



Aritmética 2/3

Multiplicação e Divisão Inteira

Edson S. Gomi

Revisão: Marco Túlio Andrade

PCS - Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Setembro, 2020

Agenda

Multiplicação

Divisão

1 Multiplicação

2 Divisão

Ao final da aula você saberá:

Multiplicação

Divisão

- Os conceitos e algorítmos principais de operações de multiplicação e divisão;
- Qual é o (*hardware*) necessário, em termos de uma Unidade de Controle e um Fluxo de Dados para se desenvolver blocos multiplicadores e blocos divisores;

Somador Completo de 1 bit

Multiplicação

Divisão

Antes de abordar os circuitos de multiplicação e divisão vamos dar uma olhada no conceito das operações de multiplicação e divisão, segundo Richard Feynman, no capítulo 22, volume 1 de seus Lições de Física:

"Uma vez que definimos a adição podemos considerar isto: Se começamos com nada e somamos um certo número a inteiro, sucessivamente por b vezes, chamamos ao resultado, que é o número no qual chegamos, de multiplicação $c = a * b$. Isto define a multiplicação de inteiros"

Raciocínio semelhante permite conceituar a operação reversa, a divisão $b = c / a$.

Unsigned Multiplication

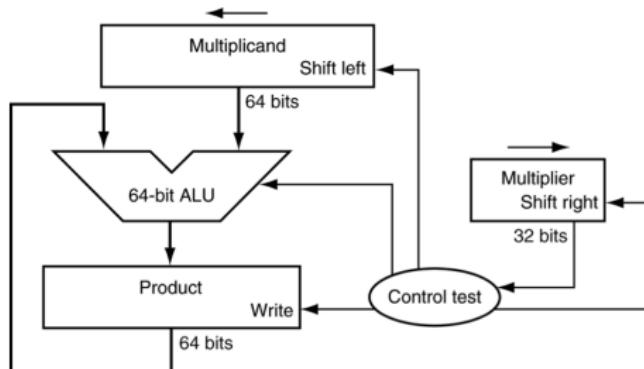
Multiplicação

Divisão

A figura ilustra o cálculo da multiplicação binária (sem sinal):

$$\begin{array}{r} \text{multiplicand} \\ \text{multiplier} \\ \times \quad \quad \quad 1001 \\ \hline \quad \quad \quad 1000 \\ \quad \quad \quad 0000 \\ \quad \quad \quad 0000 \\ \quad \quad \quad 1000 \\ \hline \quad \quad \quad 1001000 \end{array}$$

Length of product is
the sum of operand
lengths

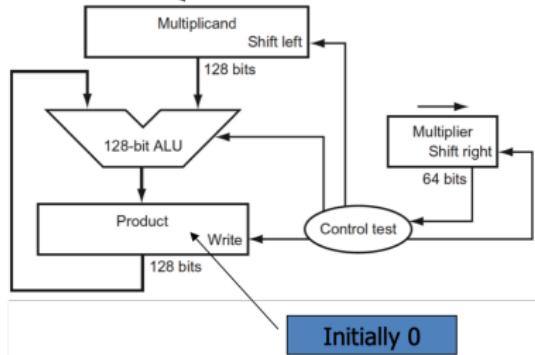
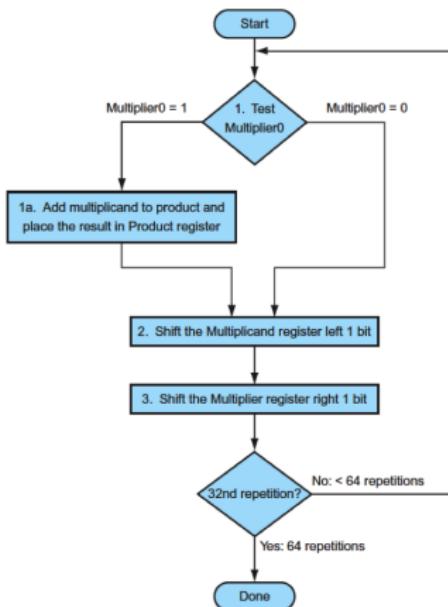


Hardware da Multiplicação

Multiplicação

Divisão

Primeira versão do hardware da multiplicação:

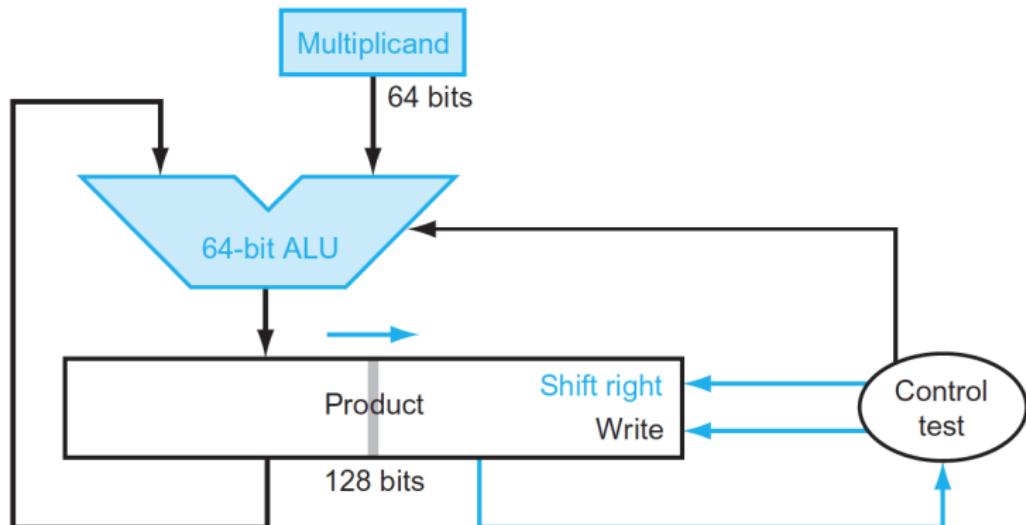


Hardware da Multiplicação

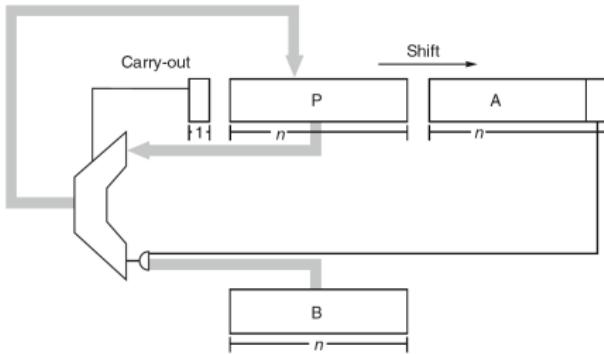
Multiplicação

Divisão

Versão refinada do hardware da multiplicação:



Algoritmo da Multiplicação



- O multiplicando $a_{n-1}a_{n-2}\dots a_0$ e o multiplicador $b_{n-1}b_{n-2}\dots b_0$ são colocados nos registradores A e B.
- O registrador P é inicializado com 0.
- Cada passo da multiplicação tem 2 partes:
 - (i) Se o bit menos significativo de A é 1, então faça $P = P + B$, senão $P = P + 0$;
 - (ii) Deslocar os registradores P e A 1 bit para a direita. O bit menos significativo de A é descartado.

Algoritmo da Multiplicação

Multiplicação

Divisão

Exemplo do cálculo de $2_{10} \times 3_{10} = 6_{10}$
($0010_2 \times 0011_2 = 0110_2$):

P	A	
0000	0010	
0000	0010	$P = P + 0 \text{ (0000)}$
0000	0001	Shift P Right (Step 0)
0011	0001	$P = P + B \text{ (0011)}$
0001	1000	Shift P Right (Step 1)
0001	1000	$P = P + 0 \text{ (0000)}$
0000	1100	Shift P Right (Step 2)
0000	1100	$P = P + 0 \text{ (0000)}$
0000	0110	Shift P Right (Step 3)

Signed Multiplication

Multiplicação

Divisão

Um método simples para multiplicar números com representação em Complemento de 2 é converter o multiplicando e o multiplicador para números não negativos. Depois fazer a multiplicação sem sinal (*unsigned*) e, caso os operandos originais tenham sinais opostos, tornar o resultado um número negativo.

Signed Multiplication: Algoritmo de Booth

Multiplicação

Divisão

Um outro método mais eficiente para multiplicar 2 números em Complemento de 2 é o Algoritmo de Booth.

Apresentaremos em sequencia o Algoritmo de Booth em sua versão mais conceitual e genérica.

