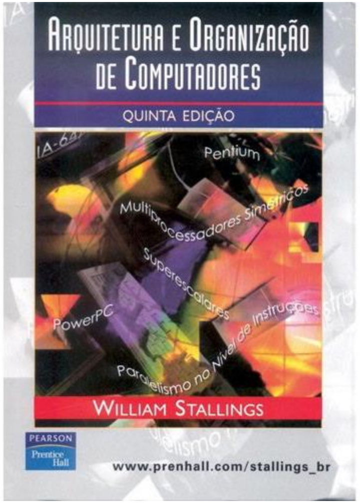
Aula 12 – Aritmética do Computador

Prof. João Fernando Mari

joaof.mari@ufv.br

Referências

- STALLINGS, W. **Arquitetura e Organização de Computadores**, 8. Ed., Pearson, 2010.
  - **Capitulo 9**

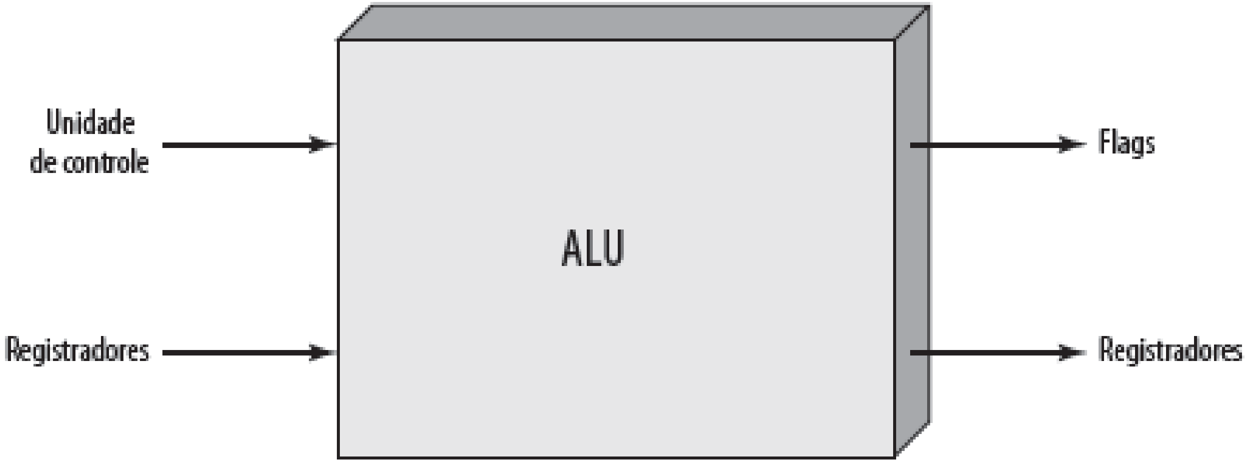


Roteiro

- Unidade Aritmética e Lógica
- Representação de inteiros
  - Sinal-magnitude
  - Complemento de dois
    - Negação especial – caso 1
    - Negação especial – caso 2
    - Intervalo de números
    - Conversão entre tamanhos
- Adição e subtração
  - Hardware para adição e subtração
- Multiplicação
  - Exemplo de multiplicação
  - Multiplicação binária sem sinal
  - Execução do exemplo
  - Fluxograma - Multiplicação Binária Sem Sinal
  - Multiplicando números negativos
  - Algoritmo de Booth
  - Exemplo do algoritmo de Booth
- Divisão
  - Divisão de inteiros binários sem sinal
  - Fluxograma para divisão binária sem sinal
  - Tratando números negativos
  - EXEMPLO: 7/3; (-7)/3; 7/(-3) e (-7)/(-3)

Unidade Aritmética e Lógica

- Responsável por realizar os cálculos.
- Tudo mais no computador existe para atender a essa unidade.
- Trata números inteiros.
- Pode tratar números de ponto flutuante (reais).



Representação de inteiros

- Somente 0's e 1's para representar tudo.
  - Números positivos armazenados em binário.
    - Ex: 41 = 00101001
  - Sem sinal de menos.
  - Sem ponto.

- Representações de números inteiros
  - Sinal-magnitude.
  - Complemento a dois.

Sinal-magnitude

- Bit mais à esquerda é bit de sinal.
  - 0 significa positivo.
  - 1 significa negativo.
  - +18 = **0**0010010.
  - -18 = **1**0010010.

