Aritmética com inteiros e

Representação e ponto flutuante

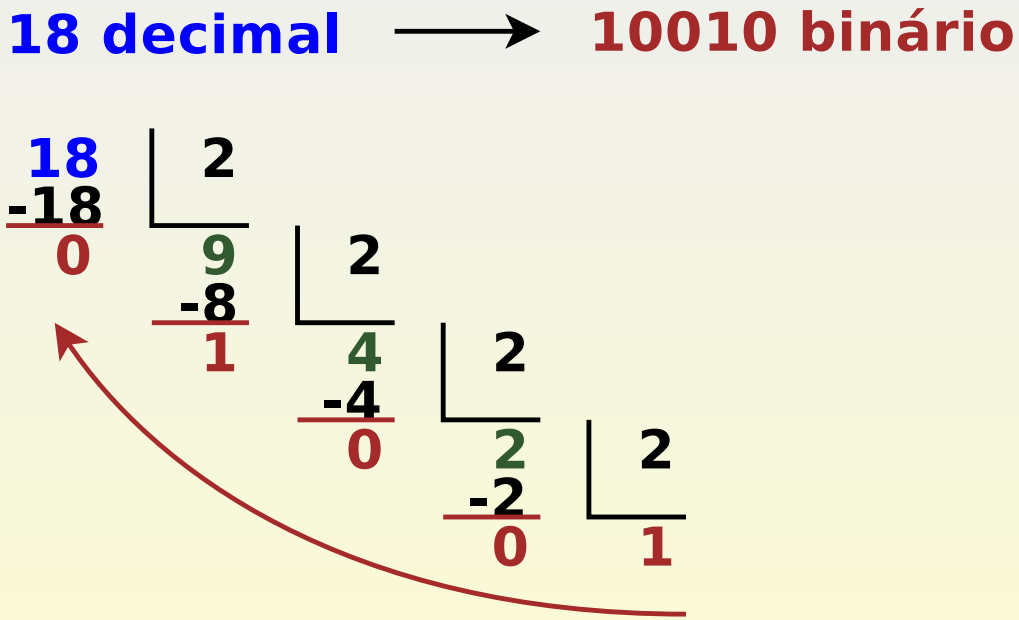
Fernando Pujaico Rivera[1]

[1]Universidade Federal de Lavras

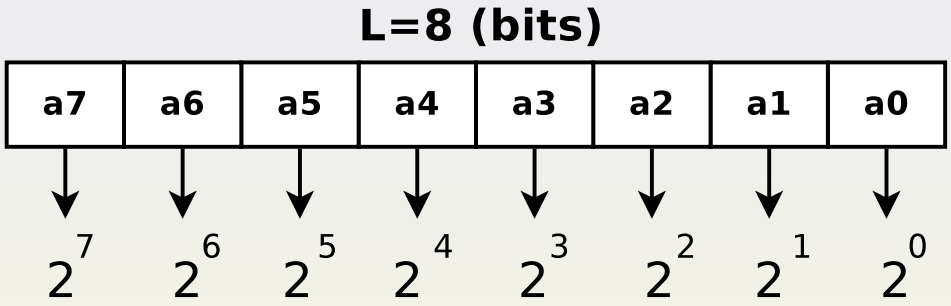
Aula-1 2016

Número inteiro positivo (decimal) → binário

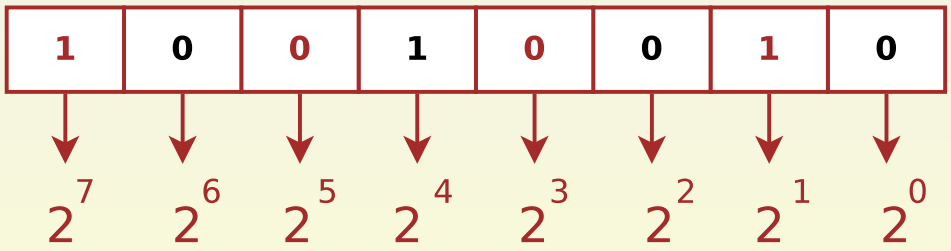
Divisões sucessivas por 2



Numero binário → inteiro positivo (decimal)