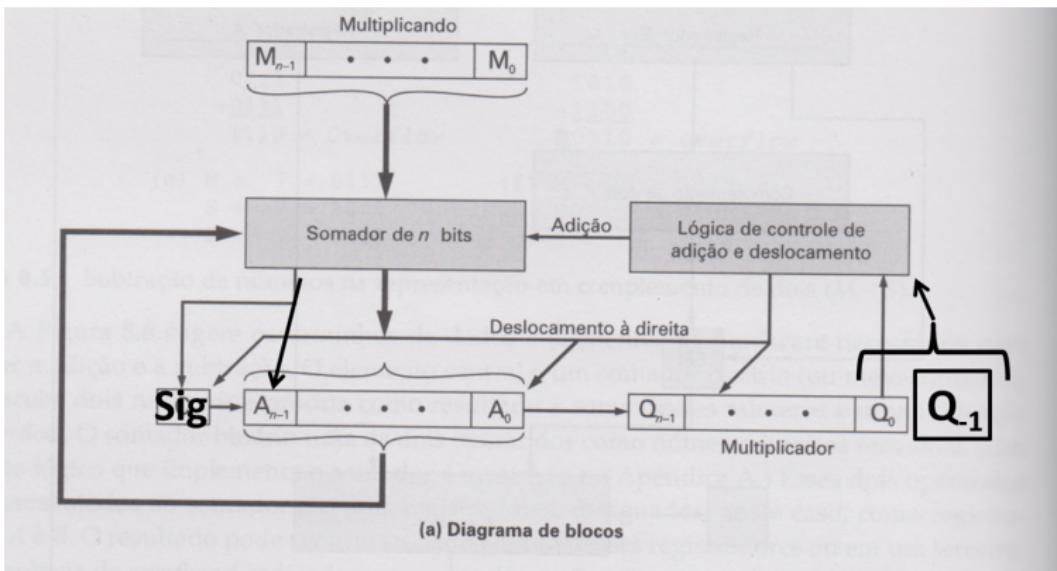
Em seguida uma outra versão do Algoritmo de Booth, onde se muda a representação de um dos operandos por meio de uma operação de recodificação dos dados e por isto é conhecida por recodificação de Booth (*Booth Recoding*).

Signed Multiplication: Algoritmo de Booth

Multiplicação

Divisão

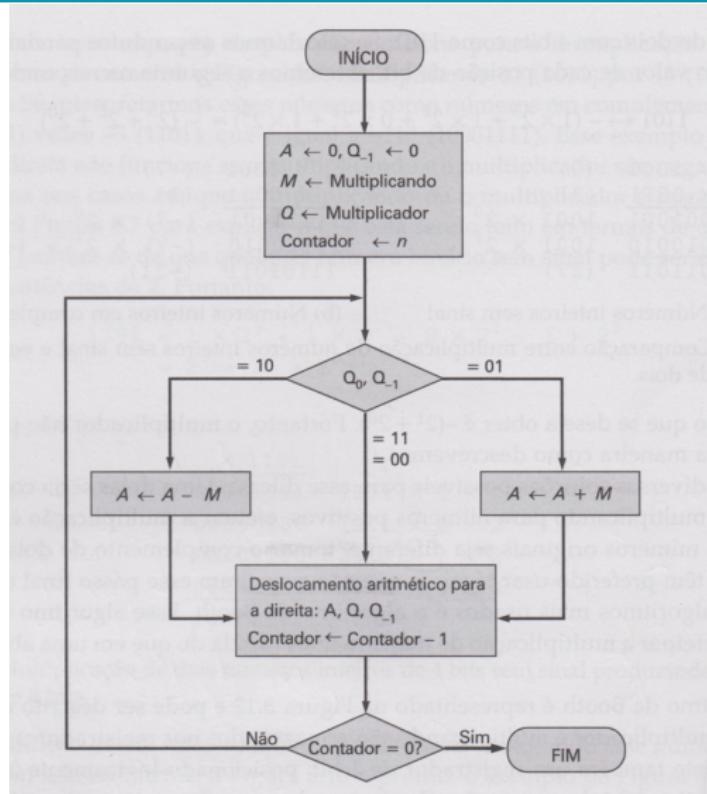
Diagrama de Blocos.