- Problemas:
  - Precisa considerar sinal e magnitude na aritmética.
  - Possui duas representações de zero (+0 e -0).
    - +0 = **0**0000000
    - -1 = **1**0000000

Complemento de dois

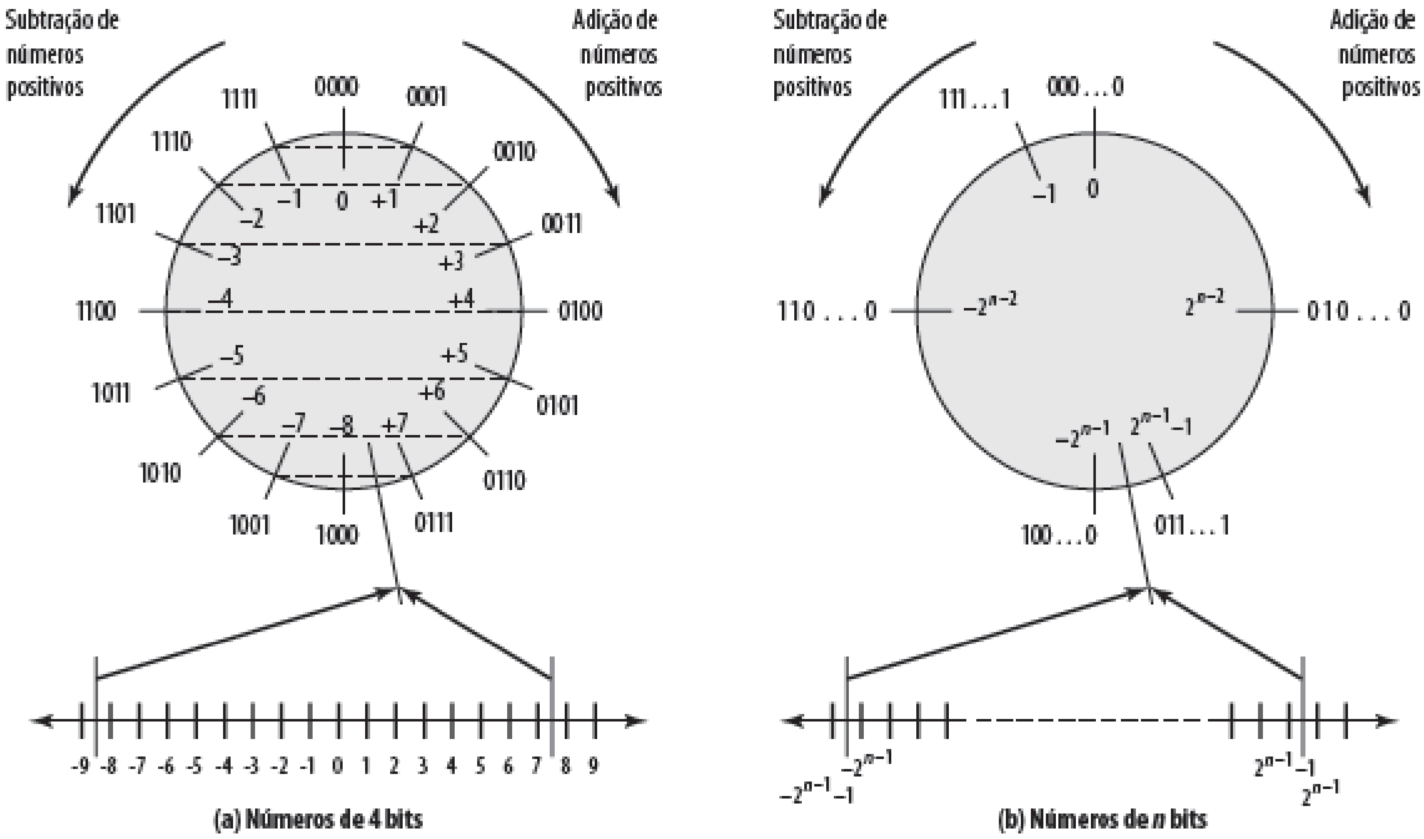
- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Complemento de dois

- Uma representação única do zero.
- Aritmética funciona com facilidade
  - Veremos mais adiante.