10010010 binário ⟶ 146 decimal

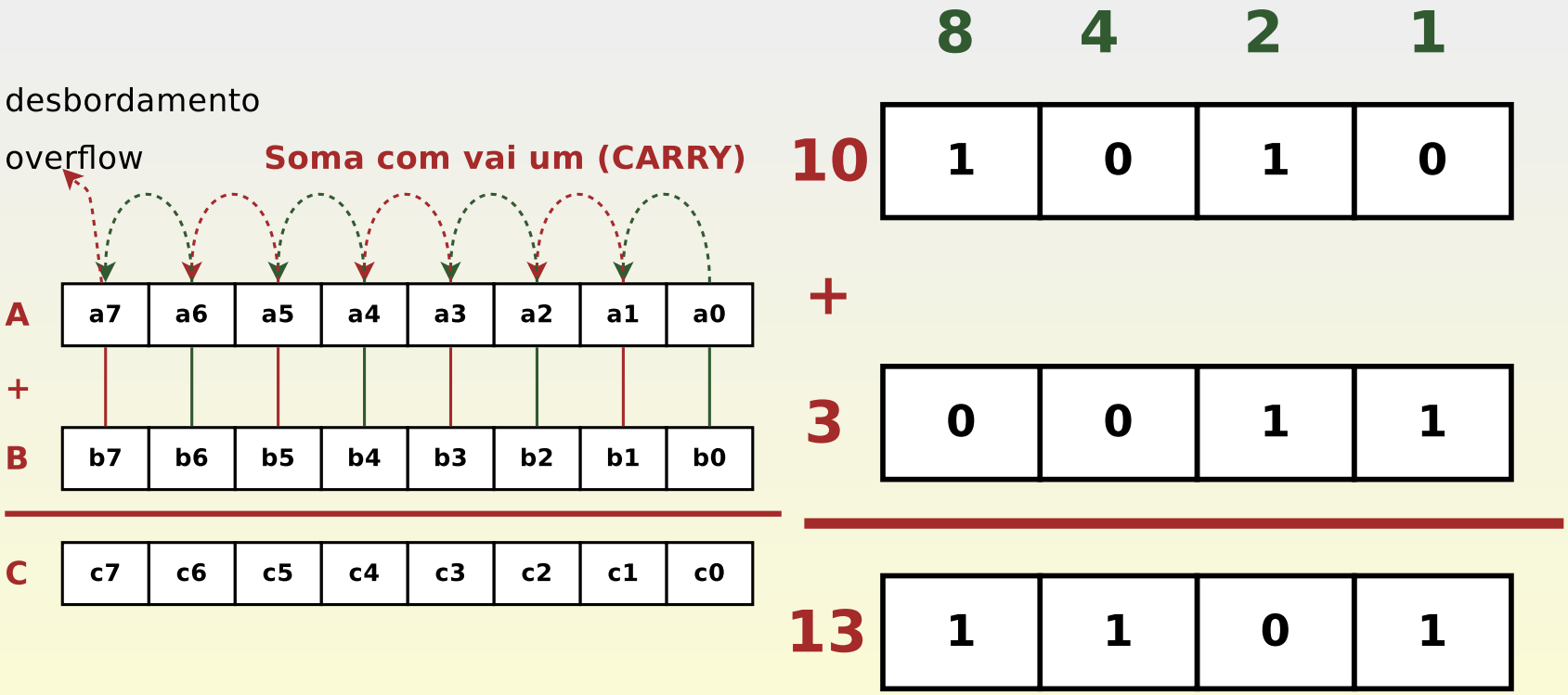


$+1x2^7$ $+0x2^6$ $+0x2^5$ $+1x2^4$ $+0x2^3$ $+0x2^2$ $+1x2^1$ $+0x2^0$

