Arquitetura de Computadores I

Aritmética Computacional

- Inteiros -

Edson Moreno

edson.moreno@pucrs.br

http://www.inf.pucrs.br/~emoreno

Sumário

- **A unidade lógico-aritmética**
- **Representação de números inteiros**
- **Aritmética de números inteiros**

Unidade lógico-aritmética

- Executa os cálculos
- Os demais componentes de computação existem em função desta unidade
- Manipula números inteiros
- Pode manipular números de ponto flutuante
  - Integrado ao chip do processador
  - Separado em um co-processador (obsoleto)

Unidade lógico-aritmética

- Baseada em dispositivos digitais simples, capazes de
  - Armazenar valores binários
  - Efetuar operações simples de lógica booleana
- Dados fornecidos a partir de
  - Registradores de entrada
  - Armazenados em registradores na saída
- Pode sinalizar situações especiais
  - Associadas as operações realizadas (flags)

Unidade lógico-aritmética

- Mas como interpretar os valores armazenados?
  - Bits são apenas Bits!! Sem nenhum significado inerente
    - Ex.: o que é: 10100101 ????
- Podem representar
  - Número, instrução, pixel (luminância), amostra de voz, etc.
- O que podemos representar com N bits?
  - Apenas 2^N coisas!
  - Logo: Bom para coisas limitadas (contáveis)
  - Ex.:
    - 26 Letras: 5 bits é suficiente
    - Caracteres ASCII: 7 bits (de 8) (A,a,!)
    - Caracteres UNICODE: 16 bits (caracteres dos idiomas conhecidos)

Sumário

- **A unidade lógico-aritmética**
- ***Representação de números inteiros***
- **Aritmética de números inteiros**

Representação numérica

- **Base binária (base 2) :**
  - **Símbolos: 0,1**
    - Ex.: $124_{10} = 1111100_2$
- **Base octal (base 8):**
  - **Símbolos: 0,1,2,3,4,5,6,7**
    - Ex.: $124_{10}=1748_8$
- **Base decimal (base 10):**
  - **Símbolos: 0,1,2,3,4,5,6,7,8,9**
    - Ex.: $124_{10} = 124_{10}$
- **Base hexadecimal (base 16):**
  - **Símbolos: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**
    - Ex.: $124_{10} = 7C_{16}$

Representação numérica sem sinal

- Generalização de cálculo do valor
  - Dado um valor representado por n dígitos na base B

$$(d_{n-1}d_{n-2}d_{n-3}\ldots d_2d_1d_0)_B$$

- O valor de representação pode ser obtido pela fórmula

$$valor = \sum_{i=0}^{n-1} d_i \times B^i$$

  - n: número de posições
  - i: posição corrente
  - d_i: valor da posição corrente
  - B: base de representação

Representação numérica sem sinal

- Intervalo de representação de um valor binário sem sinal
  - Considerada apenas os valores positivos
  - Apoiado por algumas linguagens de programação
    - Ex. C/C++
      - int e unsigned int
  - Menor valor é sempre 0
  - Maior valor alcançado quando todos os bits estão em 1
    - Exemplo de 8 bits
      - $0000\ 0000_{(2)} = 0_{(10)}$
      - $0000\ 0001_{(2)} = 1_{(10)}$
      - $0000\ 0010_{(2)} = 2_{(10)}$
      - …
      - $1111\ 1100_{(2)} = 252_{(10)}$
      - $1111\ 1101_{(2)} = 253_{(10)}$
      - $1111\ 1110_{(2)} = 254_{(10)}$
      - $1111\ 1111_{(2)} = 255_{(10)}$

Cálculo genérico do intervalo

Valor mínimo = 0

Valor máximo = $B^n - 1$

Onde,

B: base de representação

n: número de dígitos da representação

Representação numérica com sinal

- Mas nem só de representações sem sinal vive a computação
  - Qual o mecanismo mais adequado para representar valores com sinal

Sinal e magnitude	**Complemento de um**	**Complemento de dois**
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = -0	100 = -3	100 = -4
101 = -1	101 = -2	101 = -3
110 = -2	110 = -1	110 = -2
111 = -3	111 = -0	111 = -1

Qual a melhor?

Complemento de 2

- A regra baseia-se no fato de que a soma de um número com sua representação invertida deve ser -1:

$$x + \bar{x} \equiv -1 \qquad x + \bar{x} + 1 = 0 \qquad \bar{x} + 1 = -x$$

- Intervalo de representação
  - O bit mais significativo representa o menor valor negativo alcançável
  - Menor valor:
    - MSb=1, demais em 0;
    - Fórmula geral: menor valor=$(-1)* 2^n$
  - Maior valor
    - MSb=0, demais em 1;
    - Fórmula geral: maior valor=2^n-1

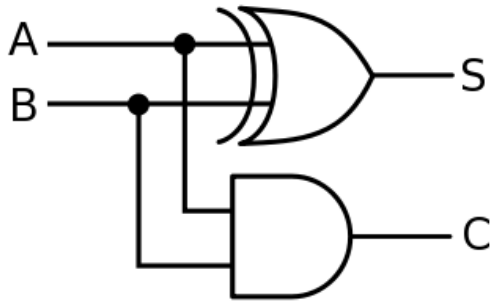
Sumário

- **A unidade lógico-aritmética**
- **Representação de números inteiros**
- ***Aritmética de números inteiros***