- A negação é muito fácil.
  - Ex.: 3 = 00000011
    - Complemento booleano gera
      - 11111100
    - Somar 1 ao LSB (bit menos significativo)
      - 11111101
  - Ex.: -3 = 11111101
    - Complemento booleano gera
      - 00000010
    - Somar 1 ao LSB
      - 00000011

Representação geométrica dos inteiros de complemento a dois



Complemento de dois – Negação especial – caso 1

- 0 = 00000000
  - Not bit a bit:
    - 11111111
  - Some 1 ao LSB:
    - +1
  - Resultado:
    - 1 00000000
  - O estouro (overflow) é ignorado, portanto:
    - -0 = +0 → CERTO!

Complemento de dois – Negação especial – caso 2

- - 128 = 10000000
  - Not bit-a-bit:
    - 01111111
  - Some 1 ao LSB:
    - +1
  - **Resultado:**
    - 10000000
  - Portanto:
    - -(-128) = -128 → ERRADO!
    - Monitorar o MSB (bit de sinal).
      - Ele deve mudar durante a negação.

- Quando somamos um valor negativo com um valor positivo é impossível ocorrer overflow.
  - O resultado tende a se aproximar de 0.
- Quando somamos dois valores com o mesmo sinal, devemos verificar se o bit de sinal muda:
  - EX: Somar dois números positivos (MSB=0) DEVE resultar em um número positivo (MSB=0).
  - EX: Somar dois números negativos (MSB=1) DEVE resultar em um número negativo (MSB=1).

Complemento de dois - Intervalo de números

- -2^{n-1} até $+2^{n-1}-1$

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

- Complemento a 2 com 16 bits:
  - $+32767 = 011111111\ 11111111 = 2^{15} - 1$
  - $-32768 = 100000000\ 00000000 = -2^{15}$

Complemento de dois - Conversão entre tamanhos

- Pacote de número positivo com zeros iniciais.
  - 8 bits
    - +18 = 00010010
  - 16 bits
    - +18 = 00000000 00010010
- Pacote de números negativos com uns iniciais.
  - 8 bits
    - -18 = 10010010
  - 16 bits
    - -18 = 11111111 10010010
- Ou seja, pacote com o MSB (bit de sinal).
  - MSB – bit mais significativo.

ADIÇÃO E SUBTRAÇÃO

Adição e subtração

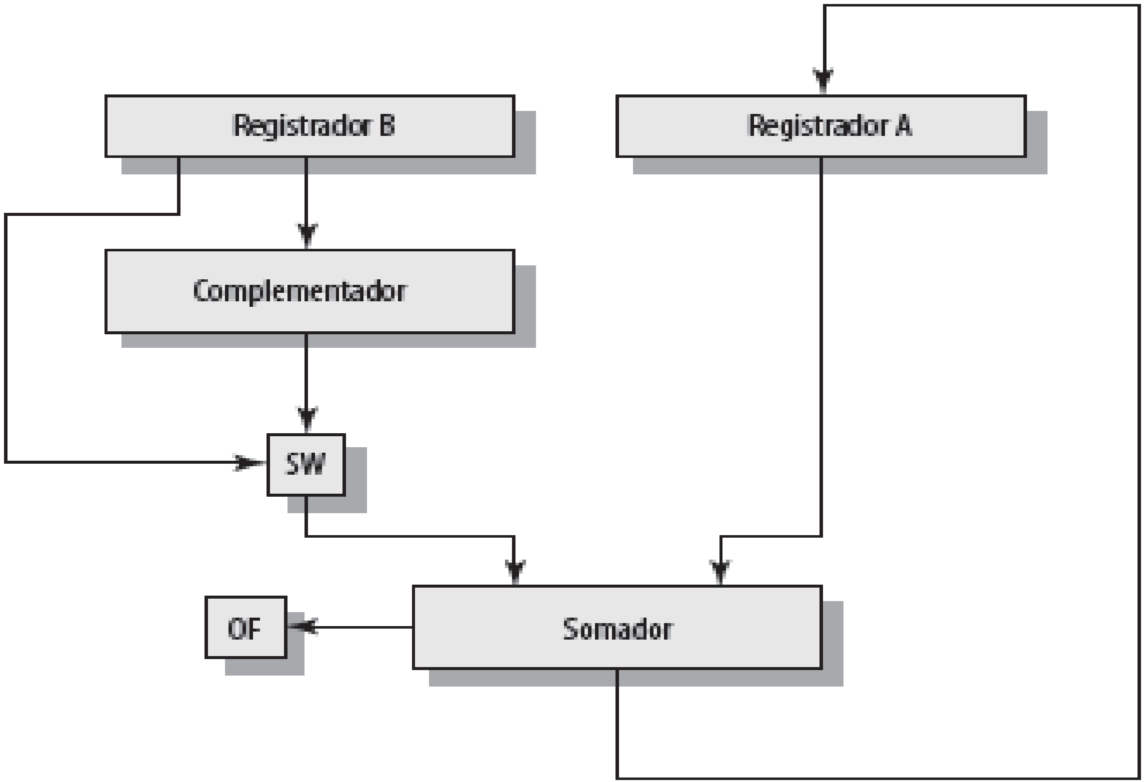
- Adição binária normal.
  - Monitore estouro no bit de sinal.