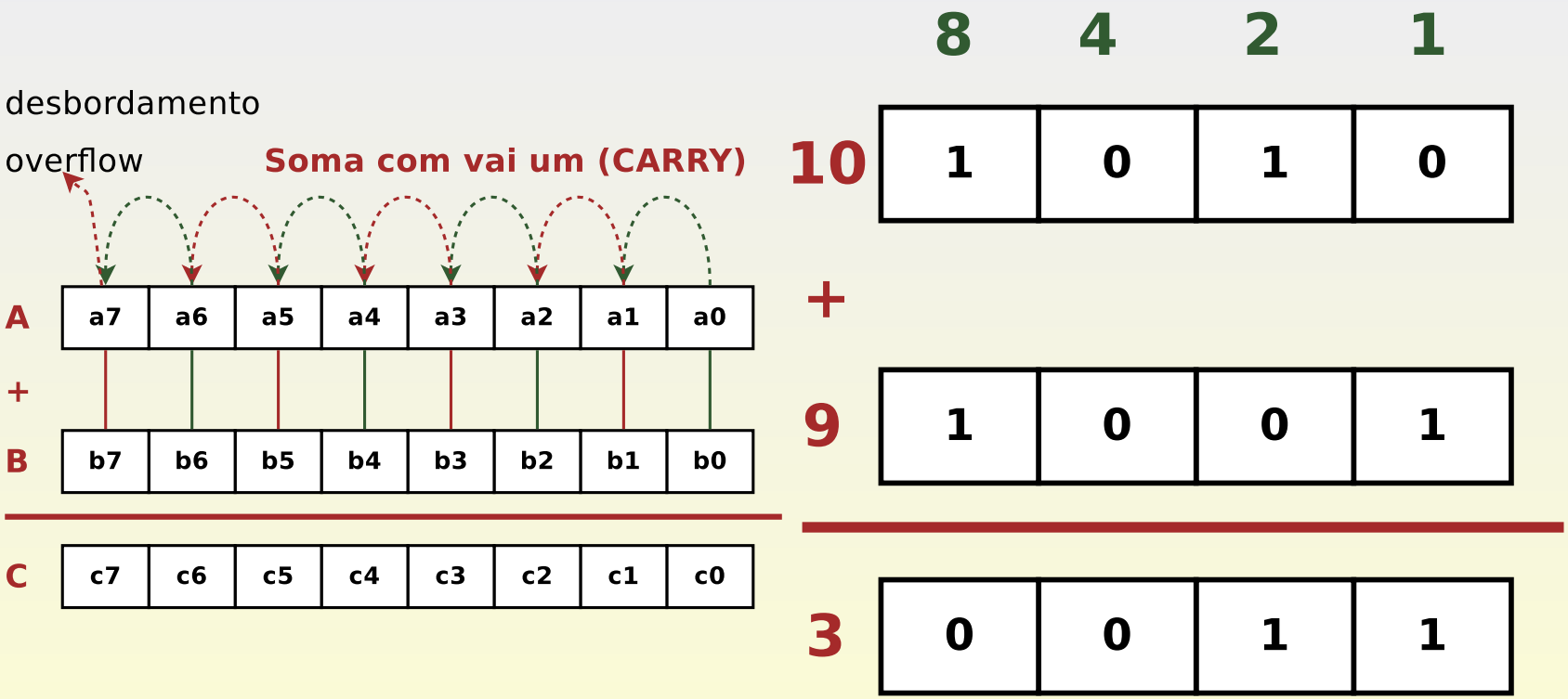
+128 +16 +2 **=146**

$$A = \sum_{i=0}^{L-1} a_i 2^i \qquad (1)$$

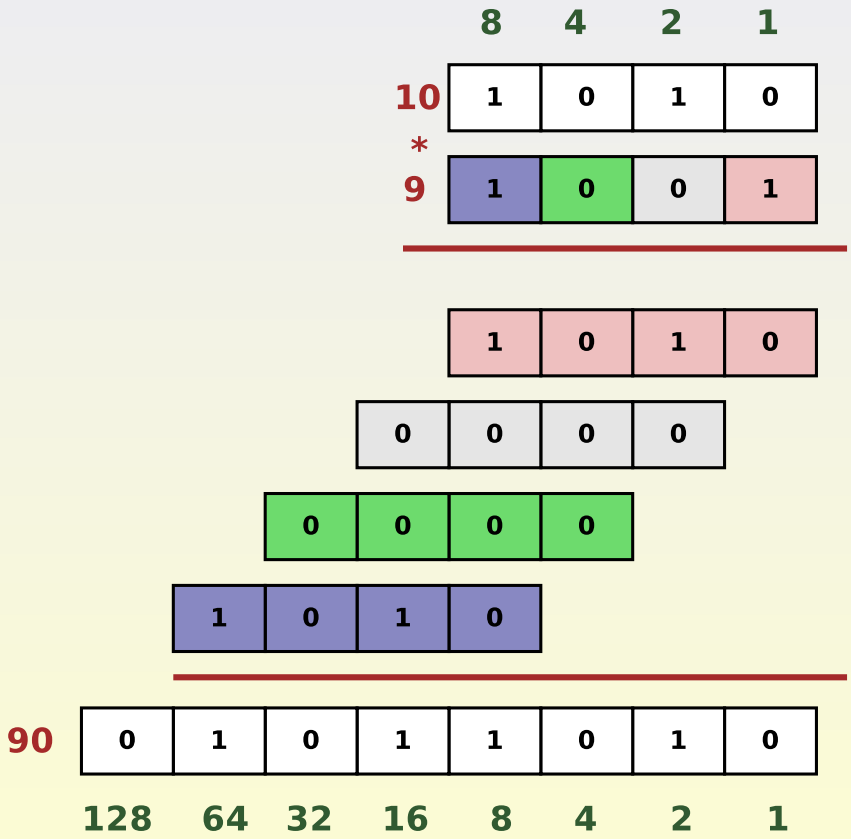