Signed Multiplication: Algoritmo de Booth

Multiplicação

Divisão



Signed Multiplication: Algoritmo de Booth

Multiplicação

Divisão

Por que o algoritmo de Booth funciona?

$$\begin{aligned} M \times (00011110) &= M \times (2^4 + 2^3 + 2^2 + 2^1) \\ &= M \times (16 + 8 + 4 + 2) \\ &= M \times 30 \end{aligned}$$

$$2^n + 2^{n-1} + \dots + 2^{n-K} = 2^{n+1} - 2^{n-K}$$

$$\begin{aligned} M \times (00011110) &= M \times (2^5 - 2^1) \\ &= M \times (32 - 2) \\ &= M \times 30 \end{aligned}$$

$$\begin{aligned} M \times (01111010) &= M \times (2^6 + 2^5 + 2^4 + 2^3 + 2^1) \\ &= M \times (2^7 - 2^3 + 2^2 - 2^1) \end{aligned}$$

Signed Multiplication: Algoritmo de Booth

Multiplicação

Divisão

Exemplos de operações com sinal [1/2]

$$\begin{array}{r} 0111 \\ \times 0011 \\ \hline \end{array} \quad (0)$$

$$11111001 \quad 1-0$$

$$0000000 \quad 1-1$$

$$\underline{000111} \quad 0-1$$

$$00010101 \quad (21)$$

$$\begin{array}{r} 0111 \\ \times 1101 \\ \hline \end{array} \quad (0)$$

$$11111001 \quad 1-0$$

$$0000111 \quad 0-1$$

$$\underline{111001} \quad 1-0$$

$$11101011 \quad (-21)$$

$$(a) \ (7) \times (3) = (21) \quad (b) \ (7) \times (-3) = (-21)$$

Signed Multiplication: Algoritmo de Booth

Multiplicação

Divisão

Exemplos de operações com sinal [2/2]

$$\begin{array}{r} 1001 \\ \times 0011 \\ \hline 00000111 & (0) \\ 00000000 & 1-0 \\ \underline{111001} & 1-1 \\ 11101011 & 0-1 \\ \hline \end{array}$$

$$\begin{array}{r} 1001 \\ \times 1101 \\ \hline 00000111 & (0) \\ 1111001 & 1-0 \\ \underline{000111} & 0-1 \\ 00010101 & 1-0 \\ \hline \end{array}$$

$$(c) \ (-7) \times (3) = (-21) \quad (d) \ (-7) \times (-3) = (21)$$

Booth Recoding

Multiplicação

Divisão

Considere o multiplicando

$62_{10} = 00111110_2 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1$. Podemos reescrever como

$$62_{10} = 64_{10} - 2_{10} = 2^6 - 2^1 = 0100000_2 - 10_2 = 010000\bar{1}0_2.$$

O símbolo $\bar{1}$ representa -1 . No algoritmo da multiplicação, $\bar{1}$ faz com que o multiplicador seja subtraído do produto parcial. A recodificação de Booth é feita facilmente com o uso desta tabela:

x_i	x_{i-1}	y_i
0	0	0
0	1	1
1	0	$\bar{1}$
1	1	0

Radix-2 Booth's Multiplication

Multiplicação

Divisão

No algoritmo da multiplicação não é preciso representar explicitamente o multiplicando recodificado. Verifique, a cada iteração i do Algoritmo da Multiplicação, o valor do bit menos significativo a_i do registrador A e execute o passo (i) do algoritmo de acordo com as seguintes regras:

- (I) Se $a_i = 0$ e $a_{i-1} = 0$ então $P = P + 0$
- (II) Se $a_i = 0$ e $a_{i-1} = 1$ então $P = P + B$
- (III) Se $a_i = 1$ e $a_{i-1} = 0$ então $P = P - B$
- (IV) Se $a_i = 1$ e $a_{i-1} = 1$ então $P = P + 0$

Para a 1a. iteração ($i = 0$) assuma que $a_{-1} = 0$.

Algoritmo de Booth : exemplo

Multiplicação

Divisão

Cálculo de $-6_{10} \times -5_{10} = 30_{10}$
 $(1010_2 \times 1011_2 = 00011110_2)$:

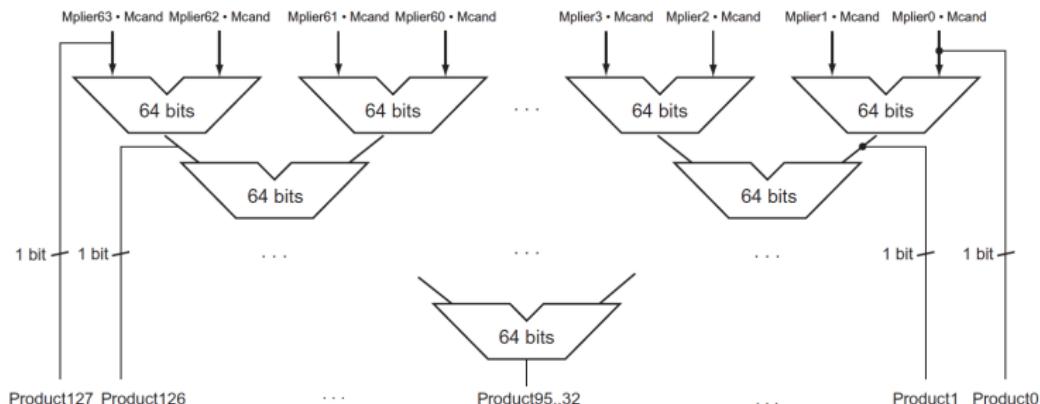
P	A	
0000	1010	Put $-6 = 1010_2$ into A, $-5 = 1011_2$ into B.
0000	1010	step 1(i): $a_0 = a_{-1} = 0$, so from rule I add 0.
0000	0101	step 1(ii): shift.
+0101		step 2(i): $a_1 = 1$, $a_0 = 0$. Rule III says subtract b (or add $-b = -1011_2 = 0101_2$).
0101	0101	
0010	1010	step 2(ii): shift.
+ 1011		step 3(i): $a_2 = 0$, $a_1 = 1$. Rule II says add b (1011).
1101	1010	
1110	1101	step 3(ii): shift. (Arithmetic shift—load 1 into leftmost bit.)
+ 0101		step 4(i): $a_3 = 1$, $a_2 = 0$. Rule III says subtract b.
0011	1101	
0001	1110	step 4(ii): shift. Final result is $00011110_2 = 30$.

Multiplicação Rápida

Multiplicação

Divisão

Como obter multiplicações mais rápidas?



Multiplicação Rápida: Circuitos Combinatórios

Multiplicação

Divisão

Multiplicador 8x8

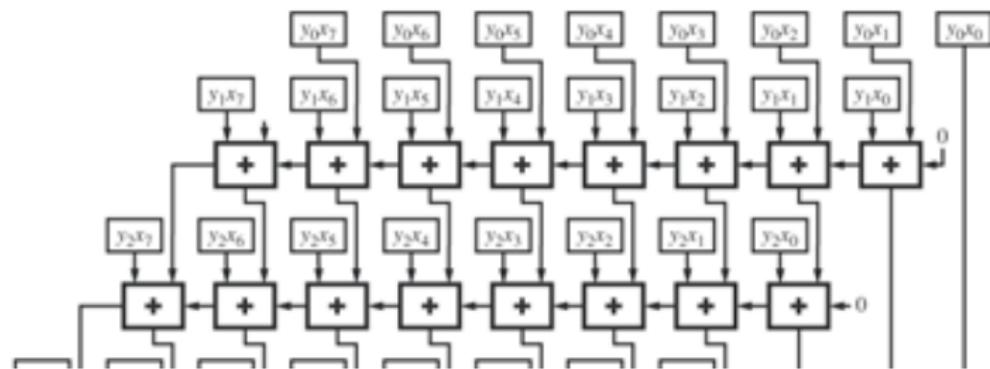
Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

y_0x_7	y_0x_6	y_0x_5	y_0x_4	y_0x_3	y_0x_2	y_0x_1	y_0x_0
y_1x_7	y_1x_6	y_1x_5	y_1x_4	y_1x_3	y_1x_2	y_1x_1	y_1x_0
y_2x_7	y_2x_6	y_2x_5	y_2x_4	y_2x_3	y_2x_2	y_2x_1	y_2x_0
y_3x_7	y_3x_6	y_3x_5	y_3x_4	y_3x_3	y_3x_2	y_3x_1	y_3x_0
y_4x_7	y_4x_6	y_4x_5	y_4x_4	y_4x_3	y_4x_2	y_4x_1	y_4x_0
y_5x_7	y_5x_6	y_5x_5	y_5x_4	y_5x_3	y_5x_2	y_5x_1	y_5x_0
y_6x_7	y_6x_6	y_6x_5	y_6x_4	y_6x_3	y_6x_2	y_6x_1	y_6x_0
y_7x_7	y_7x_6	y_7x_5	y_7x_4	y_7x_3	y_7x_2	y_7x_1	y_7x_0
+							
p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	p_{10}	p_9	p_8
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Multiplicação Rápida: Circuitos Combinatórios

