



Universidade Tecnológica Federal do Paraná – UTFPR
Bacharelado em Ciência da Computação

BCC33B – Arquitetura e Organização de Computadores

Prof. Rogério A. Gonçalves
rogerioag@utfpr.edu.br

Aula 015

ULA

Unidade Lógica e Aritmética



Unidade Lógica e Aritmética

- **Realiza as operações lógicas e aritméticas**
 - add, sub, and, or...
- **Faz os cálculos.**
- **Trata de inteiros.**
- **Pode tratar de números de ponto flutuante (reais).**
- **Pode ser FPU separada (coprocessador matemático).**
- **Pode estar em chip de FPU separado (486DX +).**

Unidade Lógica e Aritmética



Representação de Dados

Como a informação é representada num processador?

- A representação de um dado corresponde aos dígitos que escrevemos para simbolizá-lo.

- Exemplos:

Valor (quantidade) 12

Representações:

{ Hexadecimal ⇒ C
Romano ⇒ XII
Binário ⇒ 1100

Representação de Dados

Números de Ponto Fixo (Inteiros)

Números de Ponto Fixo Sem Sinal: usam representação binária convencional

Exemplo:

Binário Decimal

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

O valor do número é inteiro.
Nenhum bit é usado para
representar sinal.

Representação de Dados

Números de Ponto Fixo (Inteiros)

Números de Ponto Fixo Com Sinal

Existem 4 Métodos de Representação:

1. Sinal Magnitude
2. Complemento de 1
3. Complemento de 2
4. Notação em Excesso

Representação de Dados

Números de Ponto Fixo (Inteiros)

Representação Sinal Magnitude:

- Em decimal para representarmos as quantias +12 e -12 \Rightarrow usamos os sinais **+** e **-** para indicar se o número é positivo ou negativo
- Em Sinal Magnitude: Bit mais significativo (mais à esquerda) indica o sinal do número representado

0 indica número positivo

1 indica número negativo

Os bits restantes representam a Magnitude (valor do dado)

Representação de Dados

Números de Ponto Fixo (Inteiros)

Exemplo na Representação Sinal Magnitude:

$+12_{10} \Rightarrow$	00001100_2
$-12_{10} \Rightarrow$	10001100_2

→ Só muda o bit de sinal

Os bits restantes representam a Magnitude (valor do dado)

Representação de Dados

Números de Ponto Fixo (Inteiros)

Observações para a Representação Sinal Magnitude:

1. Há 2 representações para o número 0

$+0_{10} \Rightarrow 00000000_2$

$-0_{10} \Rightarrow 10000000_2$

-Pode gerar erros de programação
-Requer hardware mais complexo para
comparar com os dois 0s.

Dificulta testes

Representação de Dados

Números de Ponto Fixo (Inteiros)

Observações para a Representação Sinal Magnitude:

2. Intervalo de representação é menor, isto é, a quantidade de números representáveis é menor

011	+3
010	+2
001	+1
000	+0
100	-0
101	-1
110	-2
111	-3

Exemplo: $2^3=8$

Isso significa que com 3 bits poderíamos representar até 8 valores diferentes, mas devido às duas representações do valor 0 (+0 e -0) podemos representar até 7 valores diferentes

Representação de Dados

Números de Ponto Fixo (Inteiros)

Representação em Complemento de 1:

- Na representação em Complemento de 1 nós complementamos (invertemos) todos os bits 1 por 0 e os bits 0 por 1
- Exemplo:

$$\begin{array}{l} +12_{10} \Rightarrow 00001100_2 \\ -12_{10} \Rightarrow 11110011_2 \end{array}$$

Os números **positivos** também têm
o bit mais significativo em 0
e os números **negativos** em 1

Representação de Dados

Números de Ponto Fixo (Inteiros)

Observações para a Representação Complemento de 1:

1. Há também 2 representações para o número 0

$+0_{10} \Rightarrow 00000000_2$

$-0_{10} \Rightarrow 11111111_2$

-Pode gerar erros de programação

-Requer hardware mais complexo para comparar com os dois 0s.

Dificulta testes

$A \neq 00000000$

E

$A \neq 11111111$

Representação de Dados

Números de Ponto Fixo (Inteiros)

Observações para a Representação Complemento de 1:

2. Intervalo de representação é menor, isto é, a quantidade de números representáveis é menor

011	+3
010	+2
001	+1
000	+0
111	-0
110	-1
101	-2
100	-3

Exemplo: $2^3=8$

Isso significa que com 3 bits poderíamos representar até 8 valores diferentes, mas devido às duas representações do valor 0 (+0 e -0) podemos representar até 7 valores diferentes

Representação de Dados

Números de Ponto Fixo (Inteiros)

Representação em Complemento de 2:

- Na representação em Complemento de 2 nós complementamos (invertemos) todos os bits 1 por 0 e os bits 0 por 1 e somamos 1 ao resultado do Complemento de 1
- Exemplo:

$$+12_{10} \Rightarrow 00001100_2$$

Em Complemento de 2 os números positivos também têm o bit mais significativo em 0 e os números negativos em 1

$$-12_{10} \Rightarrow C1 = 11110011_2$$

+1

$$-12_{10} = 11110100_2$$

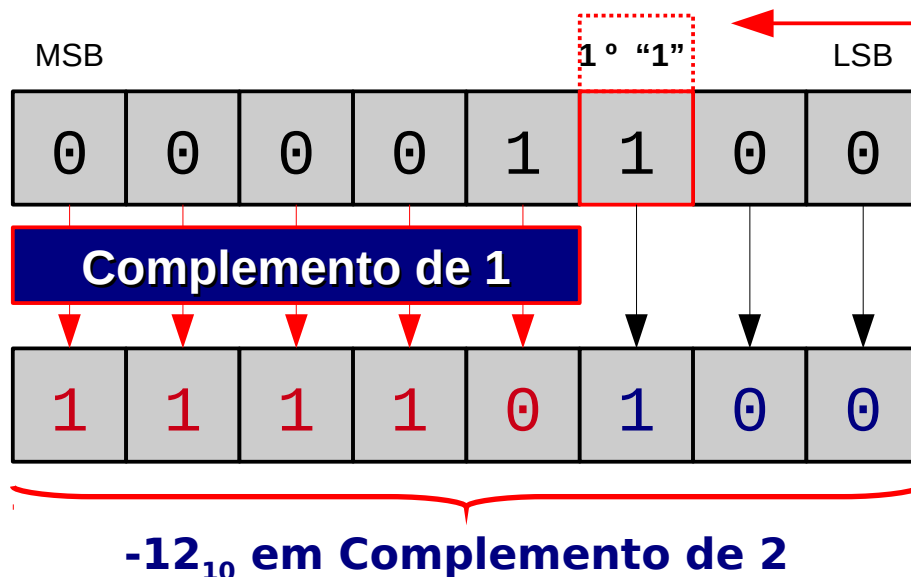
-12_{10} em Complemento de 2

Representação de Dados

Representação em Complemento de 2

- Método Alternativo
- Troque todos os bits à esquerda do bit 1 menos significativo.
- Passos:
 1. Comece à direita com o LSB e escreva os bits como eles aparecem até o primeiro 1 (inclusive ele).
 2. Tome o complemento de 1 dos bits restantes.
- Exemplo:

$$+12_{10} \Rightarrow 00001100_2$$



Representação de Dados

Números de Ponto Fixo (Inteiros)

Observações para a Representação Complemento de 2:

2. Intervalo de representação é **maior** que dos outros métodos de representação anteriores porque só há uma representação para o Zero

011	+3
010	+2
001	+1
000	+0
000	-0
111	-1
110	-2
101	-3
100	-4

Intervalo maior: 8 representações diferentes

Representação de Dados

Números de Ponto Fixo (Inteiros)

Representação em Excesso (Bias ou Deslocamento):

- A representação em Excesso tem o efeito de deslocar o número a ser representado, de forma que, o menor valor (negativo) corresponda à representação com todos os bits em zero e os valores sejam representados em ordem crescente, a partir do menor
- Exemplo em Excesso de 128:

$$+12_{10} \Rightarrow +12 + 128 = 140 = 10001100_2$$

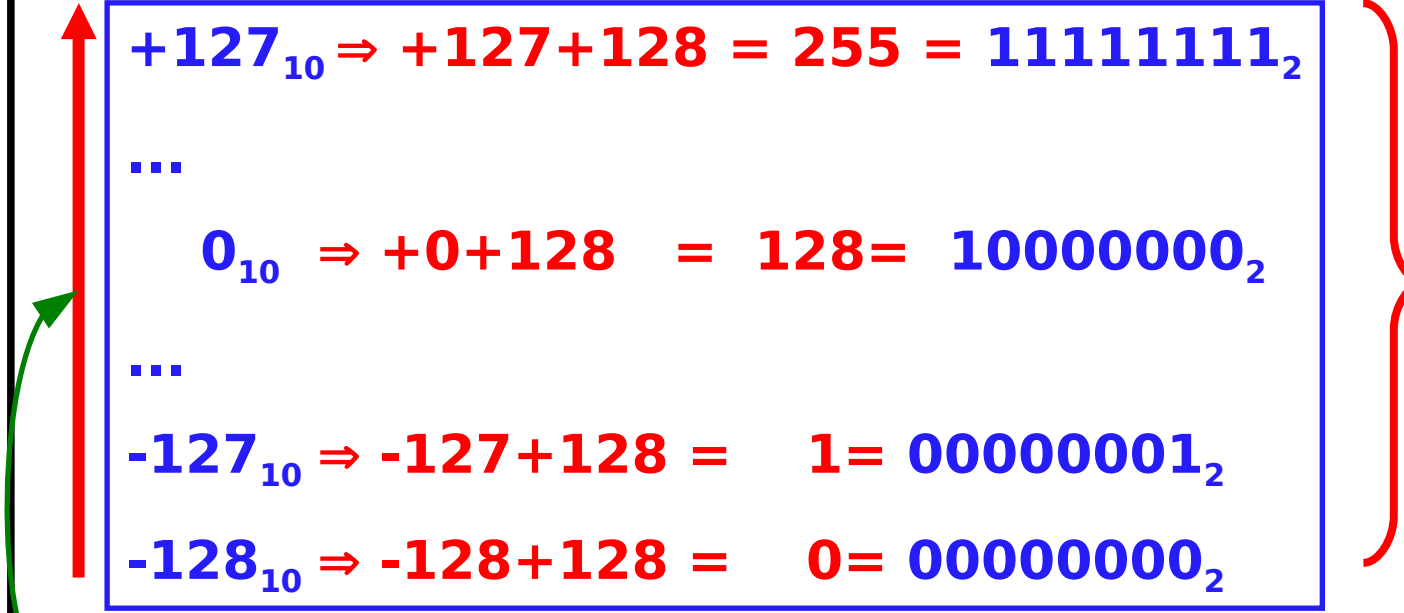
$$-12_{10} \Rightarrow -12 + 128 = 116 = 01110100_2$$

Representação de Dados

Números de Ponto Fixo (Inteiros)

Observações para a Representação Excesso:

1. Há somente 1 representação para o número 0
2. Intervalo de representação maior


$$+127_{10} \Rightarrow +127 + 128 = 255 = 11111111_2$$

...

$$0_{10} \Rightarrow +0 + 128 = 128 = 10000000_2$$

...

$$-127_{10} \Rightarrow -127 + 128 = 1 = 00000001_2$$

$$-128_{10} \Rightarrow -128 + 128 = 0 = 00000000_2$$

Com 8 bits pode-se representar $2^8=256$

números (de 0 a 255)

Ordem crescente facilita comparações entre os números

Representação de Dados

Resumo das Representações de Dados

Decimal	Sem Sinal	Sinal Magnitude	Complement o de 1	Complemen to de 2	Excesso de 4
+7	111				
+6	110				
+5	101				
+4	100				
+3	011	011	011	011	111
+2	010	010	010	010	110
+1	001	001	001	001	101
+0	000	000	000	000	100
-0	-	100	111	000	100
-1	-	101	110	111	011
-2	-	110	101	110	010
-3	-	111	100	101	001
-4	-			100	000

Representação de Dados

Números em Ponto Flutuante (Reais)

Problema: Ponto Fixo requer uma quantidade muito grande de dígitos para representar números muito grandes ou muito pequenos

- **Exemplo:** Para representar 1 Trilhão \Rightarrow Requer 40 bits à esquerda do ponto fixo



- **Exemplo:** Para representar 1 Trilionésimo no mesmo processador \Rightarrow Requer 40 bits à direita do ponto fixo