- Subtração
  - Pegue o complemento a dois do subtraendo e some ao minuendo.
  - Ou seja, a - b = a + (-b).

  - Assim, só precisamos de circuitos de adição e complemento.

Hardware para adição e subtração

- add $A, $B



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

MULTIPLICAÇÃO

Multiplicação

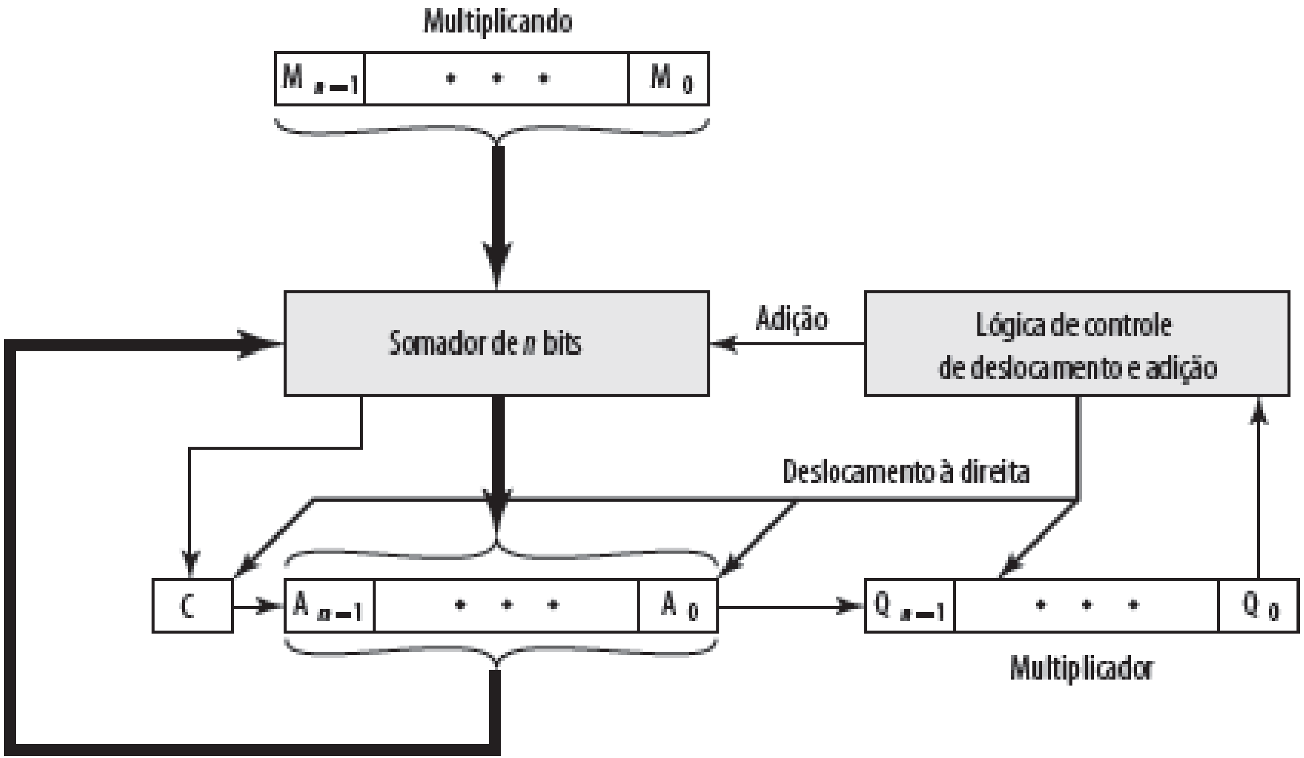
- Complexa.
- Calcule produto parcial para cada dígito.
- Cuidado com o valor da casa (coluna).
- Some produtos parciais.