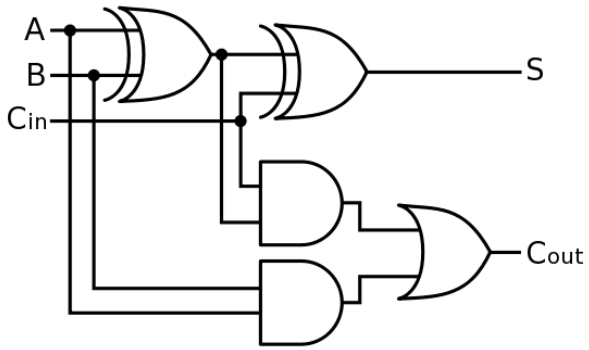
Somador de 1 bit

- Tipos básicos de construção

- Half adder
  - Circuito lógico que executa uma adição de dois operandos binários de 1 bit cada, produzindo o resultado da soma e um carry



- Full adder
  - Circuito lógico que executa uma adição de dois operandos binários de 1 bit cada, produzindo o resultado da soma e um carry



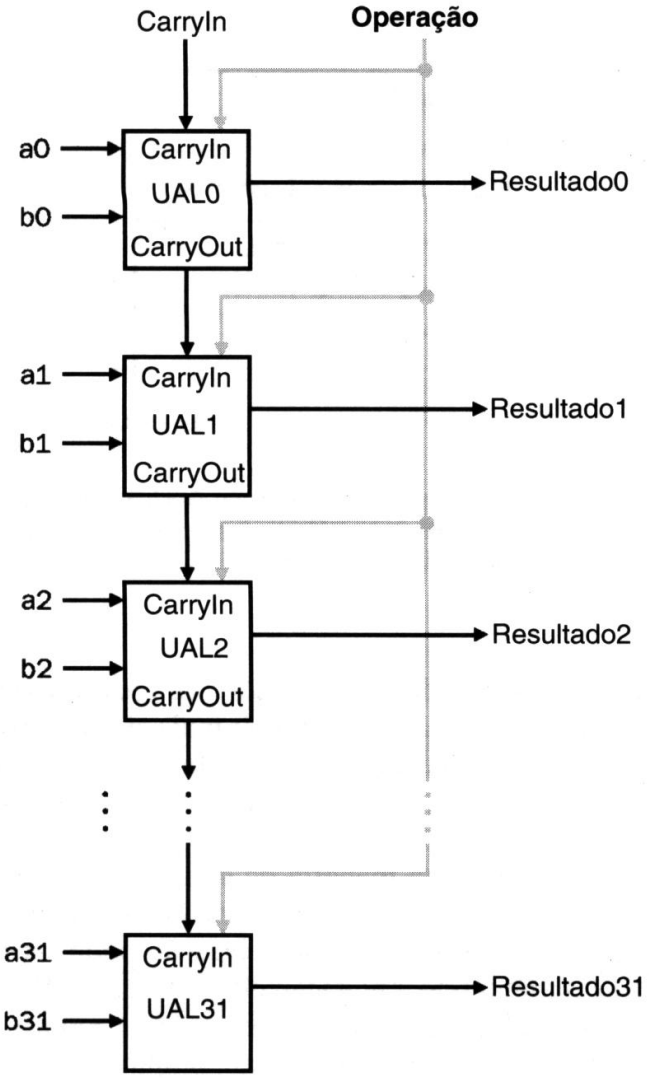
Somador de 1 bit

- Soma A + B + "vem 1"
- Gera Resultado e "vai um"
- Tabela Verdade:

Entradas			Saídas		Comentários
A	B	Vem 1	Soma	Vai 1	
0	0	0	0	0	0+0+0 = 00
0	0	1	1	0	0+0+1 = 01
0	1	0	1	0	0+1+0 = 01
0	1	1	0	1	0+1+1 = 10
1	0	0	1	0	1+0+0 = 01
1	0	1	0	1	1+0+1 = 10
1	1	0	0	1	1+1+0 = 10
1	1	1	1	1	1+1+1 = 11

Somador de múltiplos bits

- Somador em cascata (ripple carry)
  - Características
    - Múltiplos *full adder* em cadeia para a produção da soma de números de n bits
    - O primeiro somador pode ser *half adder*
    - O carry é propagado entre somadores
    - Método similar ao lápis e papel!!!