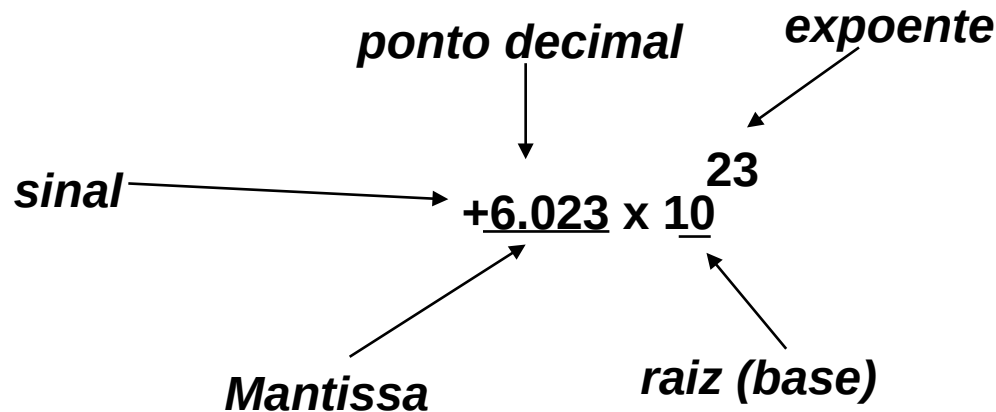


- No total precisamos de 80 bits por número

Representação de Dados

Números em Ponto Flutuante (Reais)

- Exemplo: Número de Avogrado $+6,023 \times 10^{23}$
- Intervalo: 10^{23}
- Precisão: 6,023 (3 dígitos de precisão)
- Representação do Número em Notação Científica:



Representação de Dados

Números em Ponto Flutuante (Reais)

- Obs: Há várias maneiras de se representar o mesmo número
- Exemplos:

$$3584,1 \times 10^0 = 3,5841 \times 10^3 = 0,35841 \times 10^4$$

Várias representações dificultam cálculos e comparações



Necessidade de Normalização da representação

Representação de Dados

Números em Ponto Flutuante (Reais)

Normalização: o ponto é deslocado (“flutua”) para a esquerda do dígito diferente de 0 mais à esquerda (bit mais significativo), e o expoente é ajustado

- Exemplo:

0,35841 x 10⁴

← **Forma Normalizada**

- Obs: Para representar 0 usa-se a mantissa com todos os valores em 0

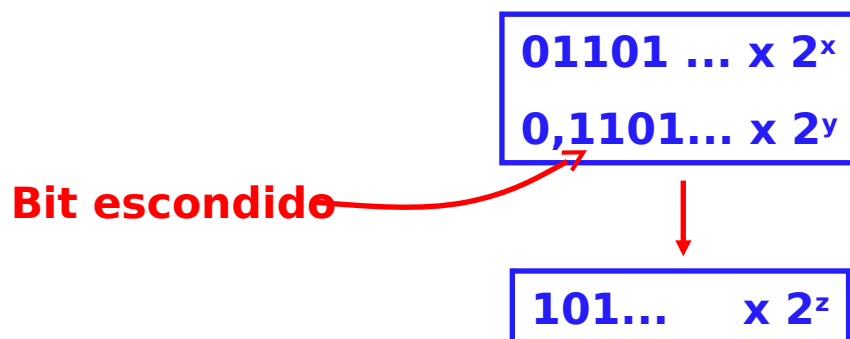
Representação de Dados

Números em Ponto Flutuante (Reais)

Observações para representação em Ponto Flutuante:

- Para representar 0 usa-se a mantissa com todos os valores em 0
- Em binário, não há necessidade de se armazenar o dígito 1 “mais significativo” da mantissa (já se sabe que ele é 1) \Rightarrow esse bit é chamado de “bit escondido”. Sobra mais espaço para o número ser representado \Rightarrow aumenta a precisão

Exemplo



Representação de Dados

Números em Ponto Flutuante (Reais)

Padrão IEEE 754

- 1980: Padronização da representação em Ponto Flutuante pela IEEE (*Institute of Electrical and Electronics Engineers*)
- Padronização:
 - Facilita a troca de dados entre diferentes computadores
 - Facilita os algoritmos aritméticos de PF, pois tratam os números sempre no mesmo formato
 - Melhora a precisão dos números representados devido ao bit escondido

Representação de Dados

Números em Ponto Flutuante (Reais)

Padrão IEEE 754 (ANSI/IEEE 754-1985)

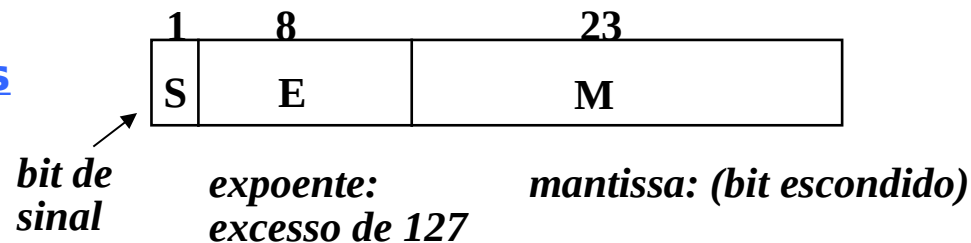
- São três formas:
 - **Precisão Simples:** 32 bits {**S** → 1, **E** → 8, **M** → 23}
 - **E** é representado em excesso de 127 e **M** efetivamente possui 24 bits, um bit escondido.
 - **Precisão Dupla:** 64 bits {**S** → 1, **E** → 11, **M** → 52}
 - **E** é representado em excesso de 1023 e **M** efetivamente possui 53 bits, um bit escondido.
 - **Precisão estendida:** 80 bits {**S** → 1, **E** → 15, **M** → 64}
 - **E** é representado em excesso de 16383 e **M** possui 64 bits, não tem bit escondido.

Representação de Dados

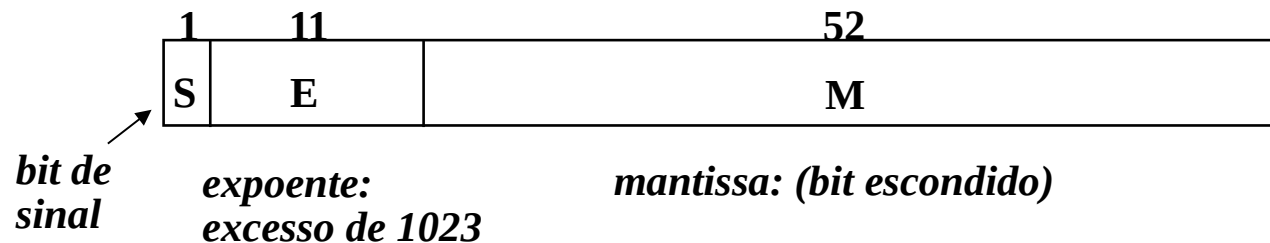
Números em Ponto Flutuante (Reais)

Formatos do Padrão IEEE 754

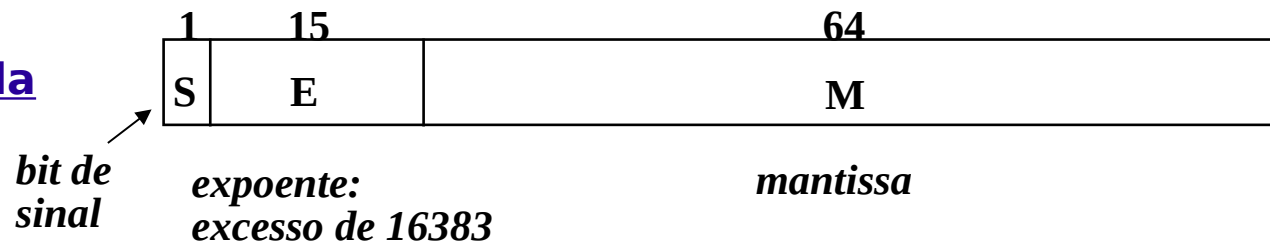
Precisão Simples



Precisão Dupla



Precisão Estendida



Representação de Dados

Números em Ponto Flutuante (Reais)

Padrão IEEE 754

- Exceções:
 - O número *0,0* é representado por 0s em todas as posições.
 - E *infinito* é representado com 1s em todas as posições do expoente e 0s em todas as posições da mantissa.

Representação de Dados

Formatos do Padrão IEEE 754

Precisão simples

Exemplo: converter o número decimal para binário

$$3,248 \times 10^4 = 32480 = 111111011100000_2 = 1,11111011100000 \times 2^{14}$$

O MSB não ocupa a posição de um bit porque ele é sempre 1

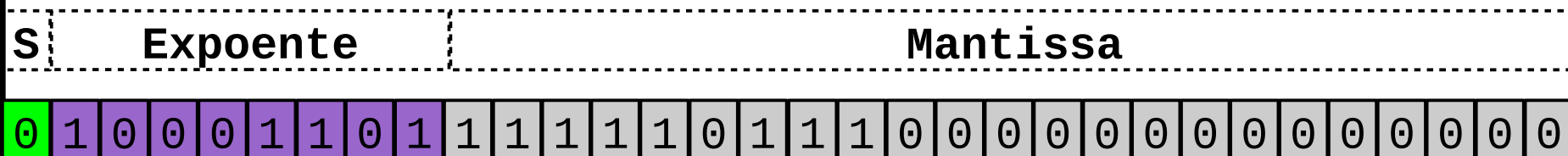
Mantissa

111110111000000000000000

Expoente (polarizado → em excesso de 127):

$$14 + 127 = 141 = 10001101_2$$

O número completo representado em ponto flutuante:



Aritmética Computacional

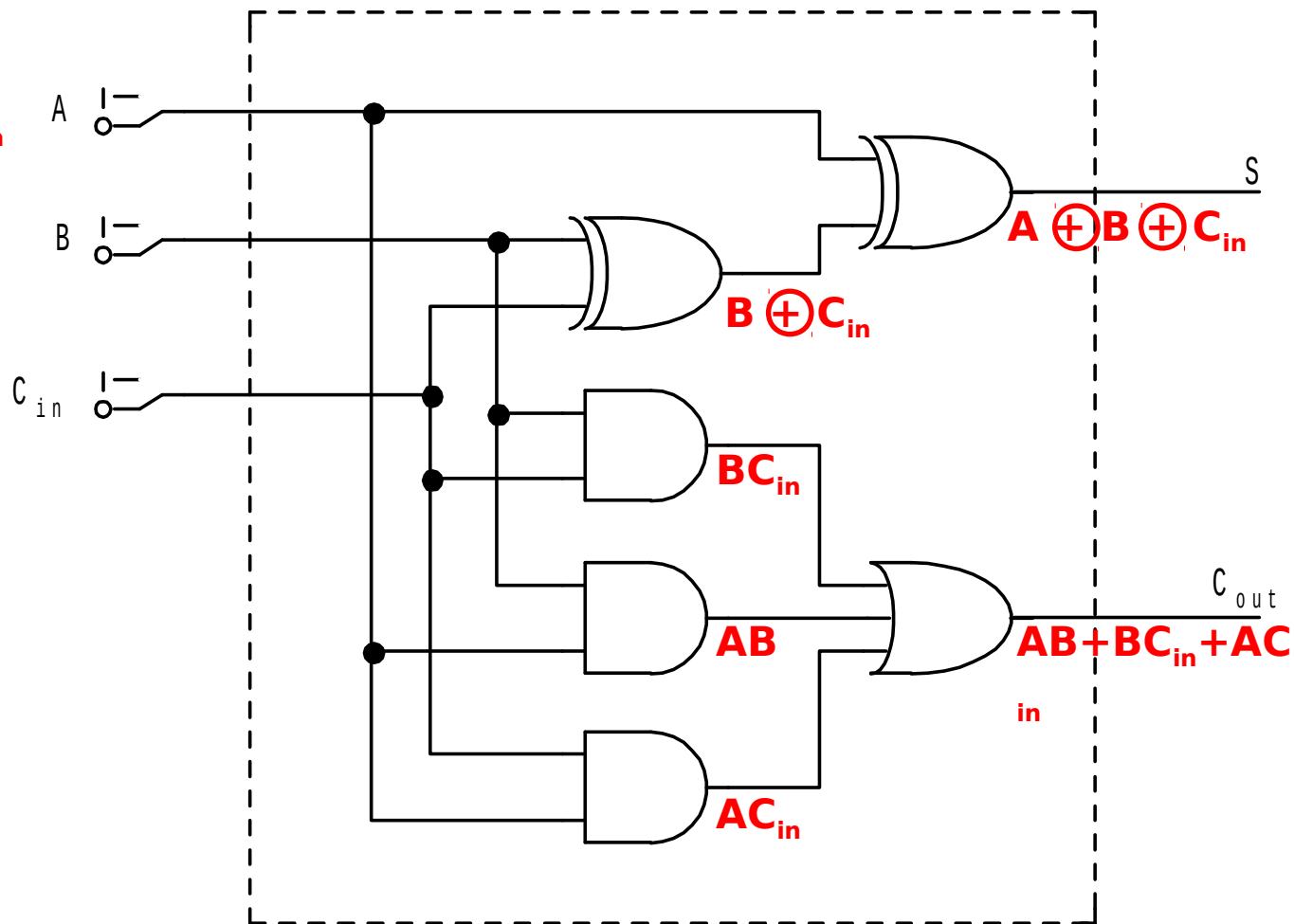
Revisão Adição e Subtração



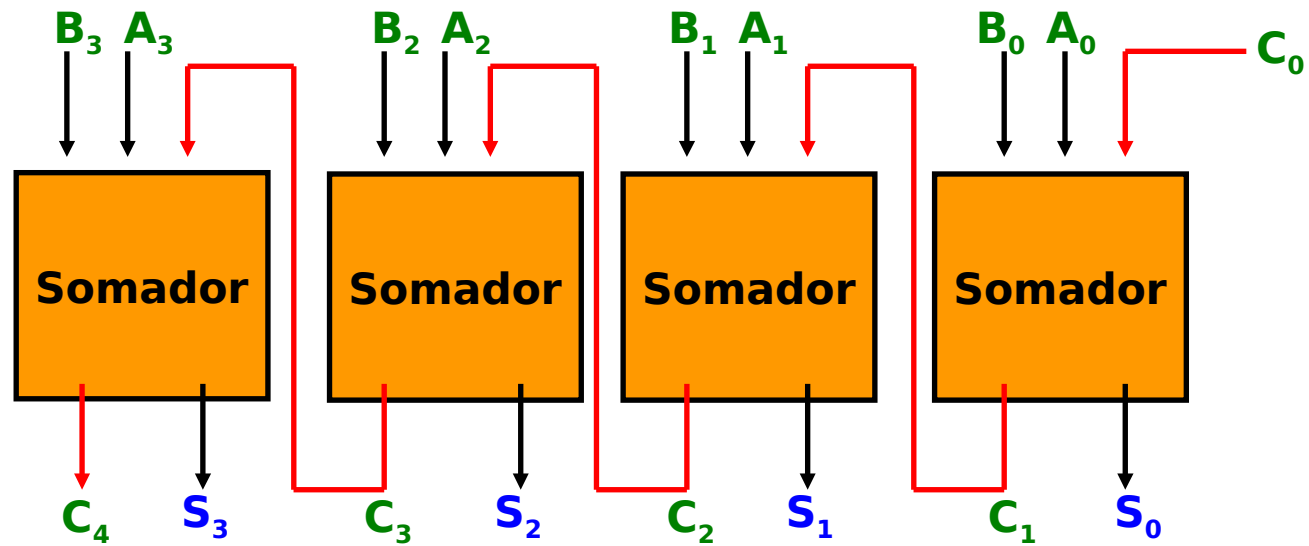
Aritmética Computacional

Circuito Somador

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + BC_{in} + AC_{in}$$



Somador de 4 bits

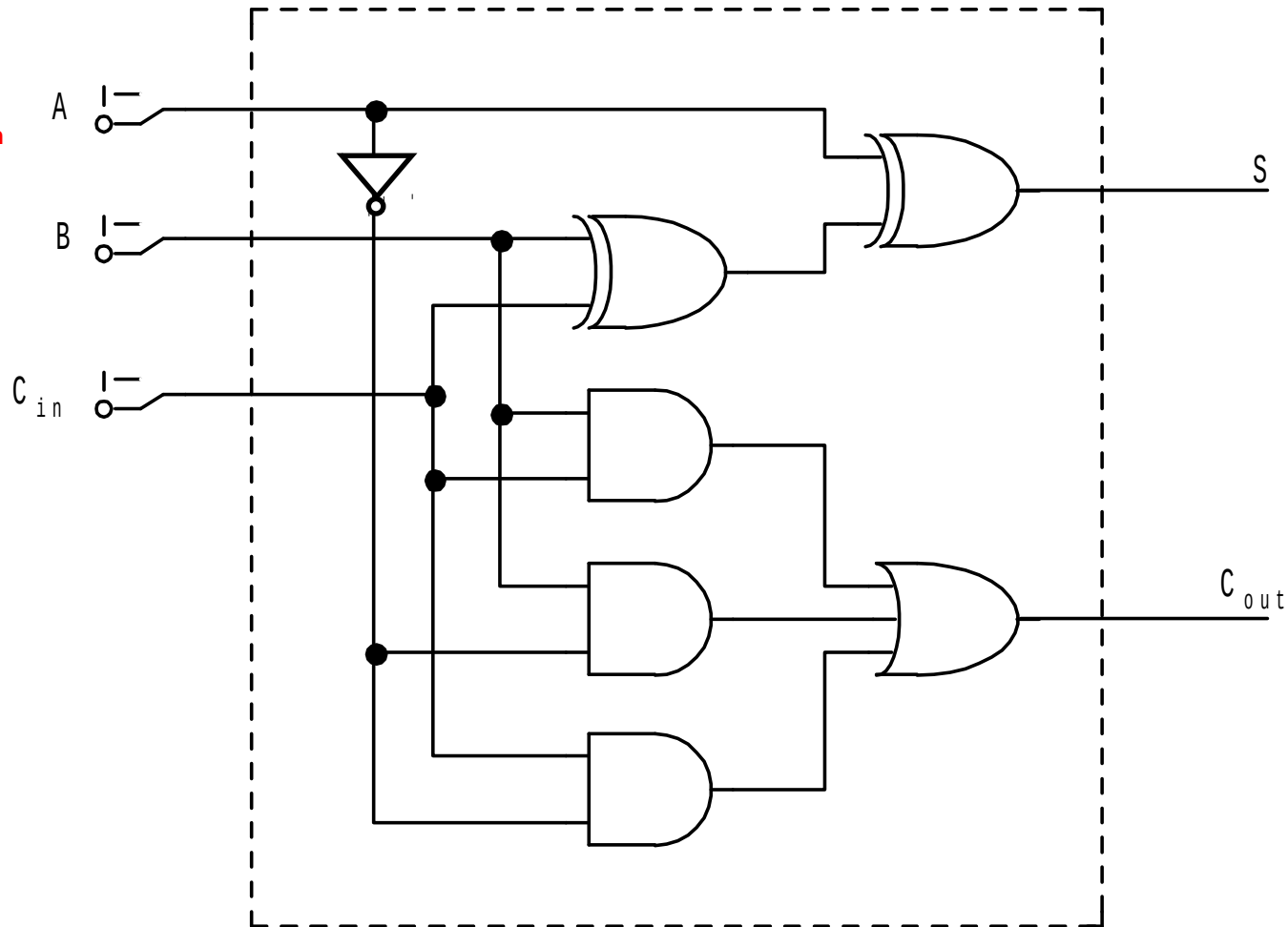


Somador Ripple-Carry

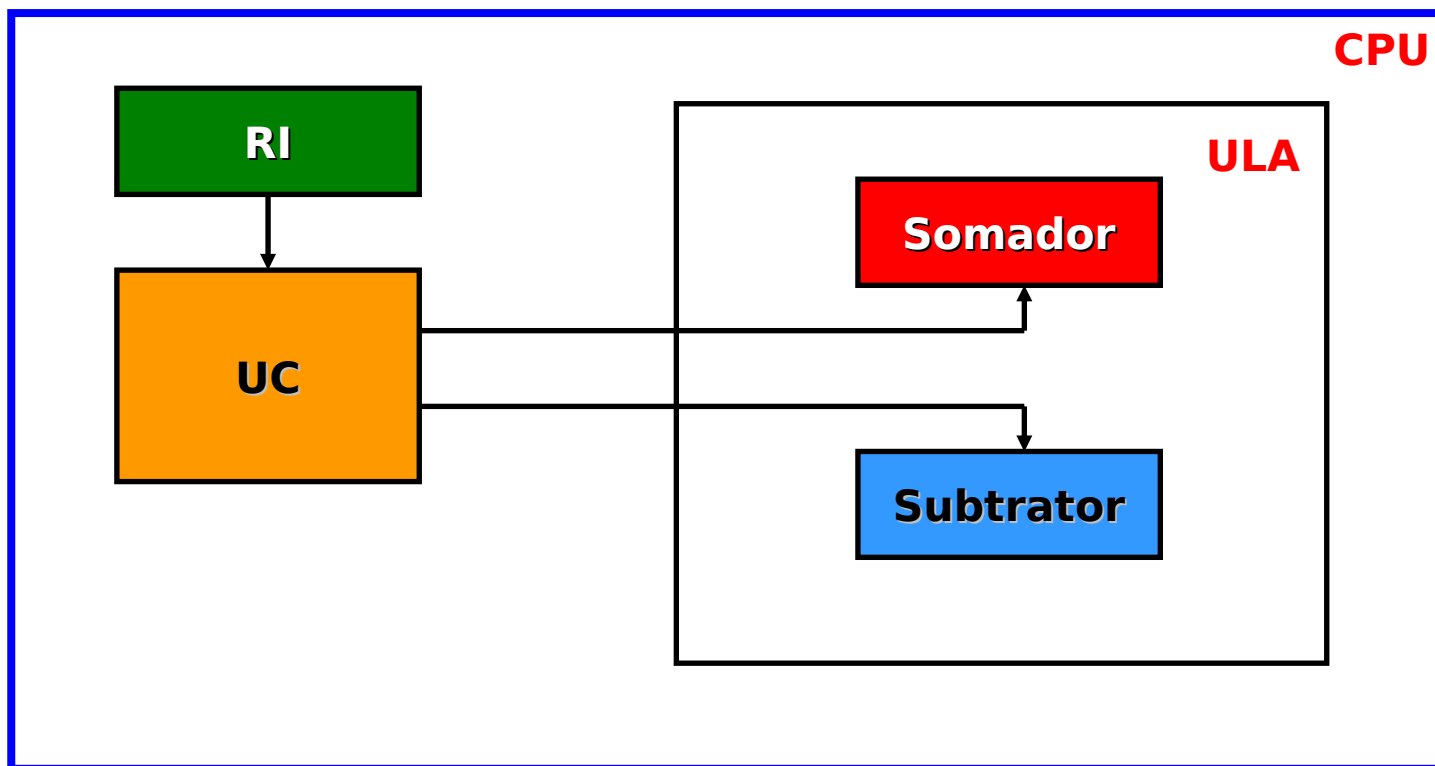
Soluções

Circuito Subtrator

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = \overline{A}B + BC_{in} + \overline{A}C_{in}$$