Soma de números inteiros positivos



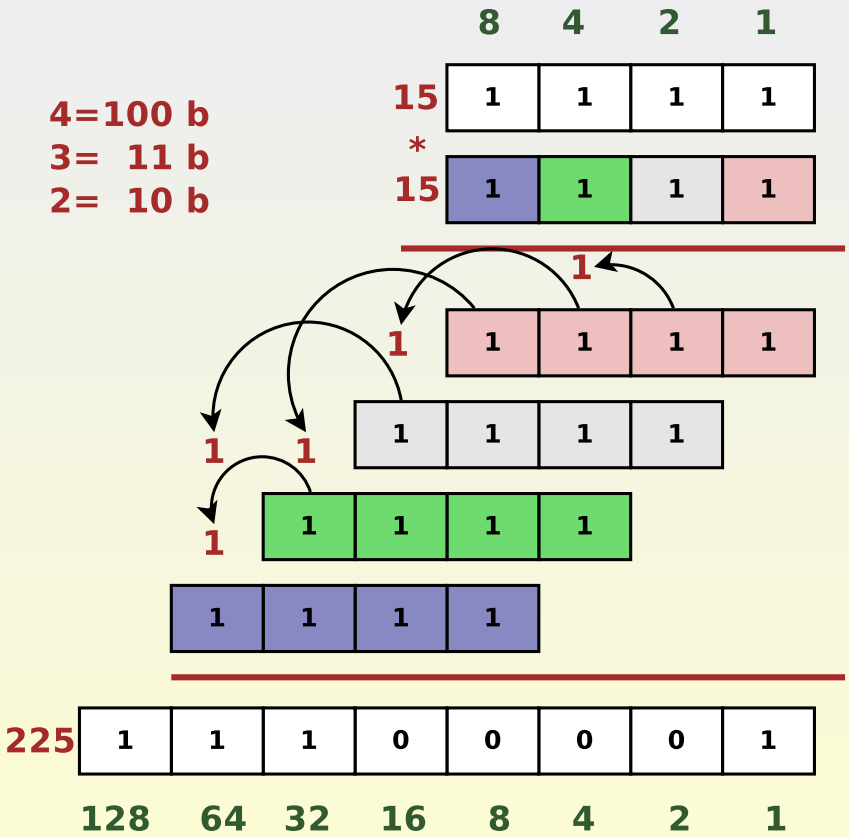
Soma de números inteiros positivos (desboradmento)