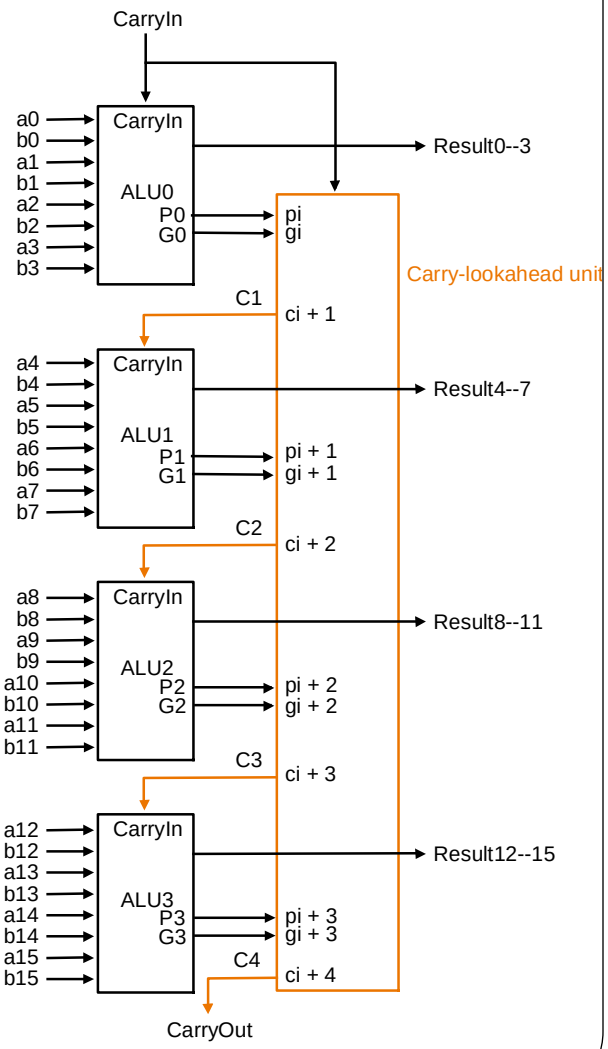


Somador de múltiplos bits

- Dois extremos de desempenho
  - Ripple carry
    - Vantagem
      - Facilita o projeto do somador (reuso de módulos)
    - Desvantagem
      - O somador é lento (caminho crítico)
      - Pois:
  - Soma de produtos / produto de somas
    - Vantagem
      - Lógica extremamente rápida (circuito específico)
    - Desvantagem
      - Circuito complexo para cálculos de muitos bits (custo alto)

Somador de múltiplos bits

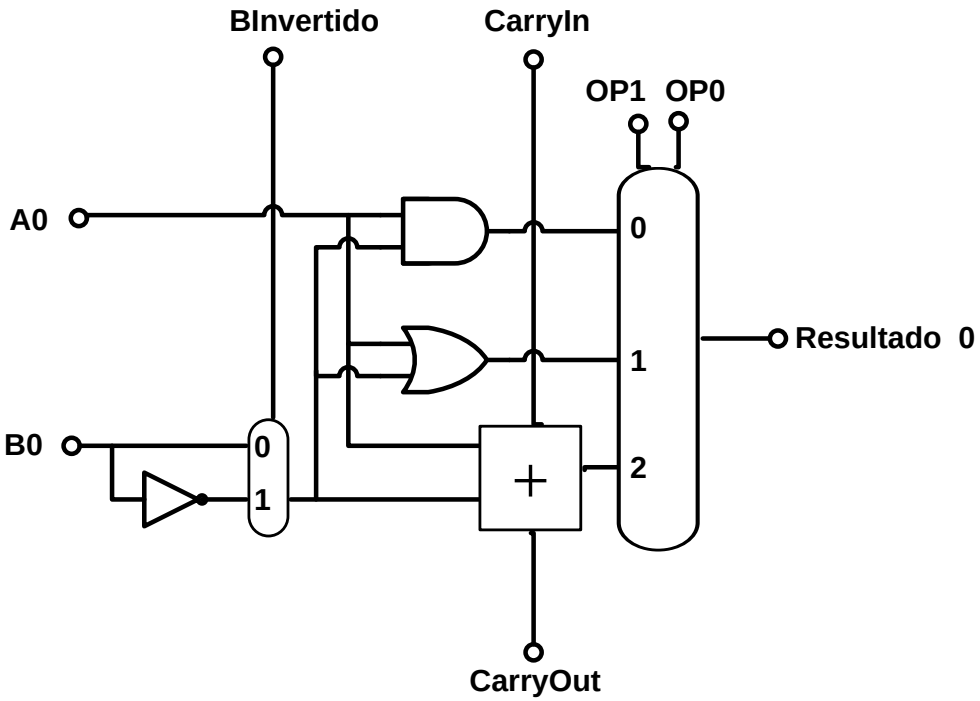
- Carry look-ahead (meio termo)
  - Característica
    - Mescla *ripple carries* (4bits)
    - Antecipa se gerará / propagará carry
      - Geração: $g_i = a_i$ and b_i
      - Propagação: $p_i = a_i$ or b_i
    - Assim
      - $C_1 = g_0$ or (p_0 and c_0)
      - $C_2 = g_1$ or (p_1 and c_1)
      - …
  - Vantagem
    - Acelera o processo de computação de soma dos operandos
    - Propagação do carry é bem mais rápido



Subtração

- A subtração é obtida somando-se o minuendo ao complemento a 2 do subtraendo, ou seja,

$$a - b = a + (\overline{b} + 1)$$



Multiplicação

- Mais complicado que a soma
  - Construídos a partir de somas e deslocamentos
- Reflete em mais
  - Tempo de execução ou
  - Área
- Base de operação
  - Opera-se o produto parcial de cada dígito
  - Desloca-se o produto parcial uma casa à esquerda
  - Adiciona-se os produtos parciais