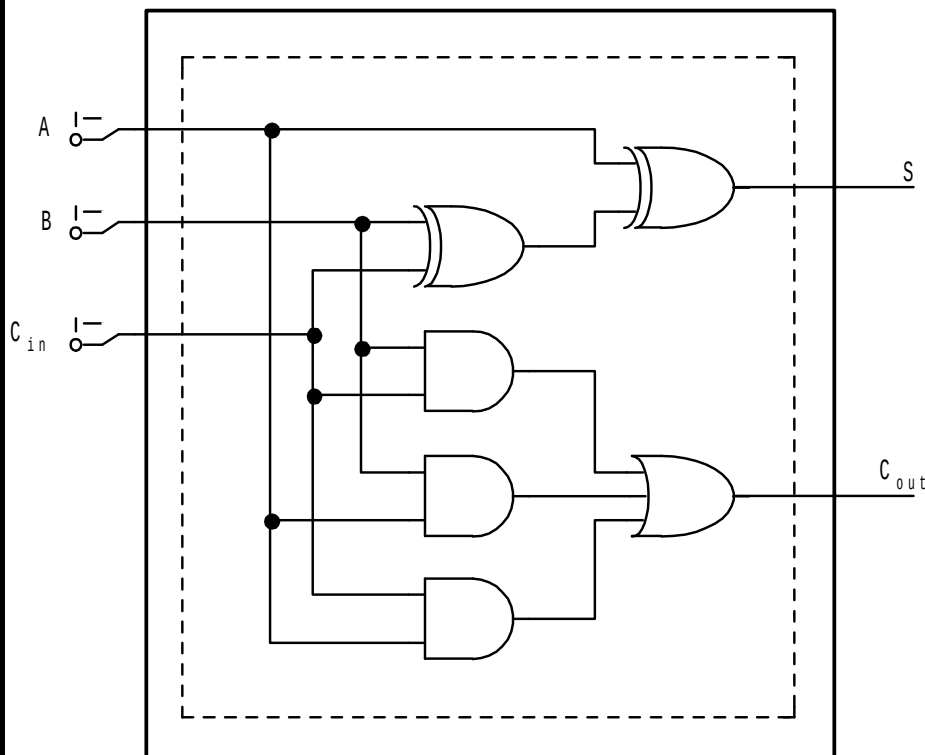
ULA: Somador e Subtrator



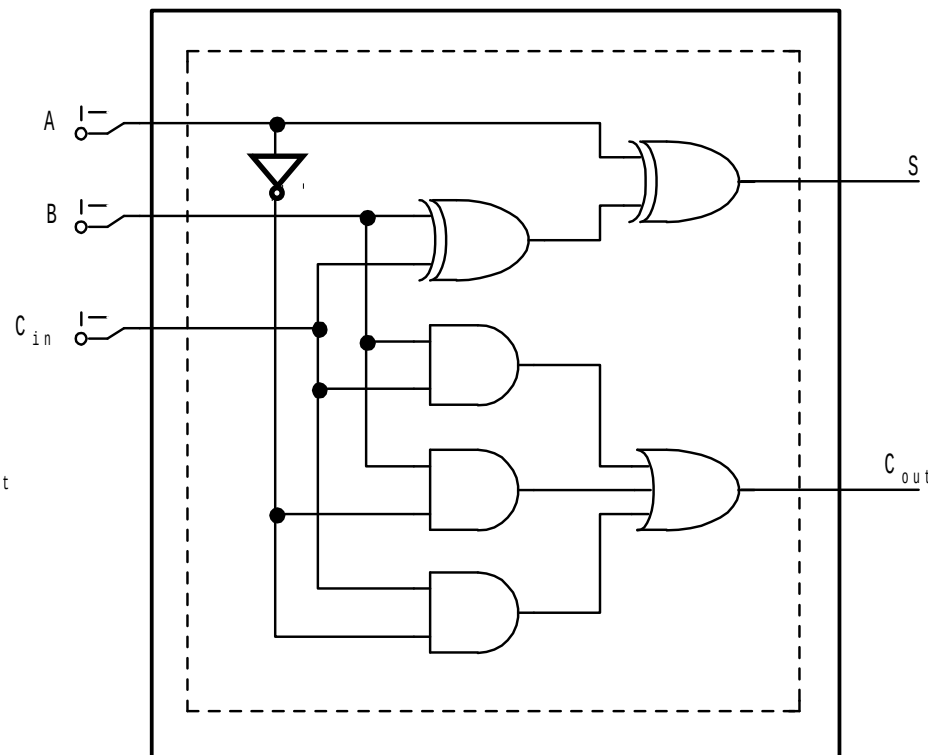
ULA com 2 circuitos para efetuar a adição e a subtração

ULA: Somador e Subtrator

Circuito Somador

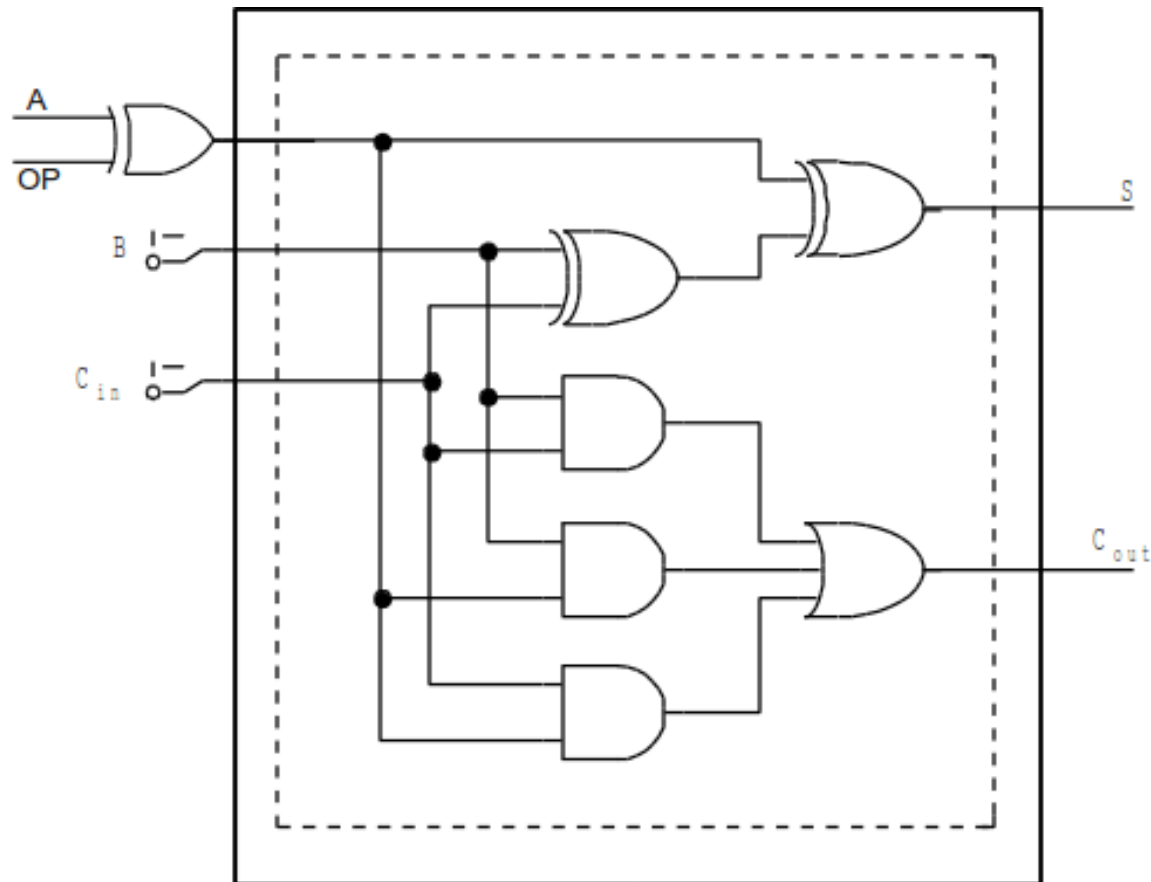


Circuito Subtrator



ULA: Somador e Subtrator

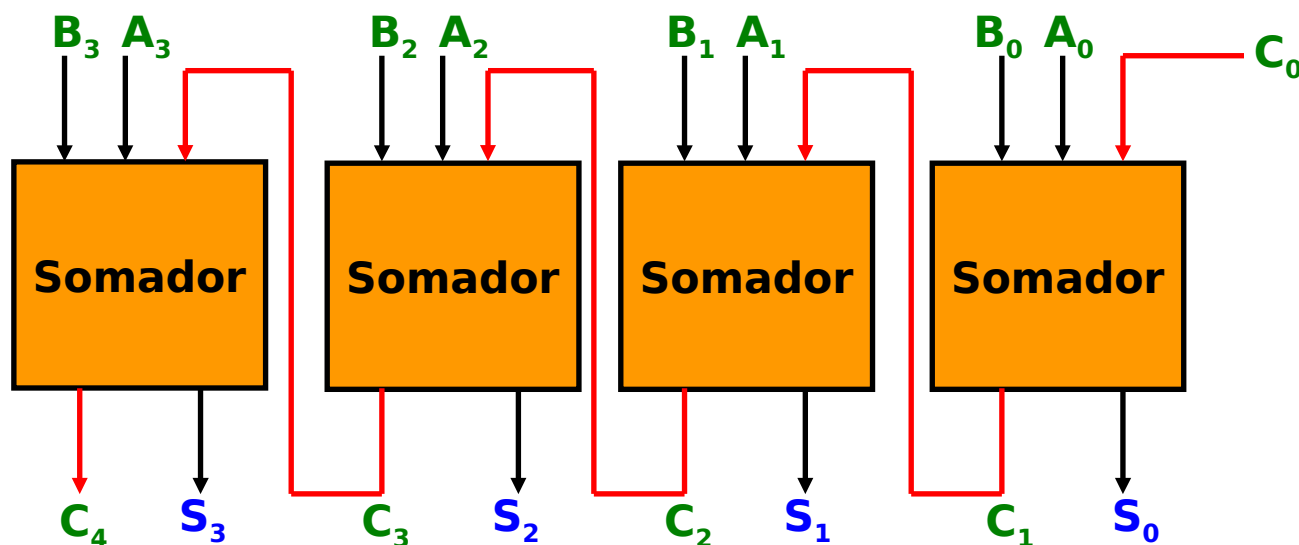
Circuito Somador/Subtrator



Somador de Alto Desempenho

Somador Convencional

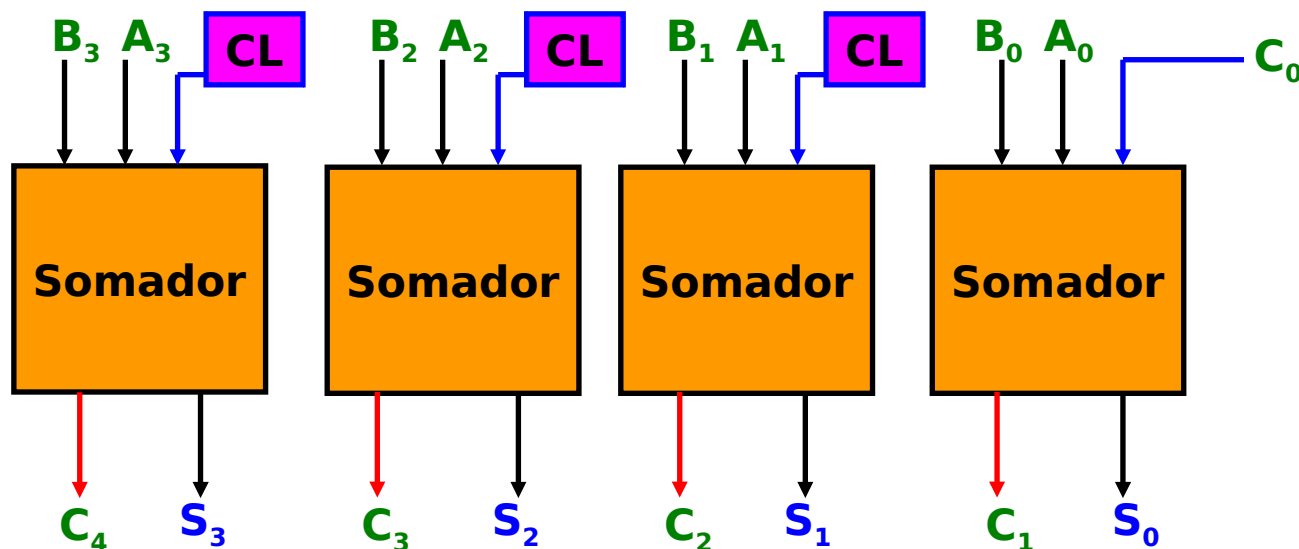
Somador Ripple-Carry



Somador Convencional: Atrasos para propagar o carry

Somador de Alto Desempenho

Somador Carry Lookahead



CL: Lógica para antecipar o carry sem passar pelo somador

Somador de Alto Desempenho

Expressão do Carry do Somador

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

1. Fatorando a expressão

$$C_{i+1} = A_i B_i + C_i (A_i + B_i)$$

2. Chamando $A_i B_i$ de G_i e $A_i + B_i$ de P_i

$$C_{i+1} = G_i + P_i C_i$$

3. Substituindo os índices para obter os carries para um somador de 4 bits

$$C_1 = G_0 + P_0 C_0$$

4. Para simplificar a análise, vamos considerar $C_0=0$ para soma

$$C_1 = G_0$$

A	B	C-out	
0	0	0	“nada”
0	1	C-in	“propaga”
1	0	C-in	“propaga”
1	1	1	“gera”

Somador de Alto Desempenho

Expressão do Carry do Somador

$$C_2 = G_1 + P_1 C_1$$

5. Substituindo $C_1 = G_0$

$$C_2 = G_1 + P_1 G_0$$

6. Obtendo C_3

$$C_3 = G_2 + P_2 C_2$$

7. Substituindo $C_2 = G_1 + P_1 G_0$

$$C_3 = G_2 + P_2 (G_1 + P_1 G_0) \Rightarrow C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0$$