Exemplo: Multiplicação

				1	0	1	1	Multiplicando (1110) M
×				1	1	0	1	Multiplicador (1310) Q
				1	0	1	1	Produtos parciais
			0	0	0	0		Produtos parciais
		1	0	1	1			Produtos parciais
+	1	0	1	1				Produtos parciais
1	0	0	0	1	1	1	1	Produto (14310)

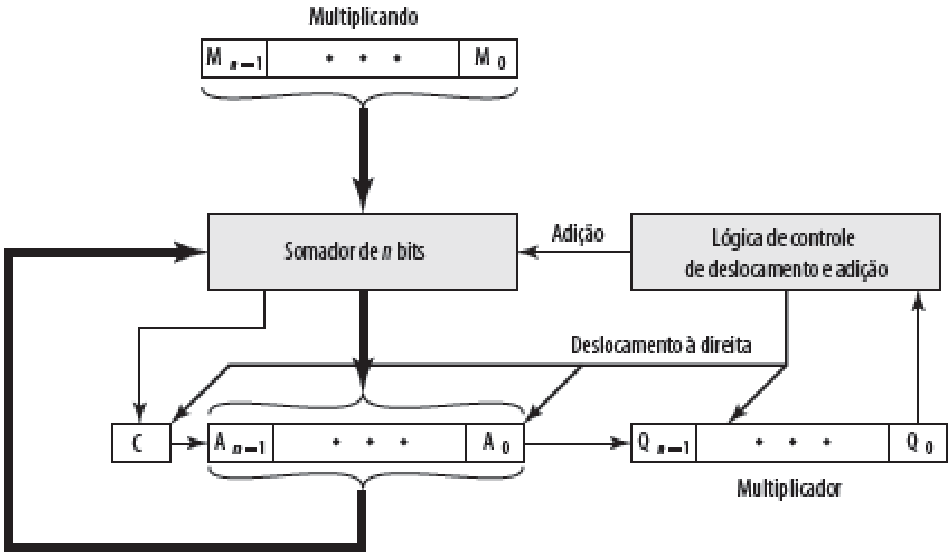
- Nota: Se o bit multiplicador for:
  - Copiar o multiplicando.
  - Caso contrário, zero.
- Nota: precisa de resultado com tamanho duplo.