Multiplicação

- Exemplo:

multiplicando 1000

multiplicador x 1001

1000

0000

0000

1000

produto 1001000

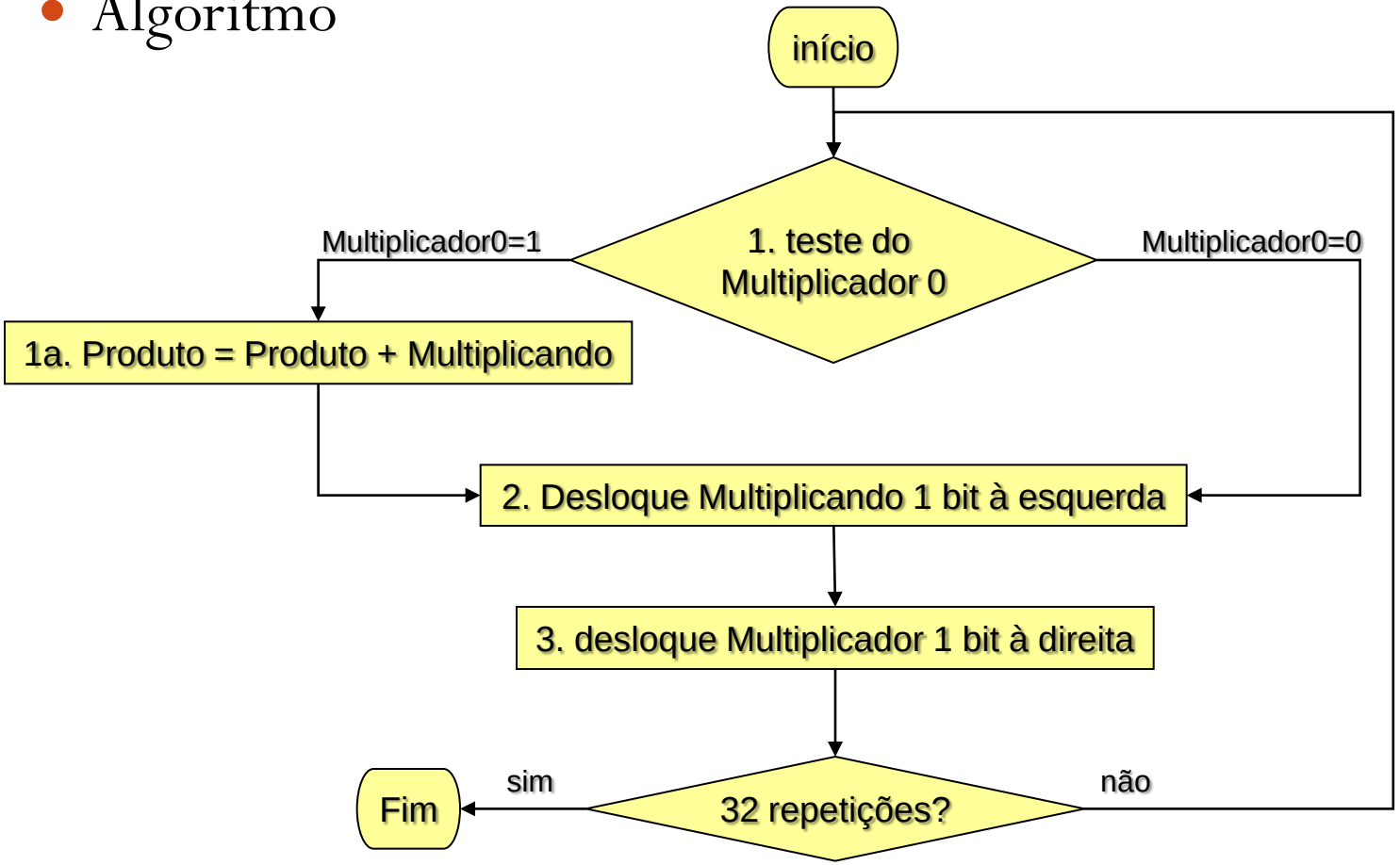
- Número de dígitos: multiplicando + multiplicador.
- 32 bits x 32 bits = 64 bits.

Multiplicação

- Cálculo feito como no coleginho
  - Simplesmente coloque um cópia do multiplicando (1 x multiplicando) no lugar apropriado, se o digito do multiplicando for igual a 1, ou
  - Coloque 0 (0 x multiplicando) no lugar apropriado, se o dígito do multiplicando for igual a 0;
  - Veremos a seguir 3 versões do algoritmo de multiplicação para 32 bits (32 x 32 bits)