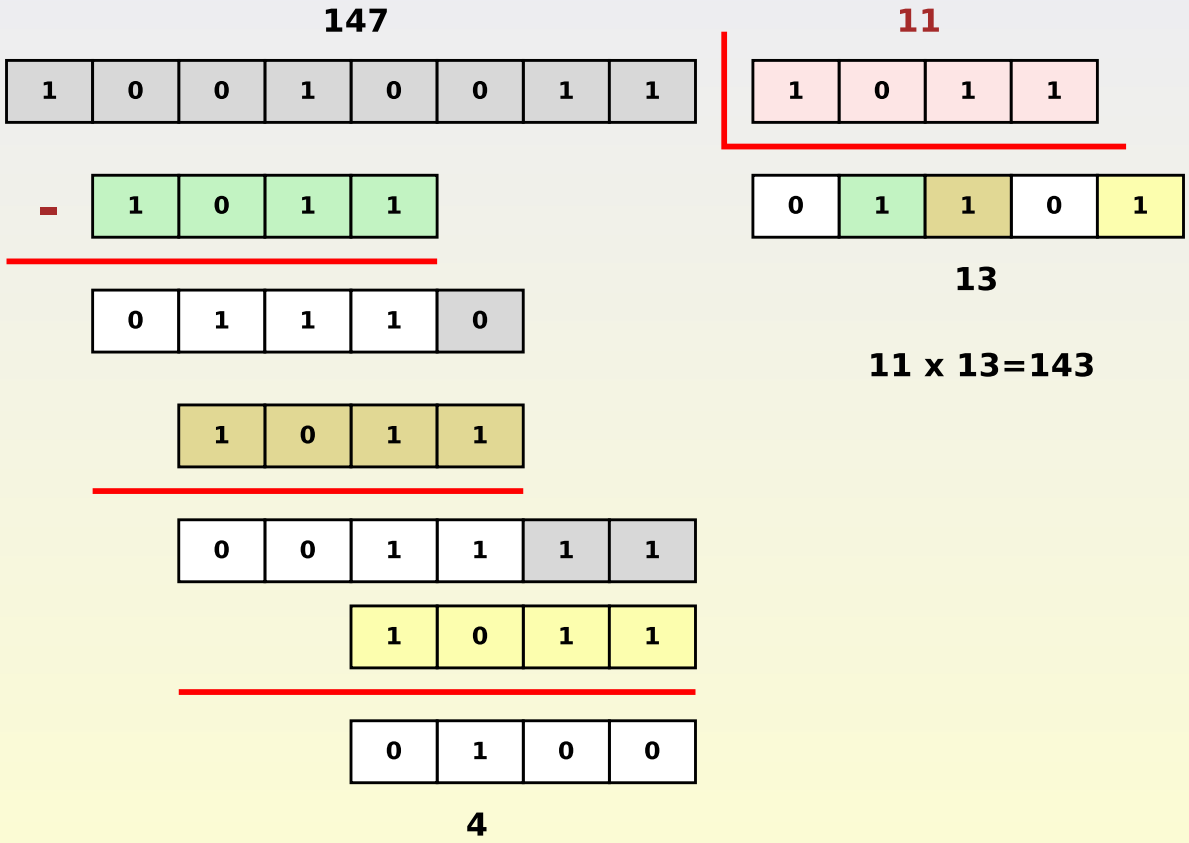
Multiplicação de números inteiros positivos



