -2 = 1110
(+5) + (-2)=+3
```

```
(e)     0101
       +0010
       -----
        0111
 M =  5 = 0101
 S = -2 = 1110
-S =  2 = 0010
(+5) + (+2)=+7
```

```
(f)     1010
       +1100
       -----

 M = -6 = 1010
 S =  4 = 0100
-S = -4 = 1100
(-6) + (-4)=-10
```

Subtração

```
(a)    0010
      +1001
      -----
       1011
 M =   2 = 0010
 S =   7 = 0111
-S =  -7 = 1001
(+2) + (-7)=-5
```

```
(b)    1011
      +1110
      -----
(overflow) 11001
 M = -5 = 1011
 S =  2 = 0010
-S = -2 = 1110
(-5) + (-2)=-7
```

```
(c)    0111
      +0111
      -----
       1110
 M =  7 = 0111
 S = -7 = 1001
-S =  7 = 0111
(+7) + (+7)=+14 (overflow)
```

```
(d)    0101
      +1110
      -----
      10011(overflow)
 M =   5 = 0101
 S =   2 = 0010
-S =  -2 = 1110
(+5) + (-2)=+3
```

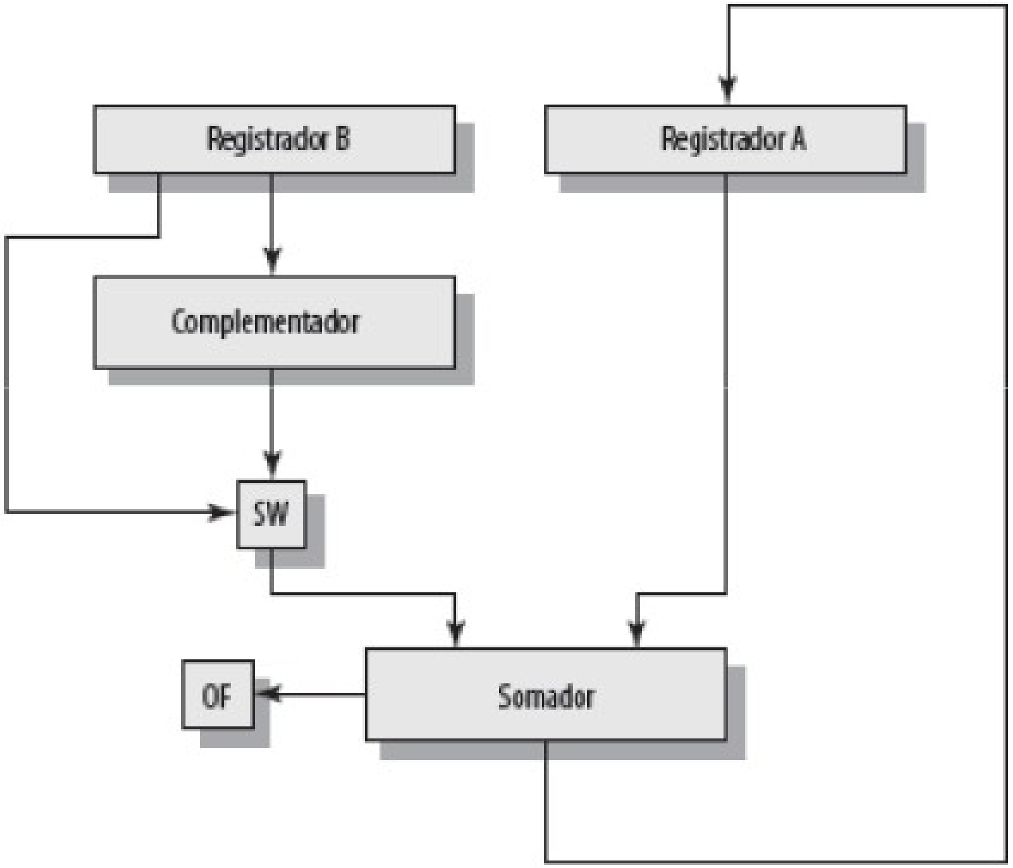
```
(e)    0101
      +0010
      -----
       0111
 M =  5 = 0101
 S = -2 = 1110
-S =  2 = 0010
(+5) + (+2)=+7
```

```
(f)    1010
      +1100
      -----
      10110
 M = -6 = 1010
 S =  4 = 0100
-S = -4 = 1100
(-6) + (-4)=-10 (overflow)
```


Prof. Clausius Duque Reis

Adição e Subtração

Hardware para adição e subtração



OF = bit de *overflow* (do inglês *overflow bit*)
SW = seletor – multiplexador (seleciona adição ou subtração)

Multiplicação

- Complexa
- Calcule produto parcial para cada dígito
- Cuidado com o valor da casa (coluna)
- Some produtos parciais

Multiplicação

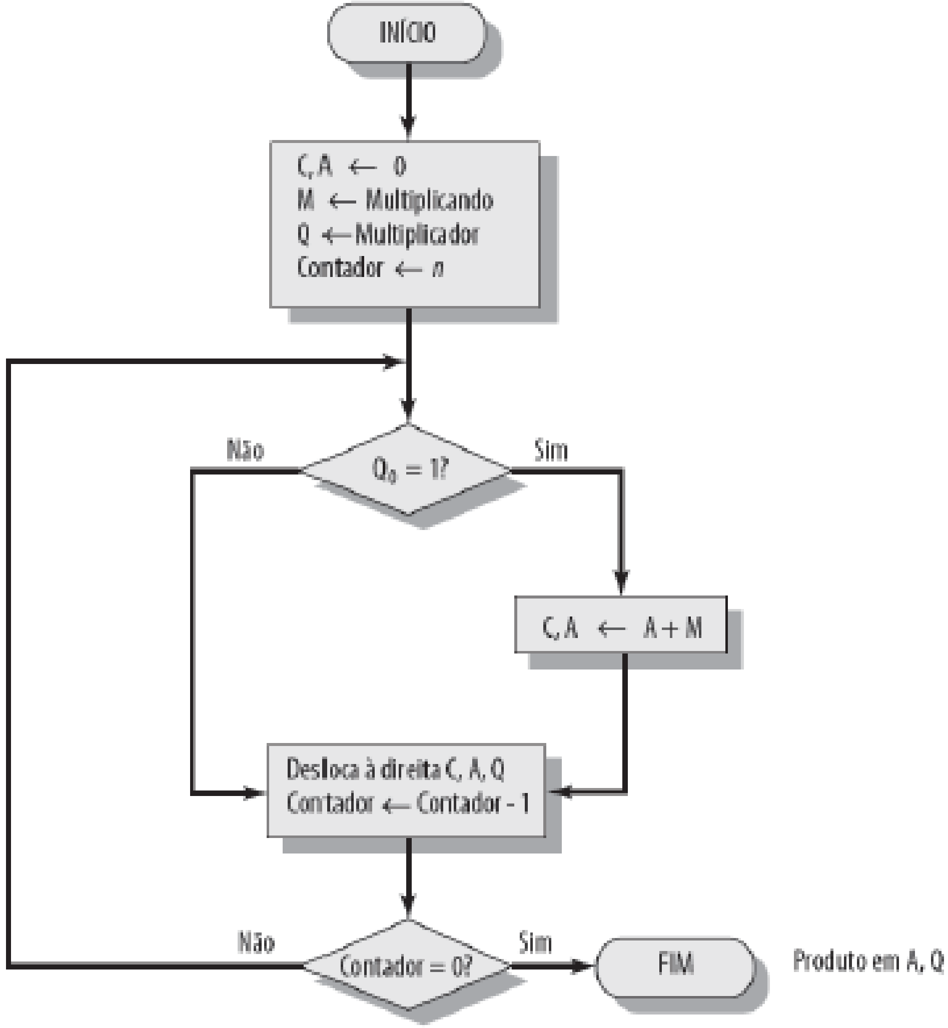
- **Multiplicação**

```
    1011     Multiplicando (11)
   x1101     Multiplicador (13)
---------
    1011
   0000