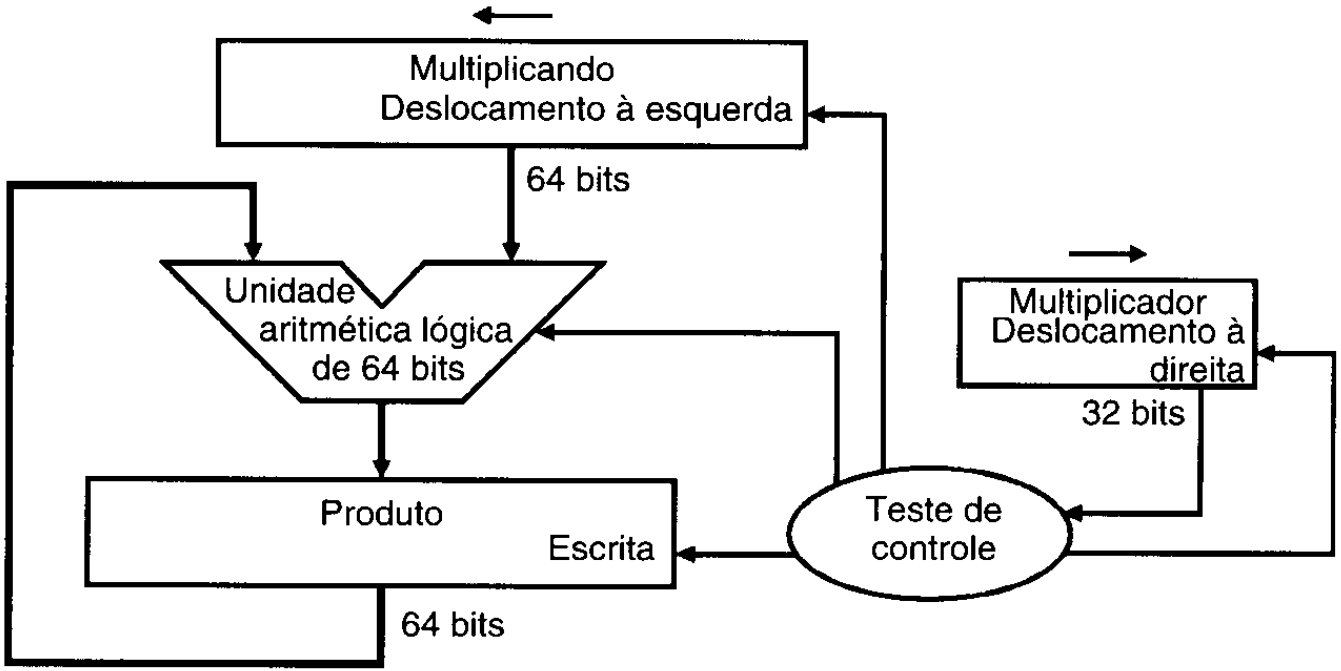
Multiplicação – Versão 1

- Algoritmo



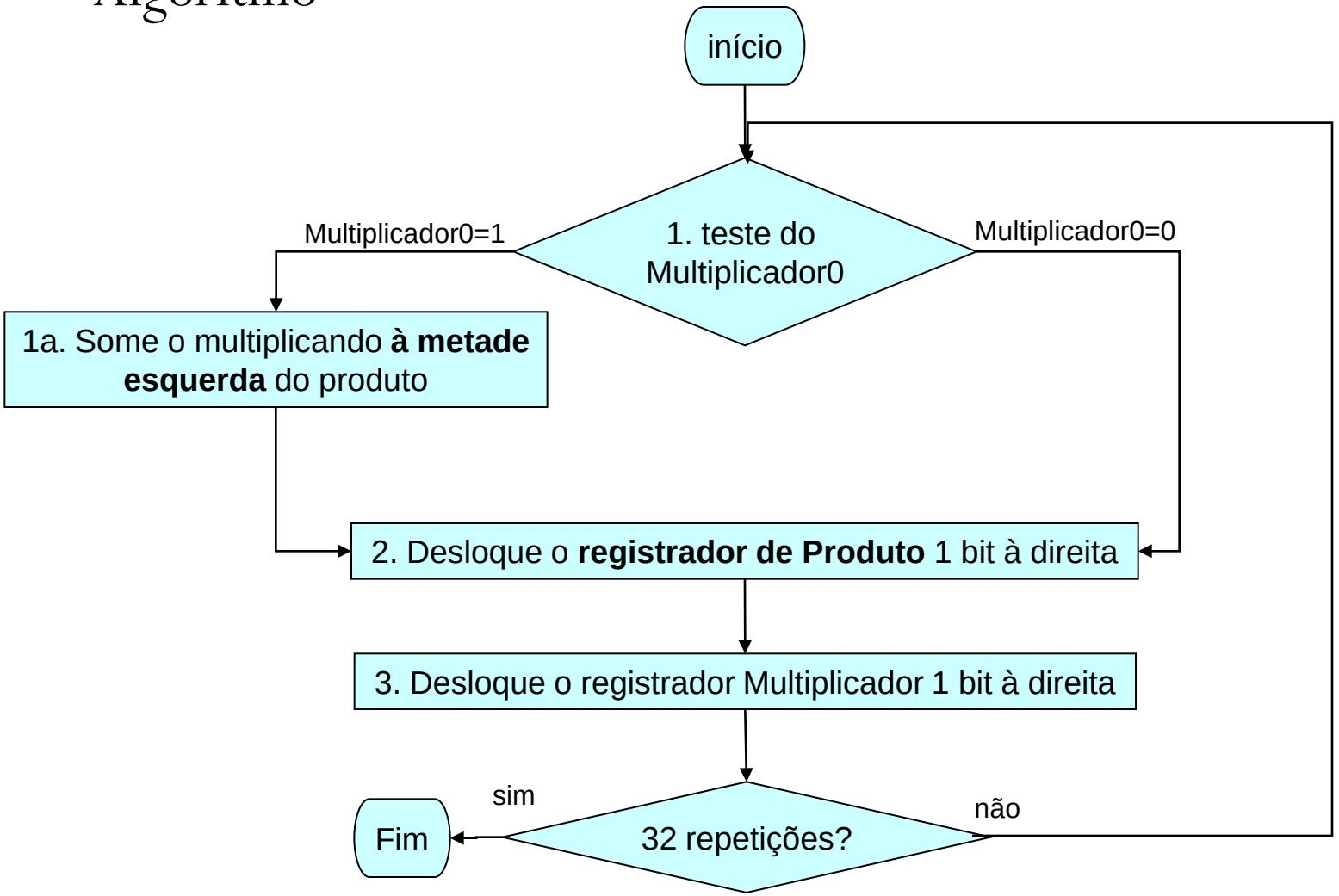
Multiplicação – Versão 1

- Implementação em hardware