Multiplicação binária sem sinal

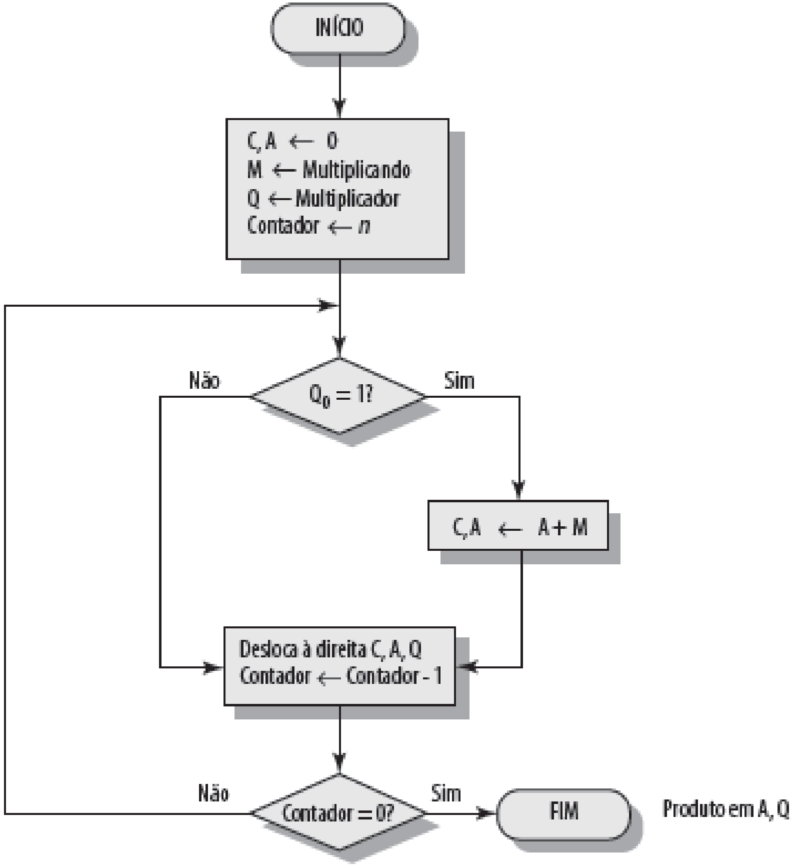


(a) Diagrama em blocos

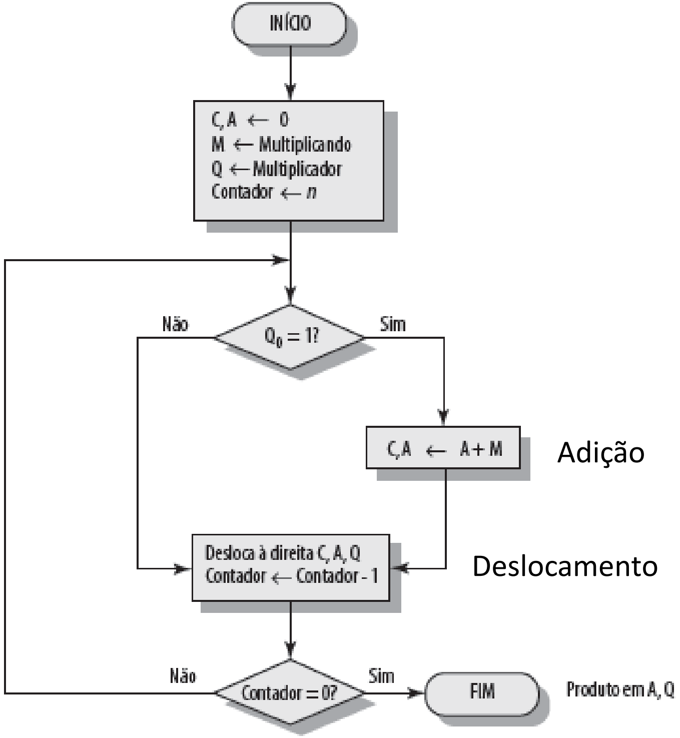
Fluxograma - Multiplicação Binária Sem Sinal



(a) Diagrama em blocos



Execução do exemplo



				1	0	1	1		M
×				1	1	0	1		Q
1	0	0	0	1	1	1	1		A Q

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.	

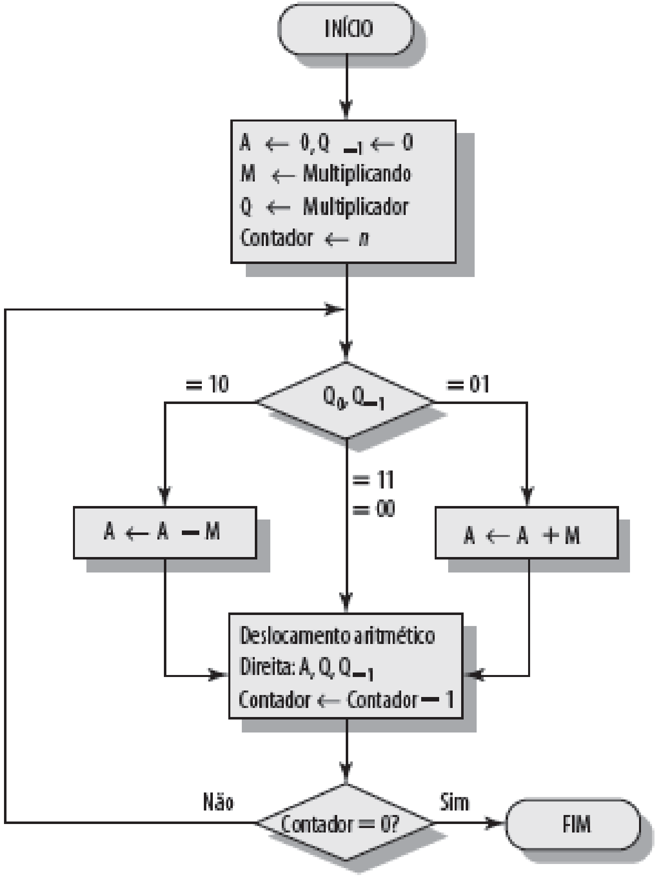
Multiplicando números negativos

- Solução 1:
  - Converta para positivo, se for preciso.
  - Multiplique como antes.
  - Se sinais diferentes, negue a resposta.
    - Exemplo:
      - 3 x 7 = 21;
      - (-3) x 7 = -21;
      - 3 x (-7) = -21;
      - (-3) x (-7) = 21;
  - Menos eficiente.