Multiplicação

Divisão

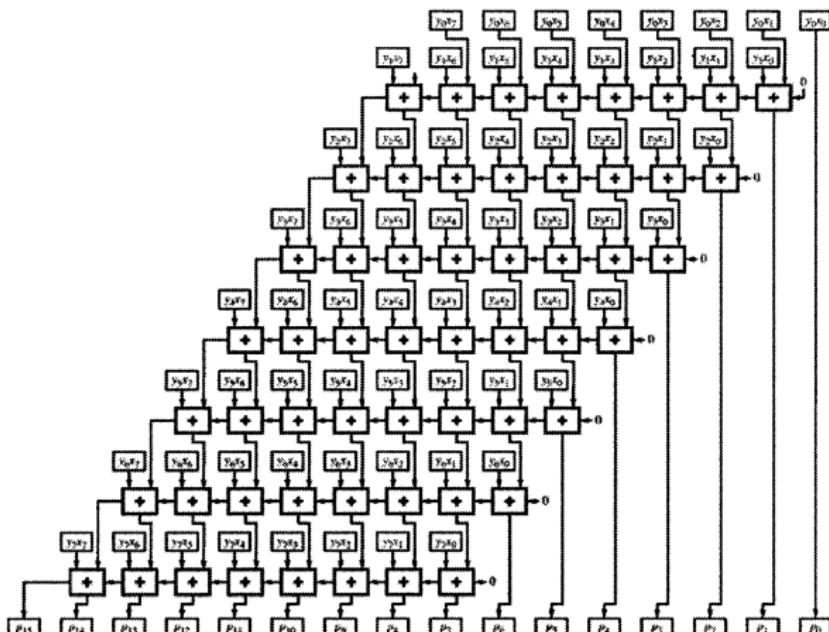


Multiplicação Rápida: Circuitos Combinatórios

Multiplicação

Divisão

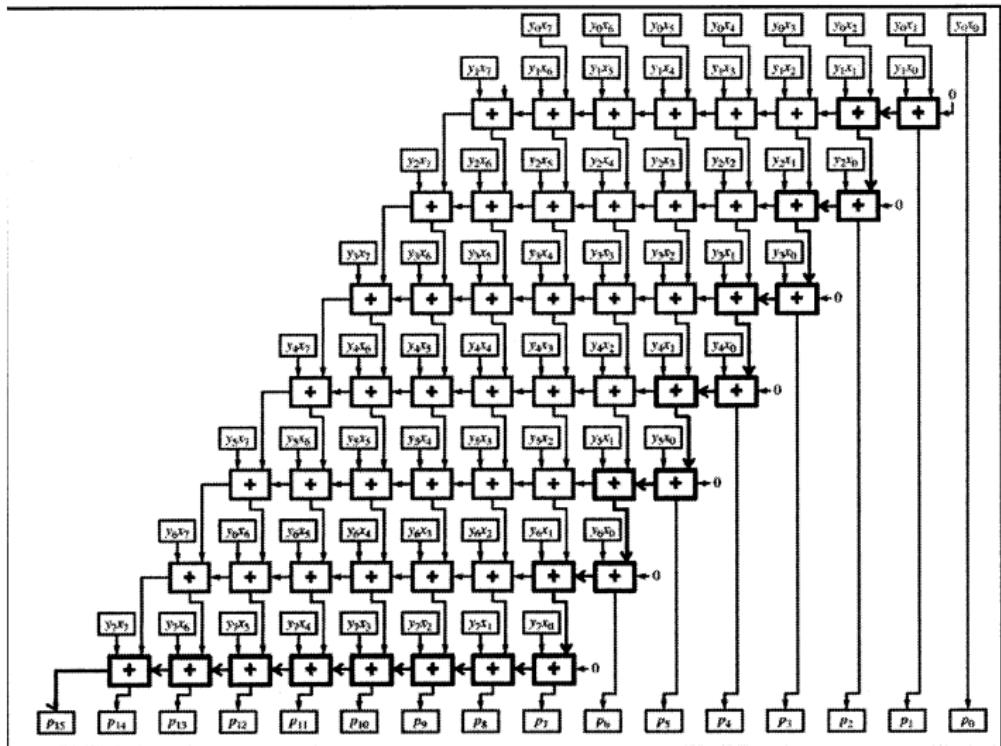
Multiplicador Combinatório (fonte: Wakerly)