  1011       Produtos parciais
 1011
---------
10001111     Produto (143)
```

- **Nota: precisa de resultado com tamanho duplo**

Multiplicação

- **Fluxograma para Multiplicação**



Multiplicação

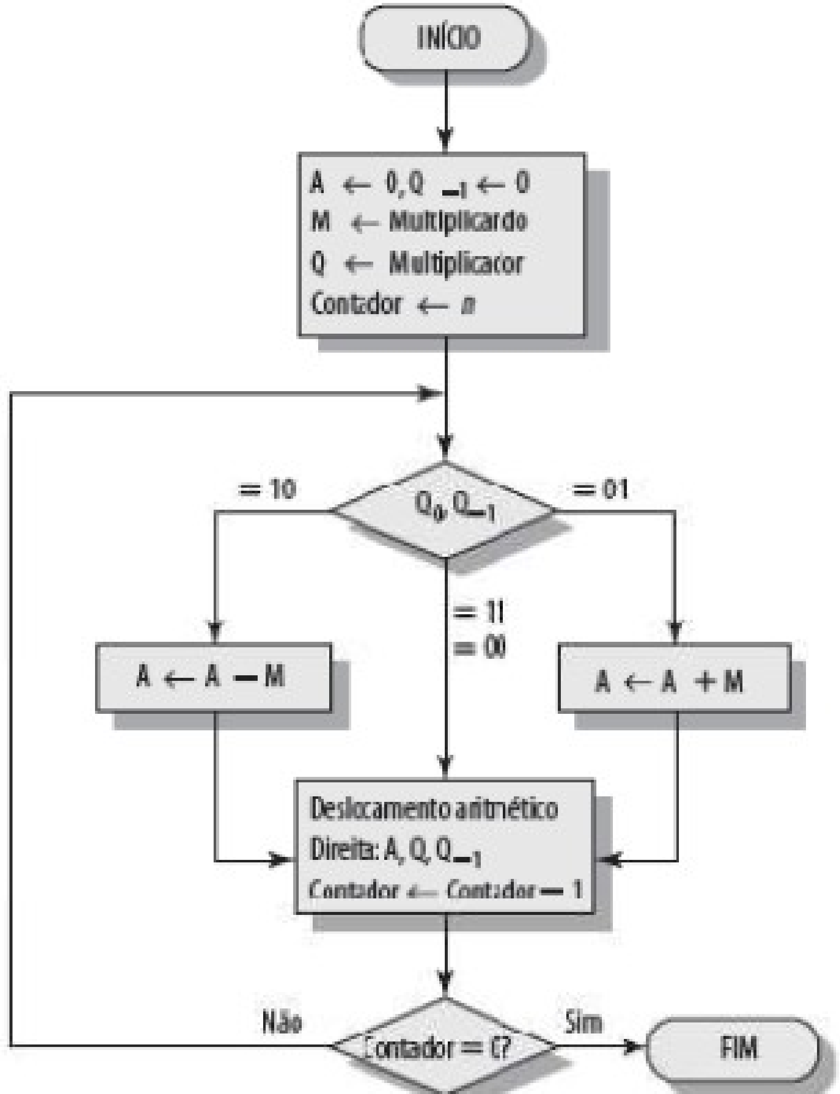
- **Multiplicação de 1101 por 1011**

C	A	Q	M		
0	0000	1101	1011	Valores iniciais	
0	1011	1101	1011	Adição	Primeiro ciclo
0	0101	1110	1011	Desl.	
0	0010	1111	1011	Desl.	Segundo ciclo
0	1101	1111	1011	Adição	Terceiro ciclo
0	0110	1111	1011	Desl.	
1	0001	1111	1011	Adição	Quarto ciclo
0	1000	1111	1011	Desl.	

Multiplicação

- **Multiplicando números negativos**
  - Isso não funciona…
  - Solução 1:
    - Converta para positivo, se for preciso.
    - Multiplique como antes.
    - Se sinais diferentes, negue a resposta.
  - Solução 2:
    - Algoritmo de Booth.