Multiplicação de números inteiros positivos



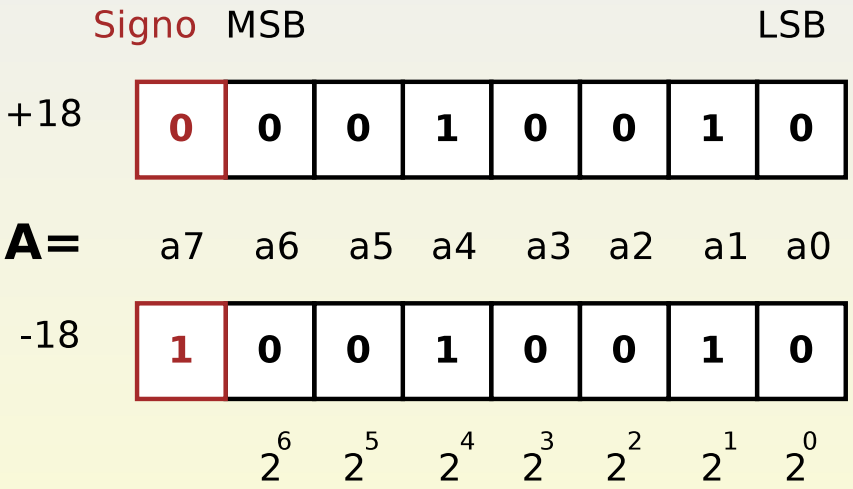
Divisão de números inteiros positivos



Num. negativo com representação Sinal-magnitude [1]

00000000=0 dec ??? 10000000=0 dec

L=8 (bits)



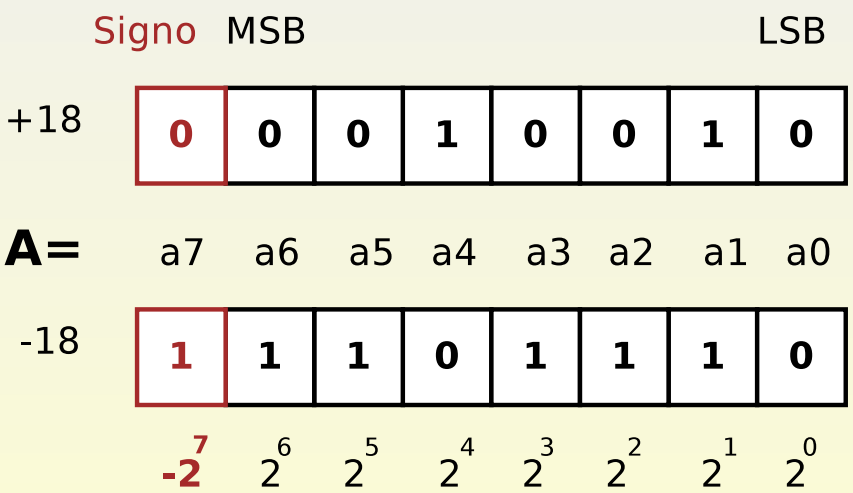
$$D_A = \begin{cases} +\sum_{i=0}^{L-2} a_i 2^i & \text{if } a_{L-1} = 0 \\ -\sum_{i=0}^{L-2} a_i 2^i & \text{if } a_{L-1} = 1 \end{cases} \quad (2)$$

Num. negativo com representação complemento a dois

$X = \overline{A} + 1$, X é o complemento a dois de A.

$-A = \overline{A} + 1$, $-A$ é o complemento a dois de A.

L=8 (bits)



$$D_A = -2^{L-1} a_{L-1} + \sum_{i=0}^{L-2} a_i 2^i \quad (3)$$

Num. negativo com representação complemento a dois

$X = \overline{A} + 1$, X é o complemento a dois de A.

$-A = \overline{A} + 1$, $-A$ é o complemento a dois de A.

L=3 (bits)

0	**0**	**0**	**0**
1	**0**	**0**	**1**
2	**0**	**1**	**0**
3	**0**	**1**	**1**
-4	**1**	**0**	**0**
-3	**1**	**0**	**1**
-2	**1**	**1**	**0**
-1	**1**	**1**	**1**