Multiplicação Rápida: Circuitos Combinatórios

Multiplicação

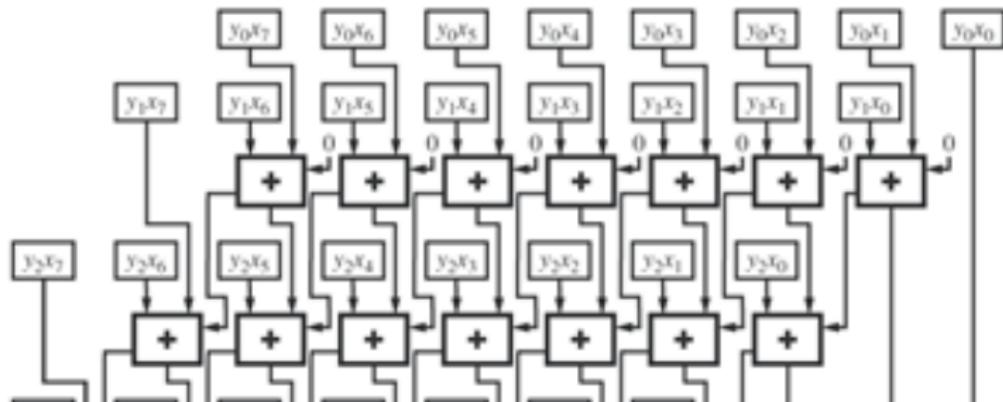
Divisão



Multiplicação Rápida: Circuitos Combinatórios

Multiplicação

Divisão

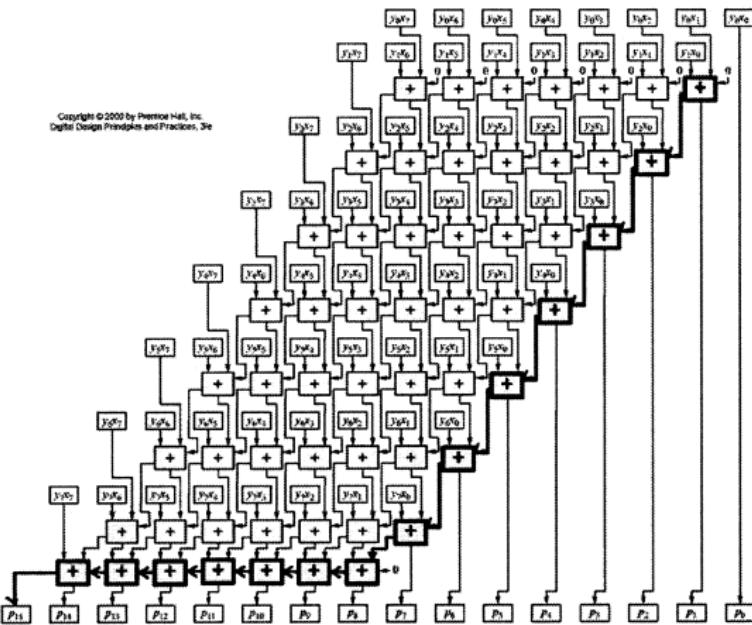


Multiplicação Rápida: Circuitos Combinatórios

Multiplicação

Divisão

Faster 8x8 Multiplier

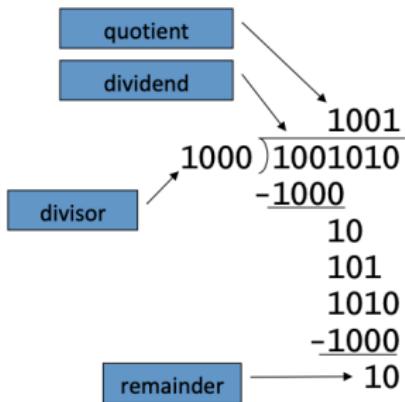


Unsigned Division

Multiplicação

Divisão

A figura ilustra o cálculo da divisão binária (sem sinal):