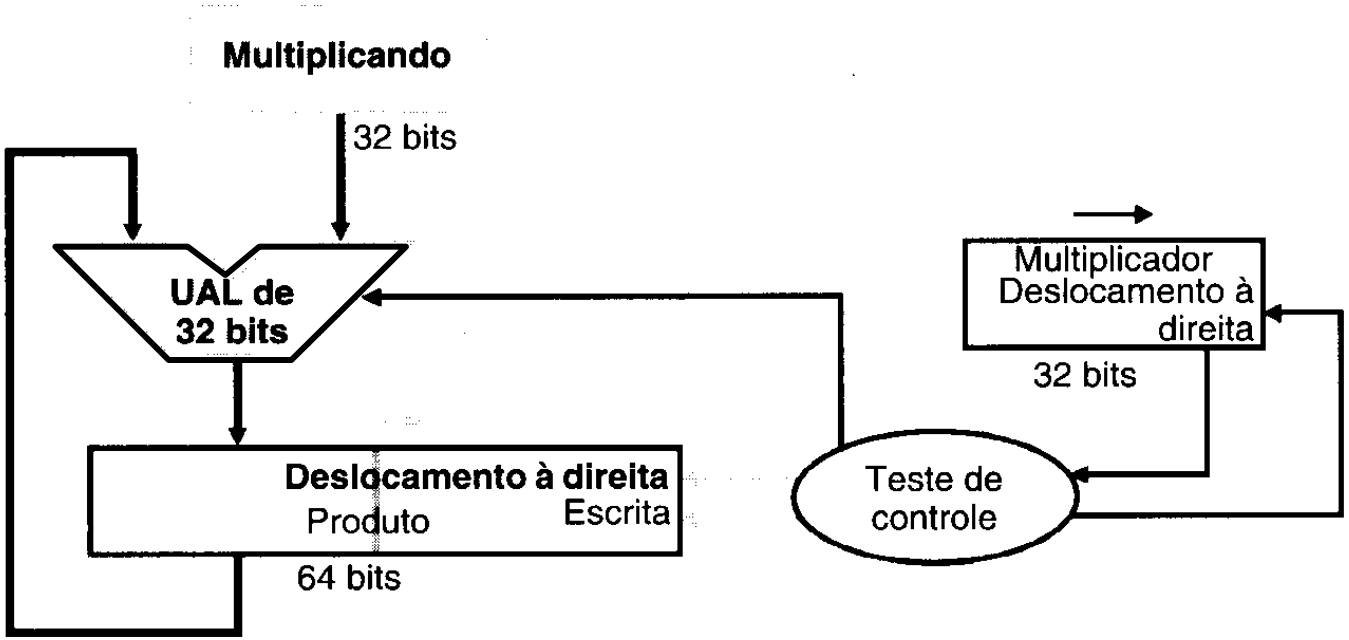
Multiplicação – Versão 2

- Algoritmo



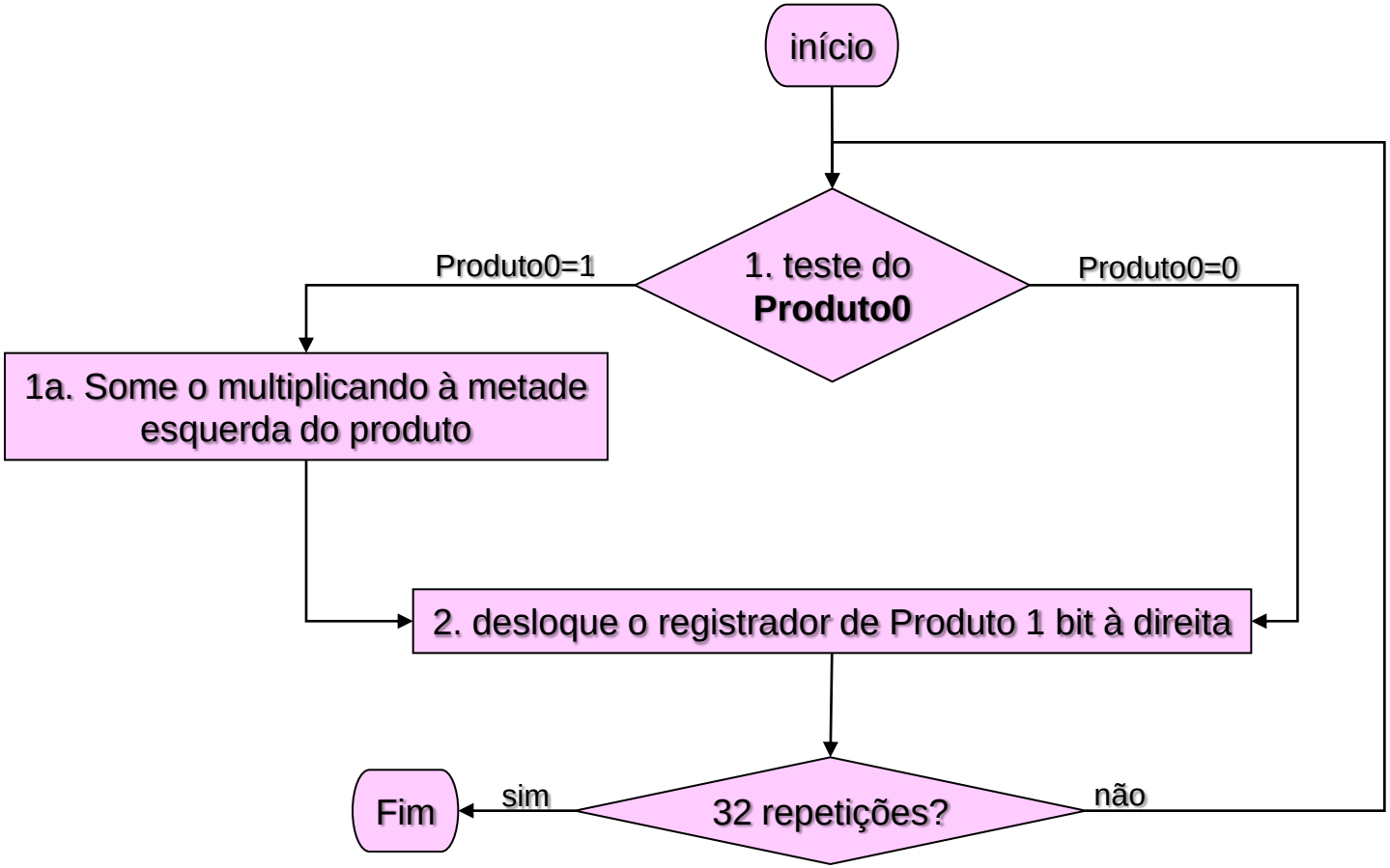
Multiplicação – Versão 2

- Implementação em hardware