Multiplicação – Algoritmo de Booth



Multiplicação – Algoritmo de Booth

- **Exemplo do algoritmo de Booth (7x3)**

A	Q	Q_{-1}	M		
0000	0011	0	0111	Valores iniciais	
1001	0011	0	0111	A ← A − M	Primeiro ciclo
1100	1001	1	0111	Deslocamento	
1110	0100	1	0111	Deslocamento	Segundo ciclo
0101	0100	1	0111	A ← A + M	Terceiro ciclo
0010	1010	0	0111	Deslocamento	
0001	0101	0	0111	Deslocamento	Quarto ciclo

Nota: É usado deslocamento aritmético para preservar o sinal

Multiplicação – Algoritmo de Booth

- Exemplo do algoritmo de Booth (7x3)

A	Q	Q_{-1}	M		
0000	$\boxed{0011}$	0	$\boxed{0111}$	Valores iniciais	
1001	0011	0	0111	A ← A − M	Primeiro ciclo
1100	1001	1	0111	Deslocamento	
1110	0100	1	0111	Deslocamento	Segundo ciclo
0101	0100	1	0111	A ← A + M	Terceiro ciclo
0010	1010	0	0111	Deslocamento	
0001	0101	0	0111	Deslocamento	Quarto ciclo

Nota: É usado deslocamento aritmético para preservar o sinal

Multiplicação – Algoritmo de Booth

- Exemplo do algoritmo de Booth (7x3)

A	Q	Q_{-1}	M		
0000	0011	0	0111	Valores iniciais	
1001	0011	0	0111	A ← A − M	Primeiro ciclo
1100	1001	1	0111	Deslocamento	
1110	0100	1	0111	Deslocamento	Segundo ciclo
0101	0100	1	0111	A ← A + M	Terceiro ciclo
0010	1010	0	0111	Deslocamento	
0001	0101	0	0111	Deslocamento	Quarto ciclo

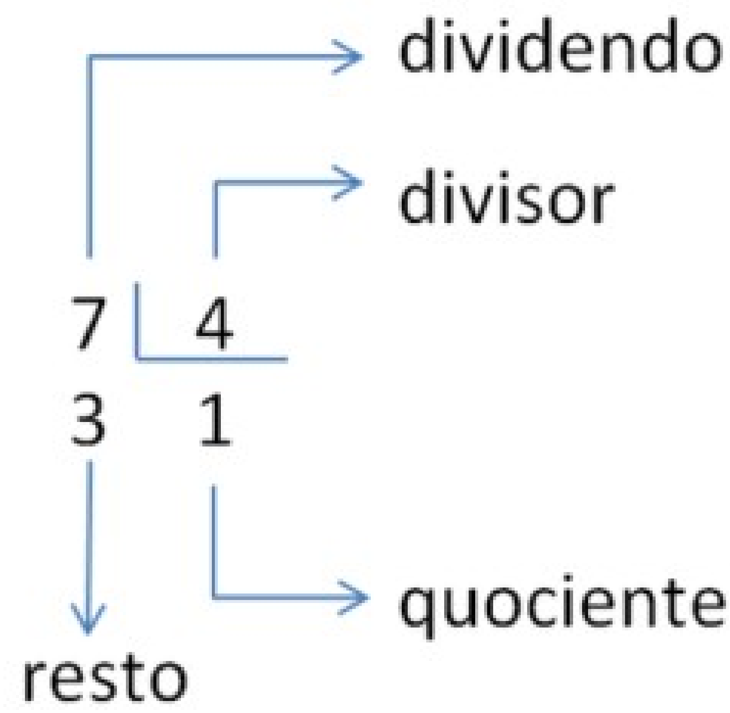
Nota: É usado deslocamento aritmético para preservar o sinal

Divisão

- Mais complexa que a multiplicação
- Números negativos são crueis!!!