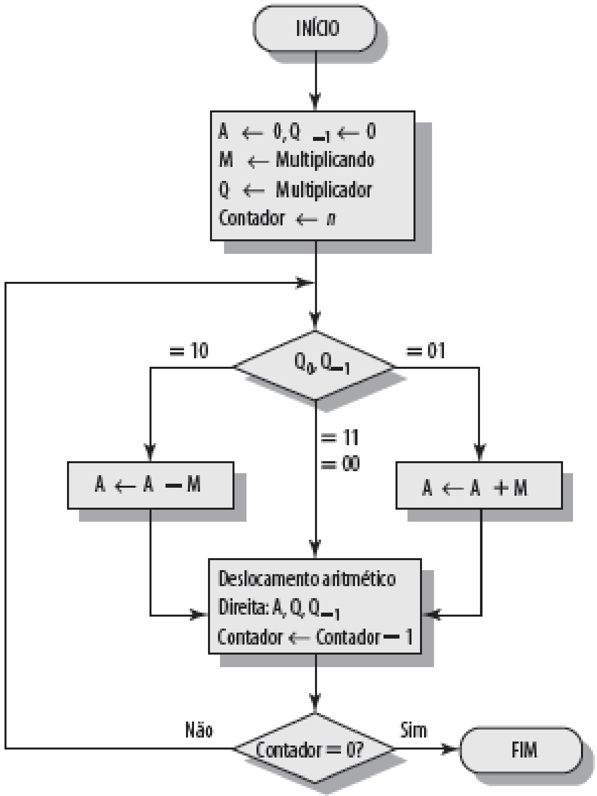
- Solução 2:
  - Algoritmo de Booth.

Algoritmo de Booth

- Funciona com números inteiros em **complemento a dois**.
  - Deslocamento aritmético a direita.
    - O valor da posição mais a esquerda é mantido após o deslocamento.
      - Ver *conversão entre tamanhos para complemento de dois.*
  - Produto em A,Q



Exemplo: Algoritmo de Booth



$$
\begin{array}{rcccccccccl}
 & & & & & 0 & 1 & 1 & 1 & & M \\
\times & & & & & 0 & 0 & 1 & 1 & & Q \\
\hline
 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & & AQ
\end{array}
$$

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

DIVISÃO

Divisão

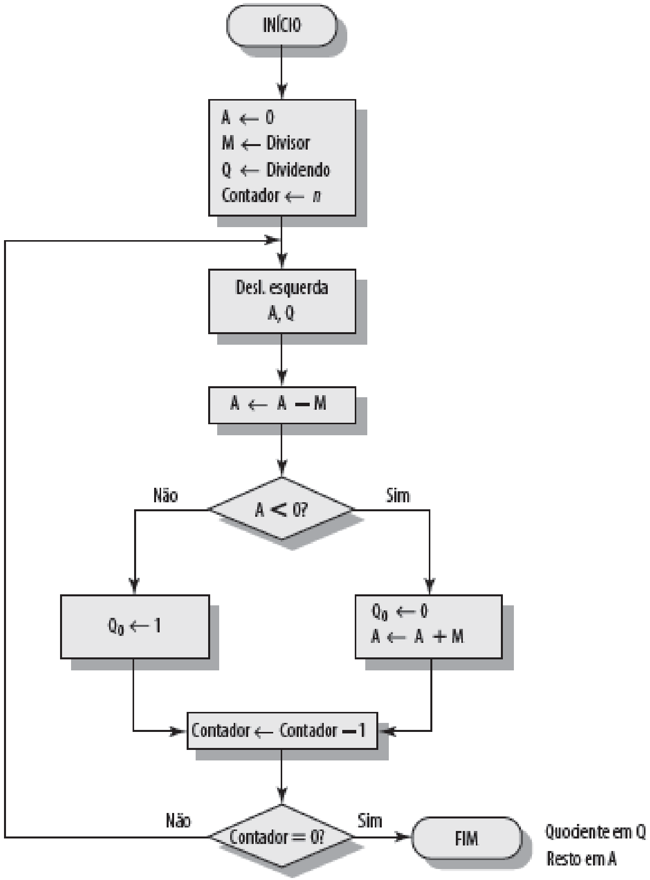
- Mais complexa que a multiplicação.
- Números negativos são realmente maus!
- Baseada na divisão longa.