$$-2^{L-1} \leq A \leq 2^{L-1} - 1 \quad (4)$$

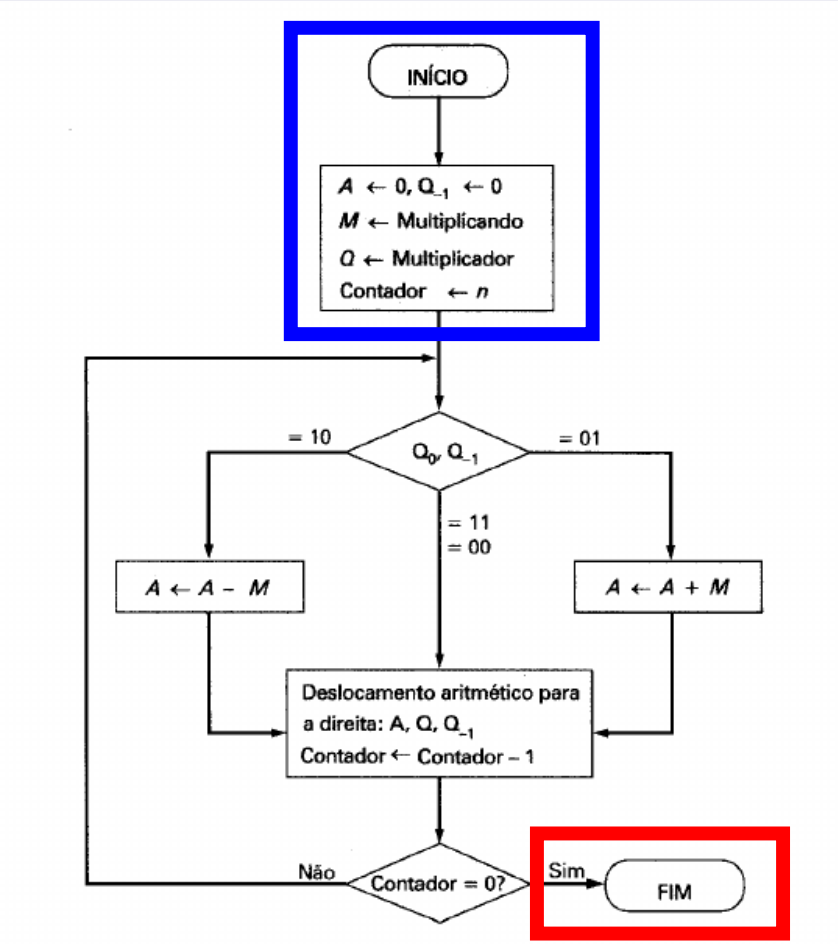
$$B - A = B + \overline{A} + 1 \quad (5)$$

Soma de números inteiros em complemento a dois

	-8	4	2	1
-6	1	0	1	0
+				
3	0	0	1	1
-3	1	1	0	1

	-8	4	2	1
-6	1	0	1	0
+				
6	0	1	1	0
0	0	0	0	0

Multiplicação de números inteiros em complemento a dois



Algoritmo de Booth para a multiplicação em complementos de dois.

M*Q=18

M=-6 **Q=-3**

M	1	0	1	0					
-M	0	1	1	0			**Q**		**Q_{-1}**
4:A	0	0	0	0	1	1	0	1	0
	0	1	1	0	1	1	0	1	0
3:A	0	0	1	1	0	1	1	0	1
	1	1	0	1	0	1	1	0	1
2:A	1	1	1	0	1	0	1	1	0
	0	1	0	0	1	0	1	1	0
1:A	0	0	1	0	0	1	0	1	1
0:A	0	0	0	1	0	0	1	0	1

16 **2**

Divisão de números inteiros com signo

1. Carregar o divisor no registrador M e o dividendo nos registradores A e Q. O dividendo deve ser expresso como um número em complemento de dois com $2n$ bits. Por exemplo, o número 0111 de 4 bits seria representado como 00000111 e o número 1001, como 11111001.
2. Deslocar o conteúdo dos registradores A e Q, juntos, um bit para a esquerda.
3. Se M e A têm o mesmo sinal, fazer $A \leftarrow A - M$; caso contrário, $A \leftarrow A + M$.
4. A operação anterior será bem-sucedida se o sinal de A for o mesmo, antes e depois da operação.
   a. Se a operação for bem-sucedida ou se ($A = 0$ e $Q = 0$), então faça $Q_0 \leftarrow 1$.
   b. Se a operação não for bem-sucedida e se ($A \neq 0$ ou $Q \neq 0$), então faça $Q_0 \leftarrow 0$ e restaure o antigo valor de A (somando M a A).
5. Repita os passos 2 a 4 enquanto houver bits a examinar em Q.
6. Ao final, o resto estará em A. Se o divisor e o dividendo tiverem o mesmo sinal, o quociente estará em Q; caso contrário, o quociente correto é o complemento de dois do número armazenado em Q.