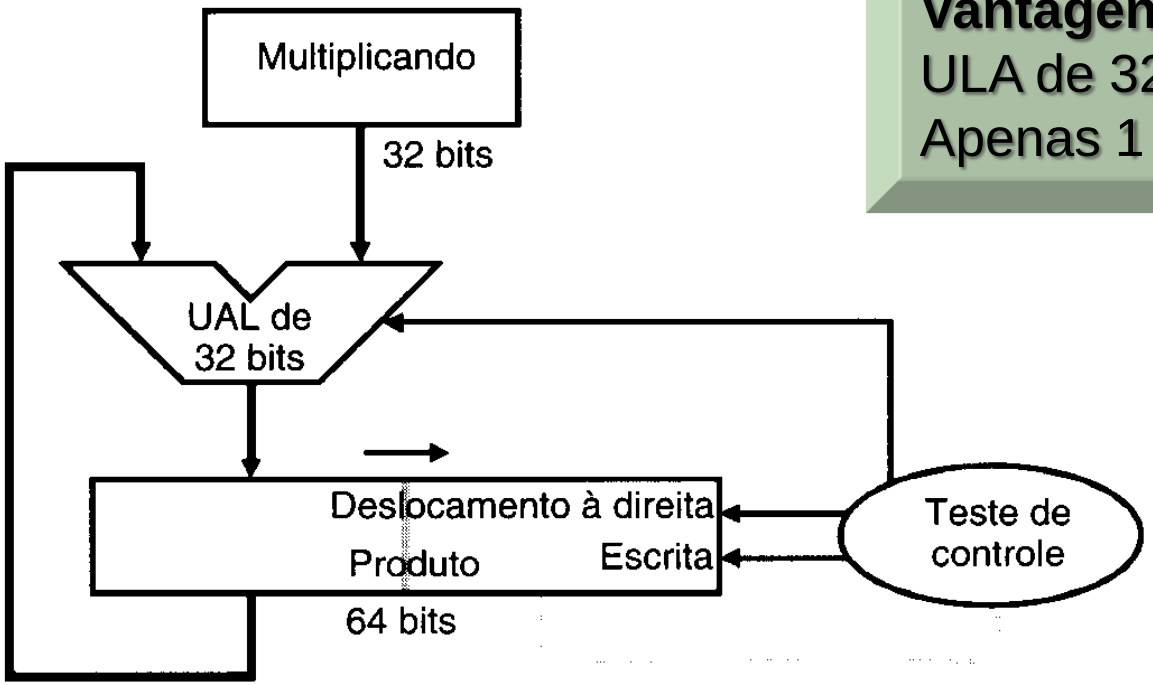
Multiplicação – Versão 3

- Algoritmo



Multiplicação – Versão 3

- Implementação em hardware



Vantagens:
ULA de 32 bits.
Apenas 1 registrador de 64 bits.