8. Obtendo C_4

$$C_4 = G_3 + P_3 C_3 \Rightarrow C_4 = G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 G_0) \Rightarrow C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

Aritmética Computacional

Multiplicação e Divisão



Aritmética Computacional

Aritmética Computacional

Multiplicação em Binário:

Exemplo

a) 0 b) 0 c) 1 d) 1

$$\begin{array}{r} 0 \times \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \times \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \times \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \times \\ \hline 1 \end{array}$$

n-1 somas parciais

$$\begin{array}{r} 1101 \\ \times 1011 \\ \hline 1101 \\ 1101 \\ 0000 \\ + 1101 \\ \hline 1000111 \\ 1 \end{array}$$

Produto final tem 2n bits →

1 0 0 0 1 1 1
1

Produto

1 1 0 1 Multiplicando (M)
x 1 0 1 1 Multiplicador (Q)
Produtos Parciais

Aritmética Computacional

Aritmética Computacional

Multiplicação em Binário:

Cada dígito do Multiplicador, a partir da direita deve multiplicar o Multiplicando gerando um produto parcial

Quando o bit da direita do Multiplicador é 0 \Rightarrow Produto Parcial é 0

Quando o bit da direita do Multiplicador é 1 \Rightarrow Produto Parcial é o próprio Multiplicando

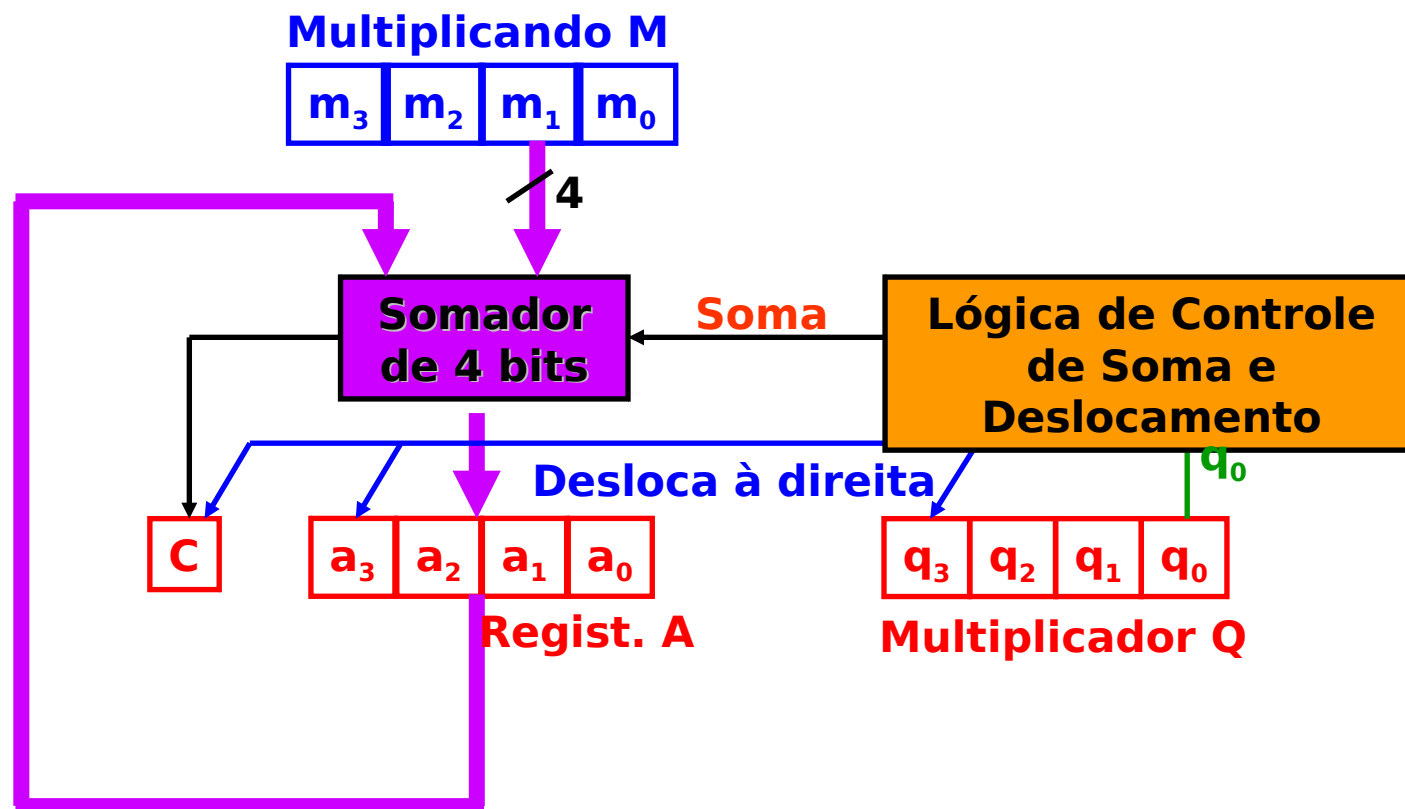
Cada Produto Parcial é deslocado 1 bit à esquerda em relação ao produto parcial anterior

$$\begin{array}{r} 1101 \text{ Multiplicando (M)} \\ \times 1011 \text{ Multiplicador (Q)} \\ \hline 1101 \\ 1101 \\ 0000 \\ + 1101 \\ \hline 10001111 \text{ Produto} \end{array}$$

Produtos Parciais

Aritmética Computacional

Multiplicação Binária

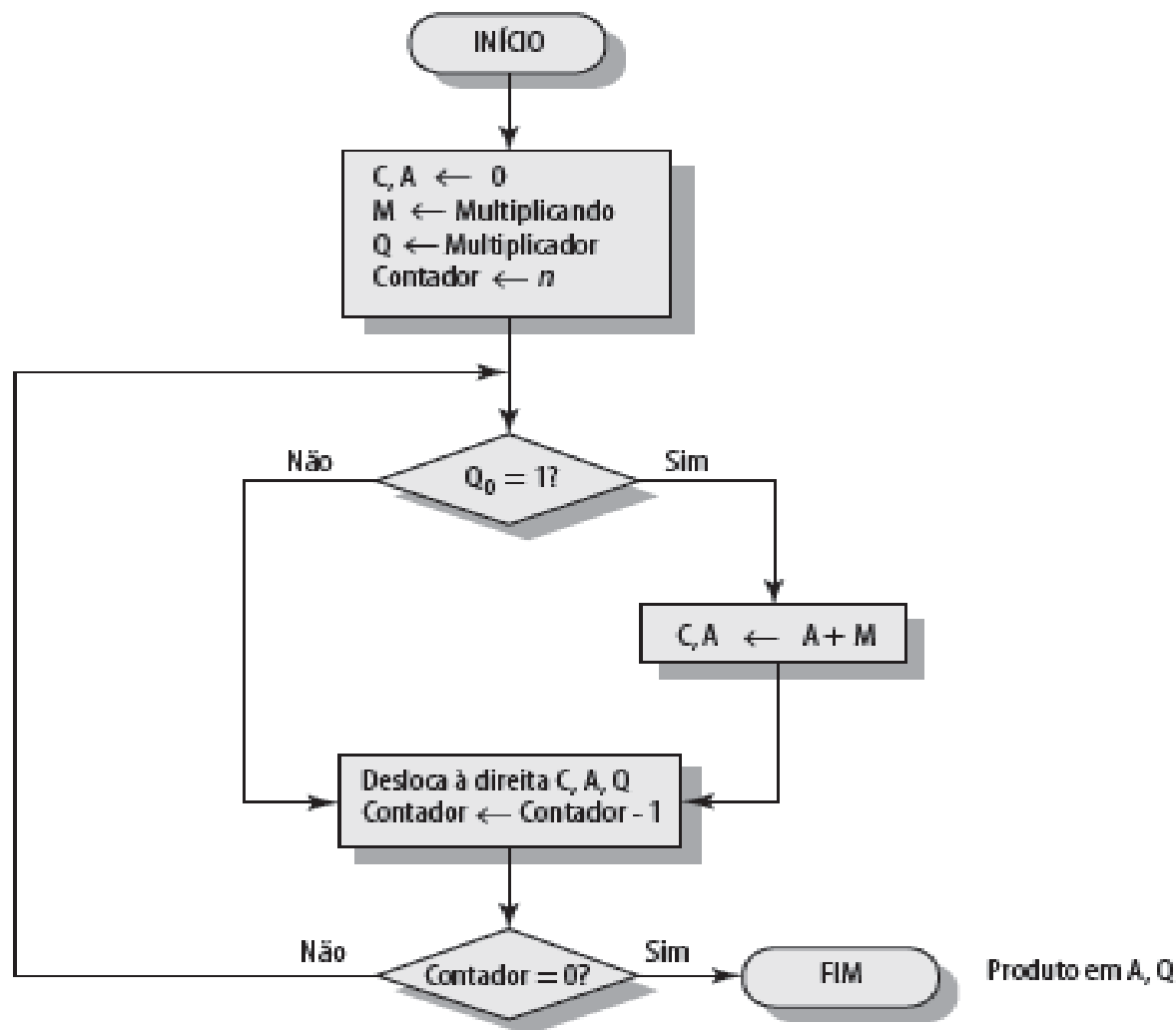


Aritmética Computacional

Algoritmo de Multiplicação

1. Registrador M \leftarrow Multiplicando
2. Registrador Q \leftarrow Multiplicador
3. Registrador A \leftarrow 0
4. Registrador C \leftarrow 0
5. Bit q_0 do Multiplicador é testado
Se $q_0=0$ {
 - Não soma M e A (o produto parcial não é somado ao Multiplicando)
 - Desloca os registradores C/A/Q para a direita}
- Se $q_0=1$ {
 - Soma M e A resultado fica em A)
 - Desloca os registradores C/A/Q para a direita}
6. Repete o passo 5, n vezes ($n = n^\circ$ de bits de M e Q)
7. Produto (resultado final) está armazenado em A e Q

Aritmética Computacional



Aritmética Computacional

Multiplicação Binária

Exemplo:

			1101
			<u>x1011</u>
	1 1 0 1 ← Multiplicando (M)		
0	0 0 0 0	1 0 1 1 ← Multiplicador (Q)	
C	A	Q	
0	0 0 0 0	1 0 1 1	$q_0=1 \Rightarrow A+M$
C	A	Q	

Aritmética Computacional

Multiplicação Binária

Exemplo:

1101

x1011

1 1 0 1 Multiplicando (M)

$n=1$ $\left\{ \begin{array}{l} C \\ 0 \end{array} \right.$

A
1 1 0 1

Q
1 0 1 1 $q_0=1 \Rightarrow A+M$

Aritmética Computacional

Multiplicação Binária

1101

x1011

Exemplo:

1 1 0 1 Multiplicando (M)

n=1 { C
0
0

A
1 1 0 1
0 1 1 0

Q
1 0 1 1
1 1 0 1 desloca para a direita

Aritmética Computacional

Multiplicação Binária

1101

x1011

Exemplo:

1 1 0 1 Multiplicando (M)

	C	A	Q
n=1 {	0	1 1 0 1	1 0 1 1
	0	0 1 1 0	1 1 0 1
<hr/>			
n=2 {	1	0 0 1 1	1 1 0 1
<hr/>			

$q_0 = 1 \Rightarrow A + M$

Aritmética Computacional

Multiplicação Binária

1101

x1011

Exemplo:

1 1 0 1 Multiplicando (M)

	C	A	Q
n=1 {	0	1 1 0 1	1 0 1 1
	0	0 1 1 0	1 1 0 1
<hr/>			
n=2 {	1	0 0 1 1	1 1 0 1
	0	1 0 0 1	1 1 1 0
<hr/>			

desloca para a direita

Aritmética Computacional

Multiplicação Binária

Exemplo:

1 1 0 1 Multiplicando (M)

1101

x1011

	C	A	Q
n=1	0	1 1 0 1	1 0 1 1
	0	0 1 1 0	1 1 0 1
<hr/>			
n=2	1	0 0 1 1	1 1 0 1
	0	1 0 0 1	1 1 1 0
<hr/>			

q₀=0 ⇒ Não soma A+M, desloca

Aritmética Computacional

Multiplicação Binária

1101

x1011

Exemplo:

1 1 0 1 Multiplicando (M)

	C	A	Q
n=1	0	1 1 0 1	1 0 1 1
	0	0 1 1 0	1 1 0 1
n=2	1	0 0 1 1	1 1 0 1
	0	1 0 0 1	1 1 1 0
n=3	0	0 1 0 0	1 1 1 1
			desloca para a direita