-7= Q=1001
-7=AQ=11111001 **3=-M=0011**

A	Q	-3= M = 1101	n =
1111	1001	Valor inicial	4
1111	0010	Deslocar	
0010=A-M		Subtrair	
1111	0010	Restaurar	3
1110	0100	Deslocar	
0001 =A-M		Subtrair	
1110	0100	Restaurar	2
1100	1000	Deslocar	
1111 =A-M		Subtrair	
1111	1001	Fazer $Q_0 = 1$	1
1111	0010	Deslocar	
0010 =A-M		Subtrair	
1111	0010	Restaurar	0

(d) (–7) ÷ (–3)

Representação em ponto flutuante [1]

	Sinal	Expoente	Mantissa
64bit	***1bit***	***k=11bits***	***52bits***
32bit	***1bit***	***k=8bits***	***23bits***

IEEE-754

$$s\ 1.M\ 2^E$$

$$s \equiv \begin{cases} + & se \quad Sinal \equiv 0 \\ - & se \quad Sinal \equiv 1 \end{cases}$$

$$M \equiv Mantissa$$

$$E \equiv Expoente - \left(2^{k-1} - 1\right)$$

IEEE-754

32bit	1	10000010	001100 ...

$$S \equiv -$$

$$M \equiv 00110000...$$

$$E \equiv 130 - 127 = 3$$

$$-\ 1.0011\ 2^3$$

$$-\ 9.5$$

Divisão de números inteiros positivos

	Single Precision (32 bits)			
	Sign	**Biased exponent**	**Fraction**	**Value**
positive zero	0	0	0	0
negative zero	1	0	0	-0
plus infinity	0	255 (all 1s)	0	∞
minus infinity	1	255 (all 1s)	0	$-\infty$
quiet NaN	0 or 1	255 (all 1s)	$\neq 0$	NaN
signaling NaN	0 or 1	255 (all 1s)	$\neq 0$	NaN
positive normalized nonzero	0	$0 < e < 255$	f	$2^{e-127}(1.f)$
negative normalized nonzero	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$

Divisão de números inteiros positivos

	Double Precision (64 bits)			
	Sign	**Biased exponent**	**Fraction**	**Value**
positive zero	0	0	0	0
negative zero	1	0	0	-0
plus infinity	0	2047 (all 1s)	0	∞
minus infinity	1	2047 (all 1s)	0	$-\infty$
quiet NaN	0 or 1	2047 (all 1s)	$\neq 0$	NaN
signaling NaN	0 or 1	2047 (all 1s)	$\neq 0$	NaN
positive normalized nonzero	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
negative normalized nonzero	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$

Divisão de números inteiros positivos

Parâmetro	Simples	Duplo
Tamanho da palavra (bits)	32	64
Tamanho do expoente (bits)	8	11
Polarização do expoente	127	1023
Expoente máximo	127	1023
Expoente mínimo	–126	–1022
Faixa de números (base 10)	10^{-38}, 10^{+38}	10^{-308}, 10^{+308}
Tamanho da mantissa (bits)*	23	52
Número de expoentes	254	2046
Número de frações	2^{23}	2^{52}
Número de valores	$1{,}98 \times 2^{31}$	$1{,}99 \times 2^{63}$

Divisão de números inteiros positivos

Floating Point Numbers	**Arithmetic Operations**
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{array}{l} X + Y = (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E} \\ X - Y = (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E} \end{array}\right\} X_E \leq Y_E$ $X \times Y = (X_S \times Y_S) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

References I

[1] William Stallings. *Arquitetura e Organização de Computadores - 8 Ed.* Prentice Hall.