Divisão de inteiros binários sem sinal

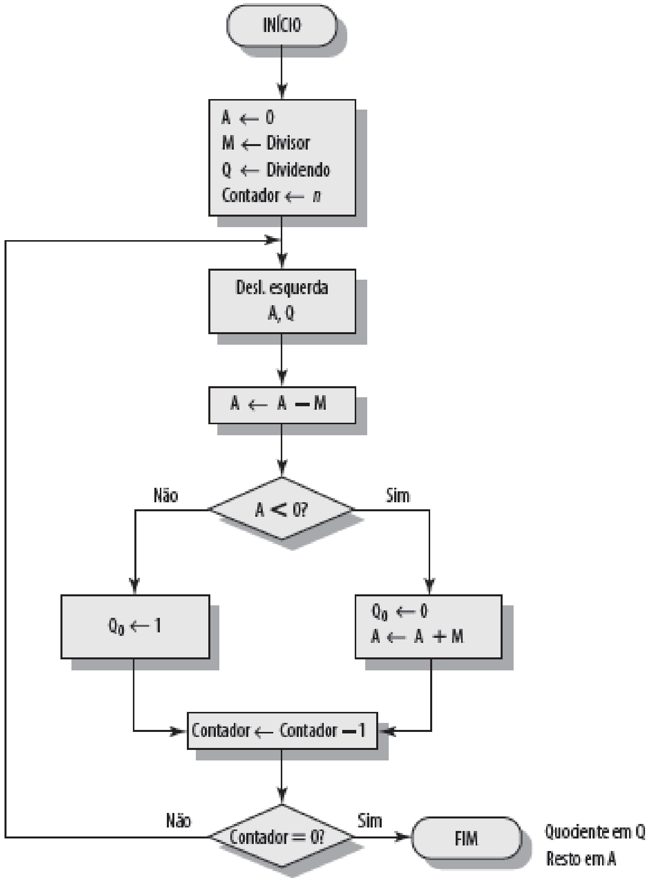
- 248/11 = 13 e restam 4

					0	0	0	0	1	1	0	1	← Quociente (A)
Divisor (M) →	1	0	1	1	1	0	0	1	0	0	1	1	← Dividendo (Q)
						1	0	1	1				
Resto parcial →					0	0	1	1	1	0			
							1	0	1	1			
Resto parcial →							0	0	1	1	1	1	
									1	0	1	1	
										1	0	0	← Resto (armazenado em Q)

Fluxograma para divisão binária sem sinal



Fluxograma para divisão binária sem sinal



A	Q	M = 0011
0000	0111	Valor inicial
0000	1110	Deslocar
1101		Subtrair
0000	1110	Restaurar
0001	1100	Deslocar
1110		Subtrair
0001	1100	Restaurar
0011	1000	Deslocar
0000		Subtrair
0000	1001	Fazer $Q_0 = 1$
0001	0010	Deslocar
1110		Subtrair
0001	0010	Restaurar

(a) (7) ÷ (3)

Tratando números negativos

- D = Q * V + R;

- Suponha todas as combinações possíveis de sinais de D e V:
  - D = 7 V = 3 → Q = 2 R = 1
  - D = 7 V = -3 → Q = -2 R = 1
  - D = -7 V = 3 → Q = -2 R = -1
  - D = -7 V = -3 → Q = 2 R = -1

Exemplo: 7/3; (-7)/3; 7/(-3) e (-7)/(-3)

- **<u>Algoritmo de divisão por restauração:</u>**

- Para realizar a divisão com operandos com sinal (complemento a dois):

1. Converta os operandos em valores sem sinal.
   – Para isso tome o complemento de 2 dos números negativos.
2. Realize a divisão utilizando o algoritmo de divisão para números sem sinal.
3. Derive o sinal de Q e de R a partir dos sinais de D e V.

A	Q	M = 0011
0000	0111	Valor inicial
0000	1110	Deslocar
1101		Subtrair
0000	1110	Restaurar
0001	1100	Deslocar
1110		Subtrair
0001	1100	Restaurar
0011	1000	Deslocar
0000		Subtrair
0000	1001	Fazer $Q_0 = 1$
0001	0010	Deslocar
1110		Subtrair
0001	0010	Restaurar

(a) (7) ÷ (3)

FIM – Aula 12