Aritmética Computacional

Multiplicação Binária

1101

x1011

Exemplo:

1 1 0 1 Multiplicando (M)

	C	A	Q
n=1	0	1 1 0 1	1 0 1 1
	0	0 1 1 0	1 1 0 1
n=2	1	0 0 1 1	1 1 0 1
	0	1 0 0 1	1 1 1 0
n=3	0	0 1 0 0	1 1 1 1
			1 $q_0=1 \Rightarrow A+M$
n=4	1	0 0 0 1	1 1 1 1

Aritmética Computacional

Multiplicação Binária

1101

x1011

Exemplo:

1 1 0 1 Multiplicando (M)

	C	A	Q
n=1	0	1 1 0 1	1 0 1 1
	0	0 1 1 0	1 1 0 1
n=2	1	0 0 1 1	1 1 0 1
	0	1 0 0 1	1 1 1 0
n=3	0	0 1 0 0	1 1 1 1
n=4	1	0 0 0 1	1 1 1 1
	0	1 0 0 0	1 1 1 1

desloca para a direita

Produto

Exercício

Faça a multiplicação de $0010_2 (M) \times 0011_2 (Q)$

Solução

Algoritmo de Multiplicação

1. Registrador M \Leftarrow Multiplicando
2. Registrador Q \Leftarrow Multiplicador
3. Registrador A \Leftarrow 0
4. Registrador C \Leftarrow 0
5. Bit q_0 do Multiplicador é testado
Se $q_0=0$ {
 - Não soma M e A (o produto parcial não é somado ao Multiplicando)
 - Desloca os registradores C/A/Q para a direita}
- Se $q_0=1$ {
 - Soma M e A resultado fica em A)
 - Desloca os registradores C/A/Q para a direita}
6. Repete o passo 5, n vezes ($n = n^\circ$ de bits de M e Q)
7. Produto (resultado final) está armazenado em A e Q

Solução

Multiplicação Binária

0010

x0011

0 0 1 0 ← Multiplicando (M)

0

C

0 0 0 0

A

0 0 1 1

Q

← Multiplicador (Q)

0

C

0 0 0 0

A

0 0 1 **1** $q_0=1 \Rightarrow A+M$

Q

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

$n=1$ $\left\{ \begin{array}{l} C \\ 0 \end{array} \right.$

A
0 0 1 0

Q
0 0 1 1 $q_0=1 \Rightarrow A+M$

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

$n=1$ { C
0
0

A
0 0 1 0
0 0 0 1

Q
0 0 1 1
0 0 0 1 desloca para a direita

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

	C	A	Q
n=1 {	0	0 0 1 0	0 0 1 1
	0	0 0 0 1	0 0 0 1
<hr/>			
n=2 {	0	0 0 1 1	0 0 0 1
<hr/>			

$q_0 = 1 \Rightarrow A + M$

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

	C	A	Q
n=1 {	0	0 0 1 0	0 0 1 1
	0	0 0 0 1	0 0 0 1
<hr/>			
n=2 {	0	0 0 1 1	0 0 0 1
	0	0 0 0 1	1 0 0 0 desloca para a direita
<hr/>			

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

	C	A	Q
n=1	0	0 0 1 0	0 0 1 1
	0	0 0 0 1	0 0 0 1
<hr/>			
n=2	0	0 0 1 1	0 0 0 1
	0	0 0 0 1	1 0 0 0
<hr/>			

q₀=0 ⇒ Não soma A+M, desloca

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

	C	A	Q
n=1	0	0 0 1 0	0 0 1 1
	0	0 0 0 1	0 0 0 1
n=2	0	0 0 1 1	0 0 0 1
	0	0 0 0 1	1 0 0 0
n=3	0	0 0 0 0	1 1 0 0desloca para a direita

Solução

Multiplicação Binária

0 0 1 0 Multiplicando (M)

	C	A	Q
n=1	0	0 0 1 0	0 0 1 1
	0	0 0 0 1	0 0 0 1
n=2	0	0 0 1 1	0 0 0 1
	0	0 0 0 1	1 0 0 0
n=3	0	0 0 0 0	1 1 0 0
n=4	0	0 0 0 0	0 1 1 0

Produto

q₀=0 ⇒ Não soma A+M, desloca

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

Dividendo (Q)

1 0 0 1 0 1 0

Divisor (M)

1 0 0 0

Quociente

Resto

Aritmética Computacional

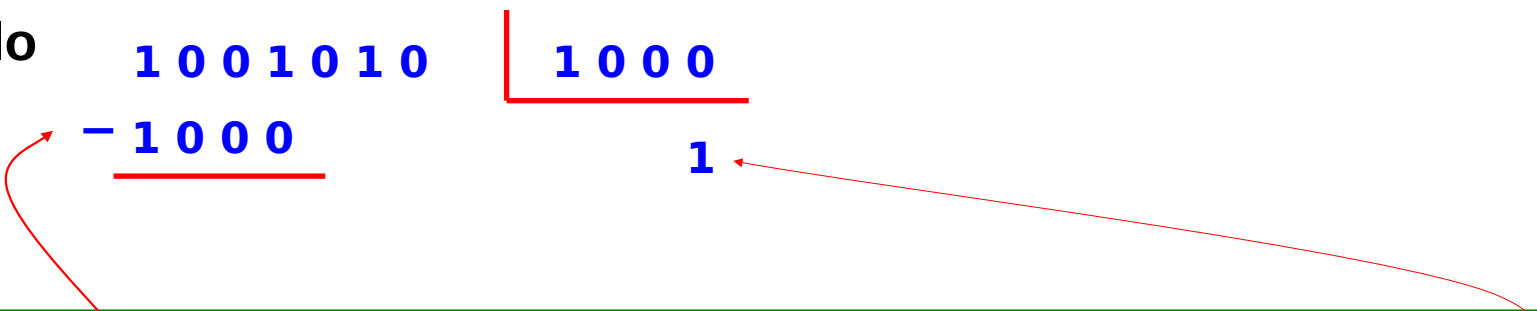
Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline \end{array}$$

1



Divisão: - Verifica quantas vezes o divisor pode ser subtraído do dividendo
- Para cada tentativa de subtração insere 1 dígito no quociente

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline 1 \end{array} \quad \begin{array}{r} 1000 \\ \hline 1 \end{array}$$

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline 10 \end{array} \quad \begin{array}{r} 1000 \\ \hline 1 \end{array}$$

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline 101 \end{array}$$
$$\begin{array}{r} 1000 \\ \hline 10 \end{array}$$

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline 1010 \end{array} \quad \begin{array}{r} 1000 \\ \hline 100 \end{array}$$

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline 1010 \\ - 1000 \\ \hline \end{array} \quad \begin{array}{r} 1000 \\ \hline 1001 \end{array}$$

Aritmética Computacional

Aritmética Computacional

Divisão:

Exemplo

$$\begin{array}{r} 1001010 \\ - 1000 \\ \hline 1010 \\ - 1000 \\ \hline 10 \end{array}$$

1000 1001 ← Quociente

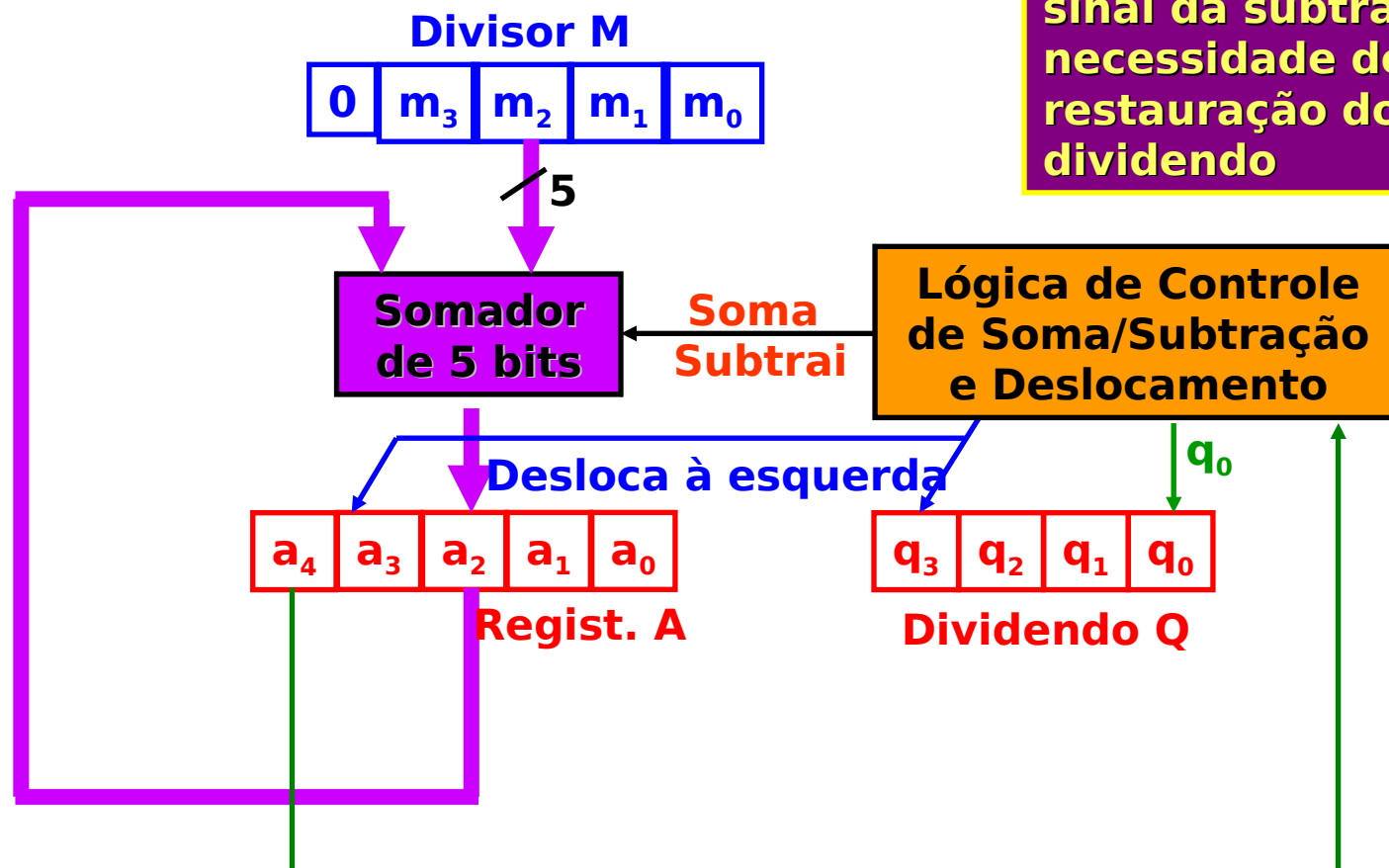
10 ← Resto

- Para verificar se o divisor “cabe” no dividendo faz-se uma subtração do Divisor.
- Se o resultado é negativo significa que ainda não dá para dividir, então restaura-se o valor do dividendo e insere 0 no quociente.
- Se o resultado é positivo significa que dá para dividir, então insere-se 1 no quociente.