Divisão

- Característica
  - Operação menos freqüente que a multiplicação e mais complexa embora baseada nos mesmos princípios
  - Oferece oportunidade de efetuar operação matemática inválida
    - Divisão por zero.
  - Representação
    - dividendo = quociente x divisor + resto

Divisão

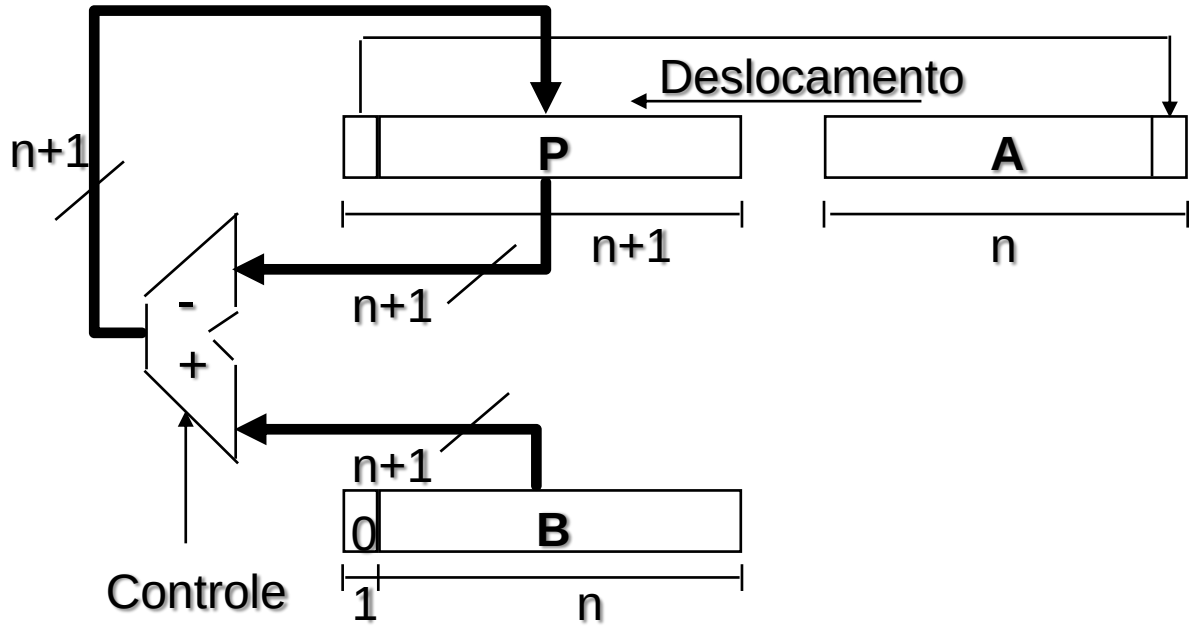
- Comportamento implementado
  - Operação envolve repetidas execuções de soma, subtração e deslocamento
  - Os bits do dividendo são analisados da esquerda para a direita, até que um número de representação maior ou igual ao divisor seja obtido
  - Enquanto não ocorrer são inseridos é '0's no quociente da esquerda para a direita

Divisão

- Comportamento implementado
  - Quando encontrado um número maior ou igual (diz-se que o divisor divide o número) é colocado um bit em 1 no quociente e o divisor é subtraído do dividendo parcial
  - O resto é chamado de resto parcial. A partir deste ponto, a divisão segue o mesmo padrão cíclico. A cada ciclo, bits adicionais do dividendo são anexados ao resto até que o resultado seja maior ou igual ao divisor.
  - O processo segue até que todos bits do dividendo sejam examinados.

Divisão

- Solução para a/b: subtrações sucessivas, n passos;



- Quatro passos de operação:
  - (1) desloca P&A p/ esq 1 bit;
  - (2) P<- P-B;
  - (3) se (P<0), A0=0 else A0 =1;
  - (4) If (P<0), restaura P fazendo P<-P+B.

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1											
2											
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

	P (conterá o resto)						A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
2											
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

	P (conterá o resto)						A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	1	1	1	1	0	0	1	0	1	1	0
2											
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	1	1	1	1	0	0	1	0	1	1	0
	1	1	1	1	0	0	1	0	1	1	0
2											
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	1	1	1	1	0	0	1	0	1	1	0
	1	1	1	1	0	0	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2											
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)						A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2											
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3											
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
4											
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4											
5											

00110 - 00101 = 001

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4	0	0	0	0	1	1	1	0	0	1	0
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4	0	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	1	1	0	0	1	0
5											

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4	0	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	1	1	0	0	1	0
5	0	0	0	1	1	1	0	0	1	0	0

Divisão A/B - Exemplo

A = 11011 (27)
B = 00101 (5)

A cada volta, mostra-se P e A após executar o passo:
1) – na primeira linha
4) – na segunda linha

Passos:
1) desloca P&A p/ esq 1 bit;
2) P ⇐ P-B;
3) If (P<0), A_0=0 else A_0 =1;
4) If (P<0), restaura P fazendo P ⇐ P+B

		P (conterá o resto)					A (conterá a divisão)				
volta	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4	0	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	1	1	0	0	1	0
5	0	0	0	1	1	1	0	0	1	0	0
	0	0	0	0	1	0	0	0	1	0	1

Resto = 2 **Quociente = 5**