- Baseada na divisão longa

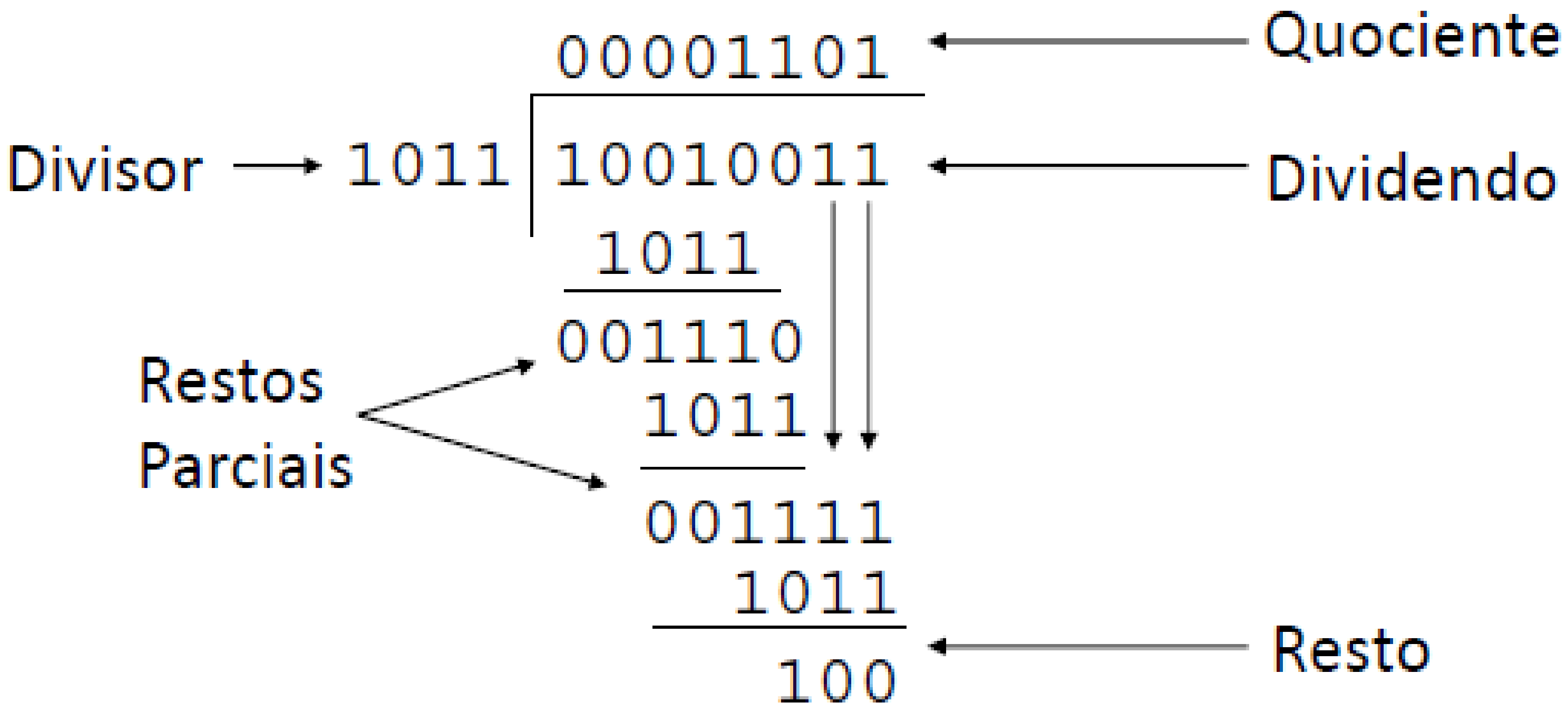
Divisão

- **Nomenclatura da divisão**



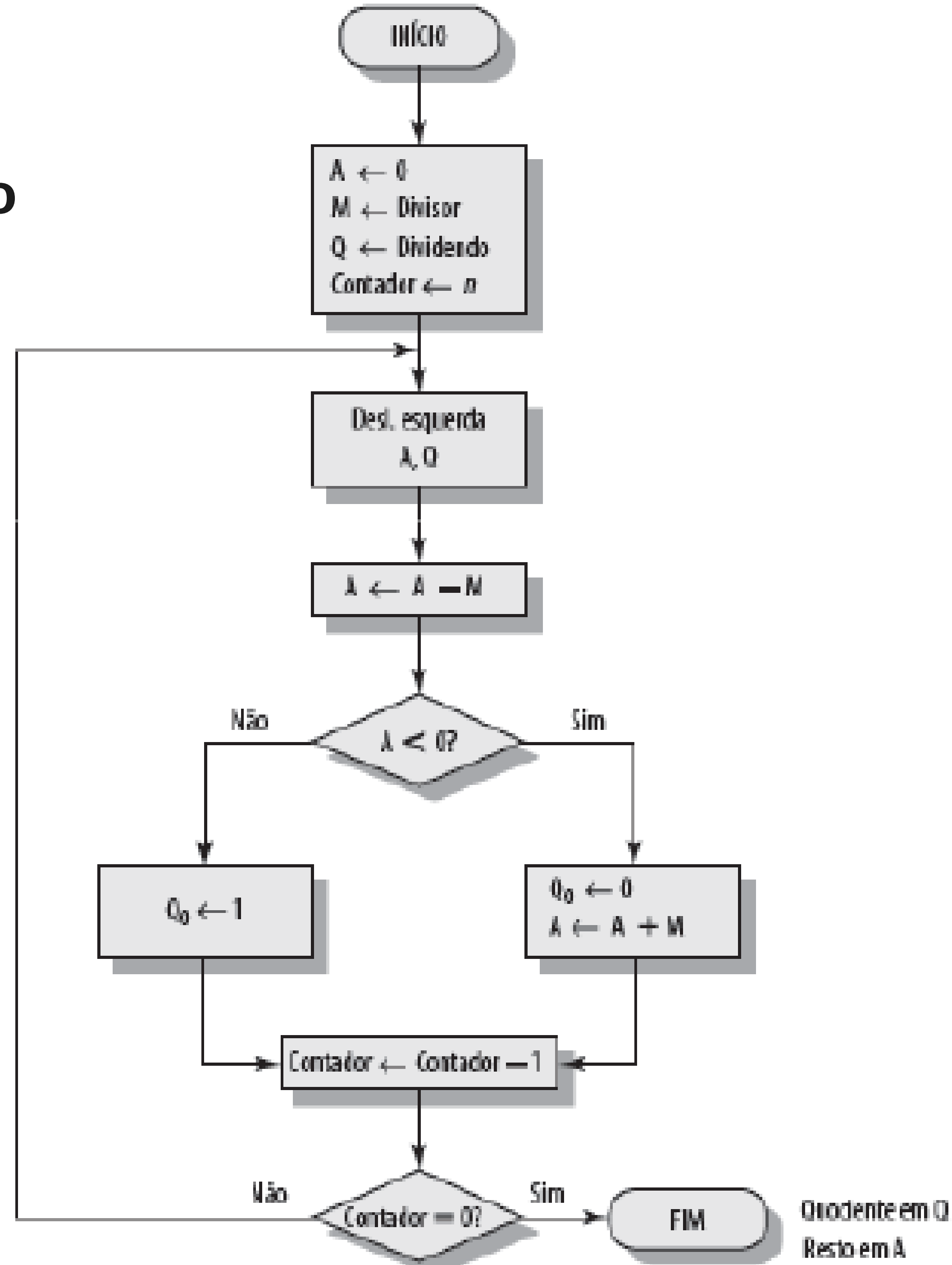
Divisão

- **Divisão de inteiros binários sem sinal**



Divisão

- **Fluxograma para divisão binária sem sinal**



Divisão

Example: Divide 15 (1111) by 4 (0100)

A	Q	M	$\overline{M}+1$	Count	Remarks
00000	1111	00100	11100	4	Initialization
00001 11101 00001	111□ 111□ 1110	- - -	- - -	- - 3	Shift Left A, Q Sub ($A \leftarrow A - M$) $Q_0 \leftarrow 0$, Add ($A \leftarrow A + M$)
00011 11111 00011	110□ 110□ 1100	- - -	- - -	- - 2	Shift Left A, Q Sub ($A \leftarrow A - M$) $Q_0 \leftarrow 0$, Add ($A \leftarrow A + M$)
00111 00011 00011	100□ 100□ 1001	- - -	- - -	- - 1	Shift Left A, Q Sub ($A \leftarrow A - M$) Set $Q_0 \leftarrow 1$
00111 00011 00011	001□ 001□ 0011	- - -	- - -	- - 0	Shift Left A, Q Sub ($A \leftarrow A - M$) Set $Q_0 \leftarrow 1$

Quotient in Q = 0011 = 3

Remainder in A = 00011 = 3