Aritmética Computacional

Divisão Binária

Usa registradores de 5 bits para detectar o sinal da subtração e a necessidade de restauração do dividendo

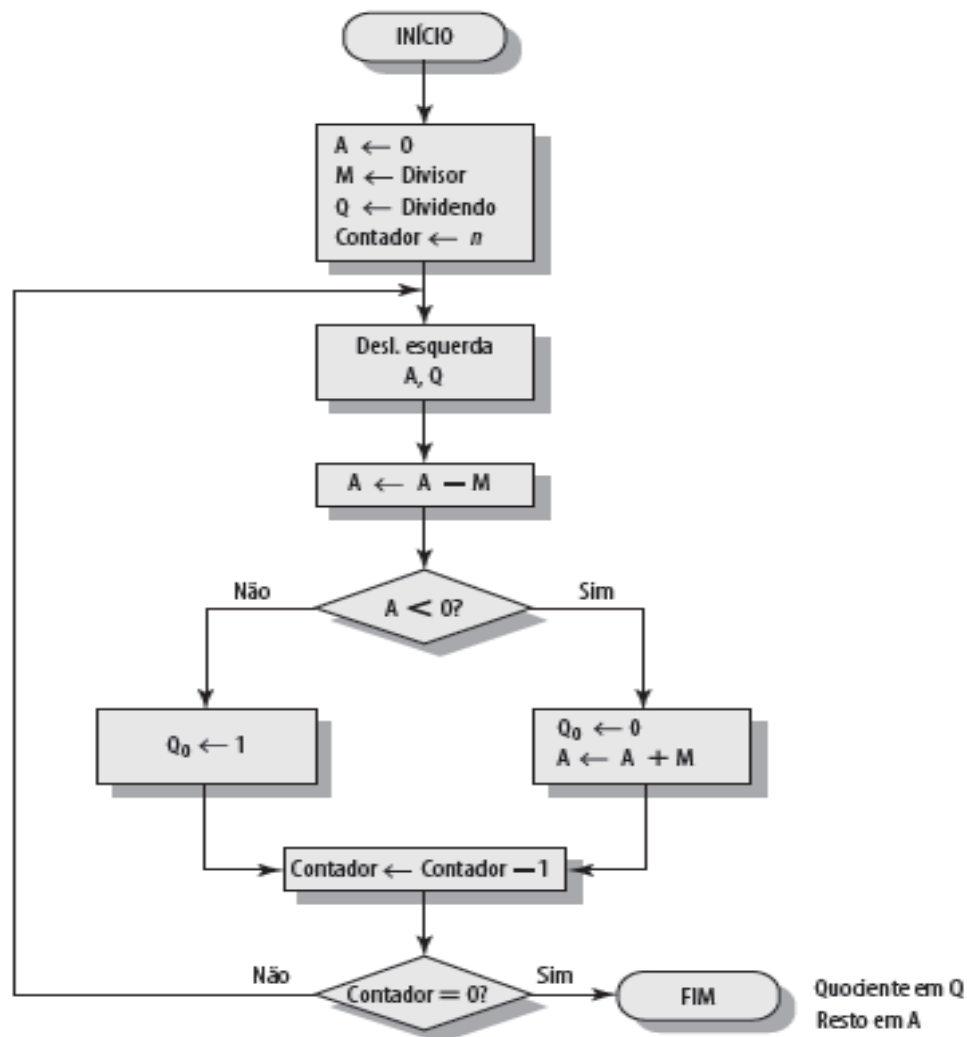


Aritmética Computacional

Algoritmo de Divisão

1. Registrador Q \leftarrow Dividendo
2. Registrador M \leftarrow Divisor
3. Registrador A \leftarrow 0
4. Bit mais significativo de M (m_4) é zerado
5. Desloca os Registradores A e Q para a esquerda
6. Subtrai A-M para saber se o divisor “cabe” no dividendo
7. Bit a_4 do Dividendo (Registrador A) é testado
Se $a_4=0$ {
 - Não soma A e M
 - $q_0 \leftarrow 1$ significa que é possível subtrair o divisor do dividendo}
- Se $a_4=1$ {
 - Soma A e M para restaurar o dividendo
 - $q_0 \leftarrow 0$ (significa que ainda não é possível fazer a divisão)}
8. Repete os passos 5 a 7, n vezes ($n = n^\circ$ de bits de M e Q)
9. Quociente está armazenado em Q e o resto em A

Aritmética Computacional



Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1

← Dividendo (Q)

Q

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

0 1 1 1

Q

1 1 1 0

← Dividendo (Q)

desloca A e Q para a esquerda

n=1

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 0 1

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0

1 1 1 0

desloca A e Q para a esquerda

A-M (Compl. 2 de M e soma)

n=1 {

$a_4=1$ →

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 0 1

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

n=1 {

$a_4=1$ →

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 0 1

0 0 0 0 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

$n=1$ {

$a_4=1$ →

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 0 1

0 0 0 0 0

0 0 0 0 0

0 0 0 0 1

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

1 1 0 0 desloca A e Q para a esquerda

n=1

$a_4=1$

n=2

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 0 1

0 0 0 0 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

0 0 0 0 1

1 1 1 1 0

1 1 0 0 desloca A e Q para a esquerda

1 1 0 0 A-M (Compl. 2 de M e soma)

n=1

$a_4=1$

n=2

$a_4=1$

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 0 1

0 0 0 0 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

0 0 0 0 1

1 1 1 1 0

0 0 0 0 1

1 1 0 0 desloca A e Q para a esquerda

1 1 0 0 A-M (Compl. 2 de M e soma)

1 1 0 0 A+M (restaura divisor)

n=1

$a_4=1$

n=2

$a_4=1$

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1 ← Dividendo (Q)

Q

n=1 {

$a_4=1$ →

0 0 0 0 0

1 1 1 0 1

0 0 0 0 0

0 0 0 0 0

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

n=2 {

$a_4=1$ →

0 0 0 0 1

1 1 1 1 0

0 0 0 0 1

0 0 0 0 1

1 1 0 0 desloca A e Q para a esquerda

1 1 0 0 A-M (Compl. 2 de M e soma)

1 1 0 0 A+M (restaura divisor)

1 1 0 0 Zera q_0 ($q_0 \leftarrow 0$)

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 1 1 1

Q

1 0 0 0

← Dividendo (Q)

desloca A e Q para a esquerda

n=3

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 0 0 0

1 0 0 0

desloca A e Q para a esquerda

A-M (Compl. 2 de M e soma)

n=3 {

$a_4=0$ →

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

1 0 0 1 $q_0 \leftarrow 1$

$n=3$ {

$a_4=0$ →

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

1 0 0 1 $q_0 \leftarrow 1$

$n=3$ {

$a_4=0$ →

$n=4$ {

0 0 0 0 1

0 0 1 0 desloca A e Q para a esquerda

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

n=3 {

$a_4=0$ →

0 1 1 1

Q

1 0 0 0

1 0 0 0

1 0 0 1

← Dividendo (Q)

desloca A e Q para a esquerda

A-M (Compl. 2 de M e soma)

$q_0 \leftarrow 1$

n=4 {

$a_4=1$ →

0 0 0 0 1

1 1 1 1 0

0 0 1 0

0 0 1 0

desloca A e Q para a esquerda

A-M (Compl. 2 de M e soma)

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

n=3 {

$a_4=0$ →

0 1 1 1

Q

1 0 0 0

1 0 0 0

1 0 0 1

← Dividendo (Q)

desloca A e Q para a esquerda

A-M (Compl. 2 de M e soma)

$q_0 \leftarrow 1$

n=4 {

$a_4=1$ →

0 0 0 0 1

1 1 1 1 0

0 0 0 0 1

0 0 1 0

0 0 1 0

0 0 1 0

desloca A e Q para a esquerda

A-M (Compl. 2 de M e soma)

A+M (restaura divisor)

Aritmética Computacional

Divisão Binária

Exemplo:

0 0 0 1 1 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

n=3

$a_4=0$

0 1 1 1

← Dividendo (Q)

Q

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

1 0 0 1 $q_0 \leftarrow 1$

n=4

$a_4=1$

0 0 0 0 1

1 1 1 1 0

0 0 0 0 1

0 0 0 0 1

0 0 1 0 desloca A e Q para a esquerda

0 0 1 0 A-M (Compl. 2 de M e soma)

0 0 1 0 A+M (restaura divisor)

0 0 1 0 Zera q_0 ($q_0 \leftarrow 0$)

Resto

Quociente

Exercício

Faça a divisão $0111_2 (Q) : 0010_2 (M)$

Aritmética Computacional

Algoritmo de Divisão

1. Registrador Q \leftarrow Dividendo
2. Registrador M \leftarrow Divisor
3. Registrador A \leftarrow 0
4. Bit mais significativo de M (m_4) é zerado
5. Desloca os Registradores A e Q para a esquerda
6. Subtrai A-M para saber se o divisor “cabe” no dividendo
7. Bit a_4 do Dividendo (Registrador A) é testado
 - Se $a_4=0$ {
 - Não soma A e M
 - $q_0 \leftarrow 1$ significa que é possível subtrair o divisor do dividendo}
 - Se $a_4=1$ {
 - Soma A e M para restaurar o dividendo
 - $q_0 \leftarrow 0$ (significa que ainda não é possível fazer a divisão)}
8. Repete os passos 5 a 7, n vezes ($n = n^\circ$ de bits de M e Q)
9. Quociente está armazenado em Q e o resto em A

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1

← Dividendo (Q)

Q

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

0 1 1 1

Q

1 1 1 0

← Dividendo (Q)

desloca A e Q para a esquerda

n=1

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 1 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

n=1 {

$a_4=1$ →

1

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 1 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

n=1 {

$a_4=1$ →

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 0 0

1 1 1 1 0

0 0 0 0 0

0 0 0 0 0

0 1 1 1

← Dividendo (Q)

Q

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

n=1 {

$a_4=1$ →

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1 ← Dividendo (Q)

Q

n=1 {

$a_4 = 1$ →

0 0 0 0 0

1 1 1 1 0

0 0 0 0 0

0 0 0 0 0

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

n=2 {

0 0 0 0 1

1 1 0 0 desloca A e Q para a esquerda

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1

Q

← Dividendo (Q)

n=1 {

$a_4=1$ →

0 0 0 0 0

1 1 1 1 0

0 0 0 0 0

0 0 0 0 0

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

n=2 {

$a_4=1$ →

0 0 0 0 1

1 1 1 1 1

1 1 0 0 desloca A e Q para a esquerda

1 1 0 0 A-M (Compl. 2 de M e soma)

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1 ← Dividendo (Q)

Q

n=1 {

$a_4=1$ →

0 0 0 0 0

1 1 1 1 0

0 0 0 0 0

0 0 0 0 0

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

n=2 {

$a_4=1$ →

0 0 0 0 1

1 1 1 1 1

0 0 0 0 1

1 1 0 0 desloca A e Q para a esquerda

1 1 0 0 A-M (Compl. 2 de M e soma)

1 1 0 0 A+M (restaura divisor)

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1 ← Dividendo (Q)

Q

n=1 {

$a_4=1$ →

0 0 0 0 0

1 1 1 1 0

0 0 0 0 0

0 0 0 0 0

1 1 1 0 desloca A e Q para a esquerda

1 1 1 0 A-M (Compl. 2 de M e soma)

1 1 1 0 A+M (restaura divisor)

1 1 1 0 Zera q_0 ($q_0 \leftarrow 0$)

n=2 {

$a_4=1$ →

0 0 0 0 1

1 1 1 1 1

0 0 0 0 1

0 0 0 0 1

1 1 0 0 desloca A e Q para a esquerda

1 1 0 0 A-M (Compl. 2 de M e soma)

1 1 0 0 A+M (restaura divisor)

1 1 0 0 Zera q_0 ($q_0 \leftarrow 0$)

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 1 1 1

Q

1 0 0 0

← Dividendo (Q)

desloca A e Q para a esquerda

n=3

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 1

0 1 1 1

← Dividendo (Q)

Q

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

n=3 {

$a_4=0$ →

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 0 0 1 1

0 0 0 0 1

0 0 0 0 1

0 1 1 1

← Dividendo (Q)

Q

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

1 0 0 1 $q_0 \leftarrow 1$

$n=3$ {

$a_4=0$ →

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1

Q

← Dividendo (Q)

n=3 {

$a_4 = 0$ →

0 0 0 1 1

0 0 0 0 1

0 0 0 0 1

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

1 0 0 1 $q_0 \leftarrow 1$

n=4 {

0 0 0 1 1

0 0 1 0 desloca A e Q para a esquerda

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1

Q

← Dividendo (Q)

n=3 {

$a_4=0$ →

0 0 0 1 1

0 0 0 0 1

0 0 0 0 1

1 0 0 0 desloca A e Q para a esquerda

1 0 0 0 A-M (Compl. 2 de M e soma)

1 0 0 1 $q_0 \leftarrow 1$

n=4 {

$a_4=0$ →

0 0 0 1 1

0 0 0 0 1

0 0 1 0 desloca A e Q para a esquerda

0 0 1 0 A-M (Compl. 2 de M e soma)

Solução

Divisão Binária

0 0 0 1 0 ← Divisor (M)

0 0 0 0 0

A

0 1 1 1

Q

← Dividendo (Q)

n=3 {

$a_4=0$ →
0 0 0 1 1
0 0 0 0 1
0 0 0 0 1

1 0 0 0 desloca A e Q para a esquerda
1 0 0 0 A-M (Compl. 2 de M e soma)
1 0 0 1 $q_0 \leftarrow 1$

n=4 {

$a_4=0$ →
0 0 0 1 1
0 0 0 0 1
0 0 0 0 1

0 0 1 0 desloca A e Q para a esquerda
0 0 1 0 A-M (Compl. 2 de M e soma)
0 0 1 1 $q_0 \leftarrow 1$

Resto

Quociente

Exercícios

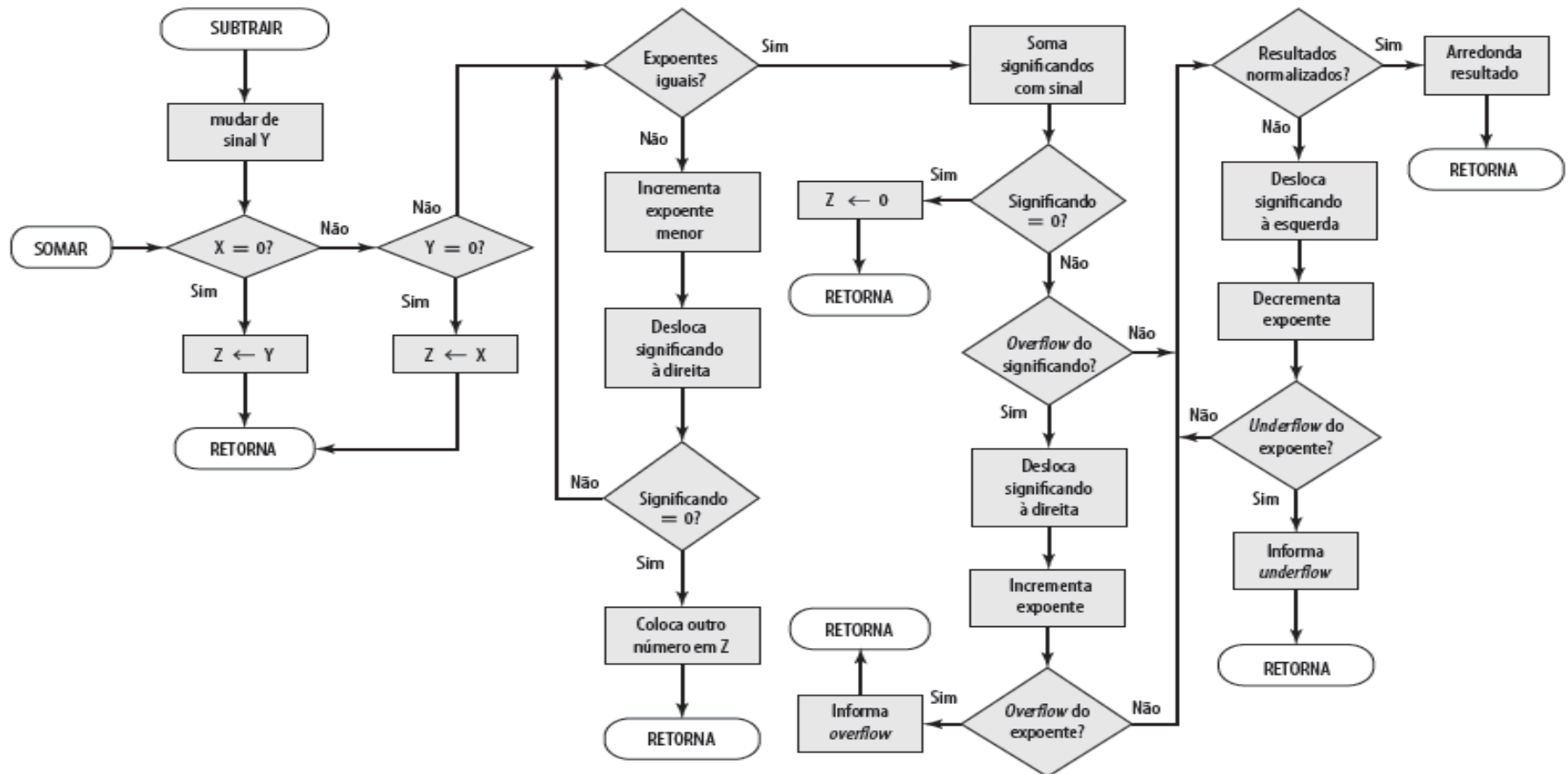
Faça a multiplicação de $0110_2 (M) \times 0011_2 (Q)$

Faça a divisão de $1110_2 (Q) : 0110_2 (M)$

Aritmética de ponto flutuante (+/-)

- Verifique zero.
- Alinhe significandos (ajustando expoentes).
- Soma ou subtraia significandos.
- Normalize resultado.

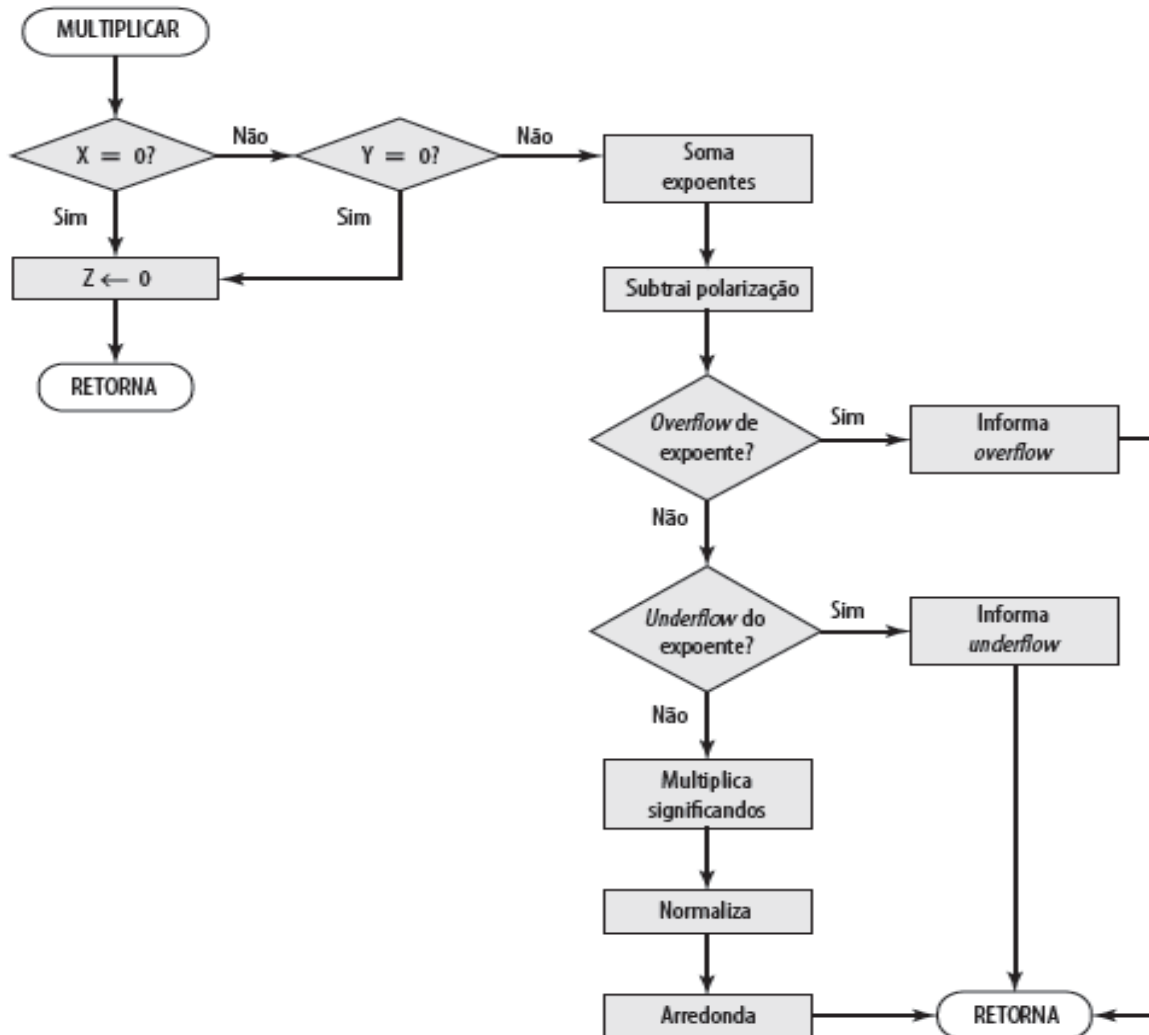
Fluxograma da adição e subtração de ponto flutuante



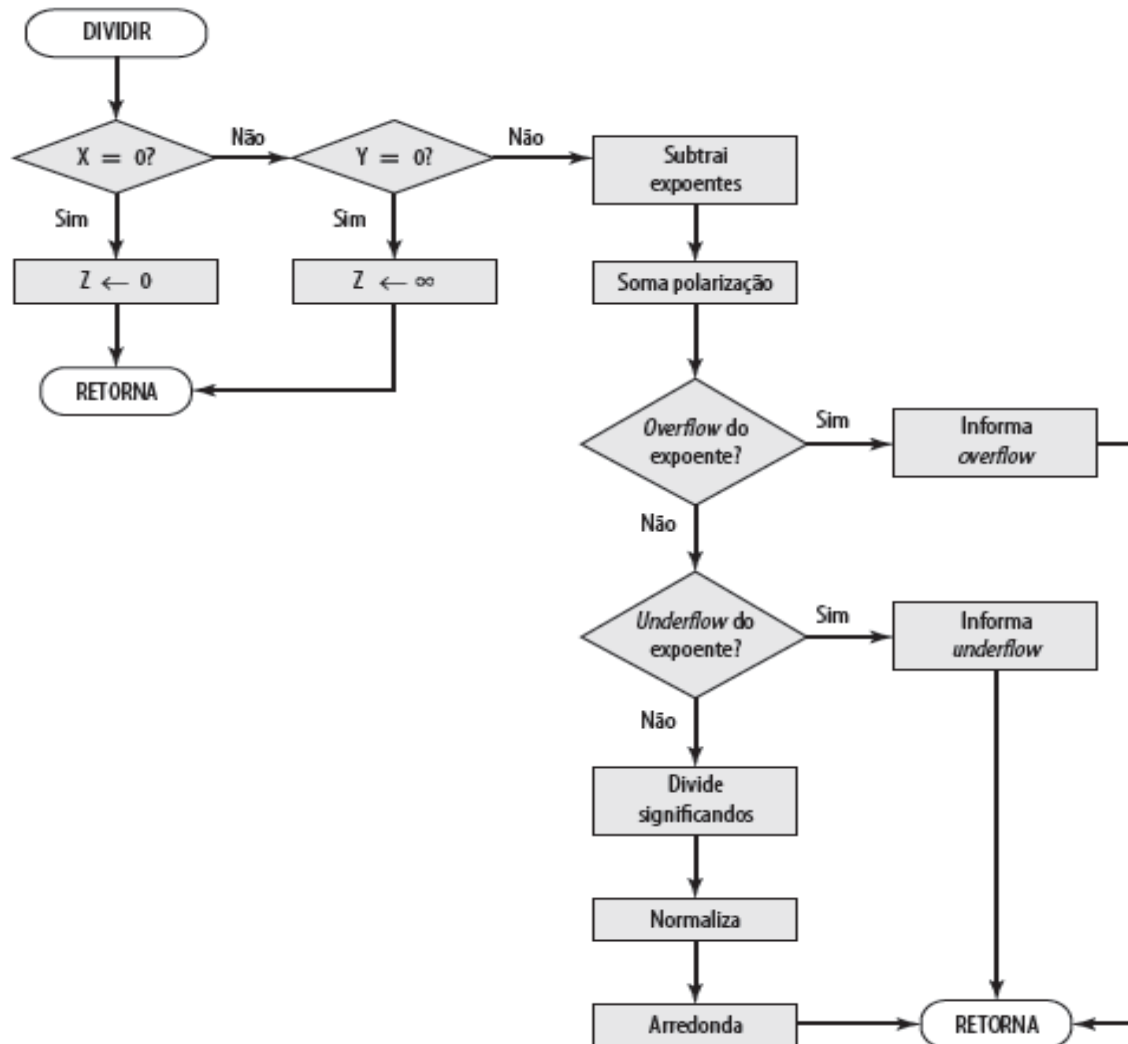
Aritmética de ponto flutuante (* e /)

- Verifique zero.
- Soma/subtraia expoentes .
- Multiplique/divida significandos (observe sinal).
- Normalize.
- Arredonde.
- Todos os resultados intermediários devem ser em armazenamento de tamanho duplo.

Multiplicação em Ponto Flutuante



Divisão em Ponto Flutuante



Resumo da Aula de Hoje

Tópicos mais importantes:

- **Aritmética Computacional**
- **Circuitos Aritméticos**
 - Circuito Multiplicador
 - Circuito Divisor
- **Entregar folha com:**
 - Nome
 - RA
 - Data de Hoje
 - Resumo

Referências

- **Notas de Aulas do Prof. João Angelo Martini do DIN-UEM.**