n -bit operands yield n -bit quotient and remainder

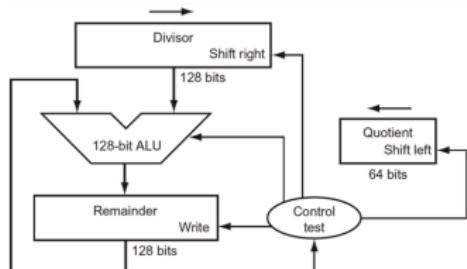
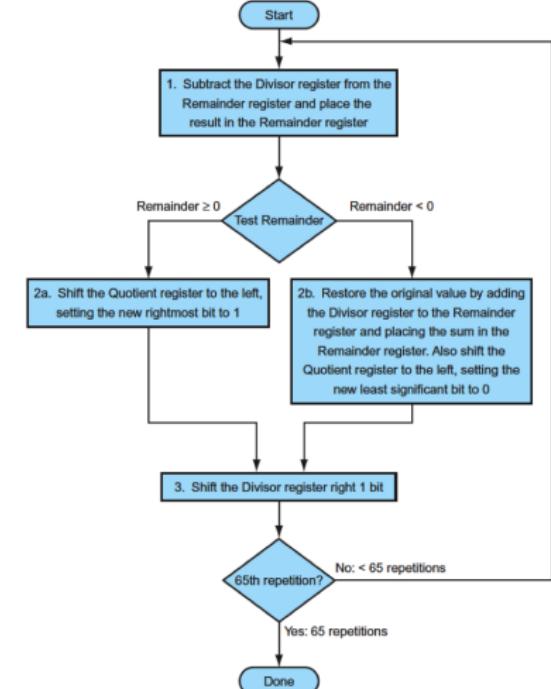
- Check for 0 divisor
- Long division approach
 - If divisor \leq dividend bits
 - 1 bit in quotient, subtract
 - Otherwise
 - 0 bit in quotient, bring down next dividend bit
- Restoring division
 - Do the subtract, and if remainder goes < 0 , add divisor back
- Signed division
 - Divide using absolute values
 - Adjust sign of quotient and remainder as required

Hardware da Divisão

Multiplicação

Divisão

Primeira versão do hardware da divisão:



Exemplo de Divisão

Multiplicação

Divisão

Execução de $7_{10} \div 2_{10} = 0000111_2 \div 0010_2 = 3_{10} = 0011_2$
com resto $1_{10} = 0001_2$

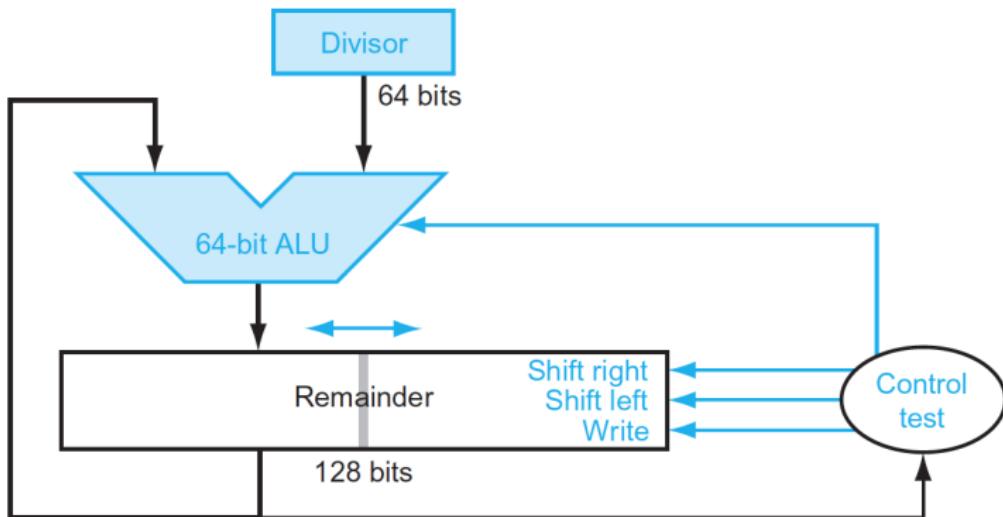
Iteration	Step	Quotient	Divisor	Reminder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem - Div	0000	0010 0000	0110 0111
	2b: Rem < 0 \Rightarrow +Div, LSL Q, Q0 = 0	0000	0010 0000	0000 0111
	3: Shift Div right	0000	0001 0000	0000 0111
2	1: Rem = Rem - Div	0000	0001 0000	0111 0111
	2b: Rem < 0 \Rightarrow +Div, LSL Q, Q0 = 0	0000	0001 0000	0000 0111
	3: Shift Div right	0000	0000 1000	0000 0111
3	1: Rem = Rem - Div	0000	0000 1000	0111 1111
	2b: Rem < 0 \Rightarrow +Div, LSL Q, Q0 = 0	0000	0000 1000	0000 0111
	3: Shift Div right	0000	0000 0100	0000 0111
4	1: Rem = Rem - Div	0000	0000 0100	0000 0011
	2a: Rem \geq 0 \Rightarrow LSL Q, Q0 = 1	0001	0000 0100	0000 0011
	3: Shift Div right	0001	0000 0010	0000 0011
5	1: Rem = Rem - Div	0001	0000 0010	0000 0001
	2a: Rem \geq 0 \Rightarrow LSL Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

Hardware da Divisão

Multiplicação

Divisão

Versão refinada do hardware da divisão:

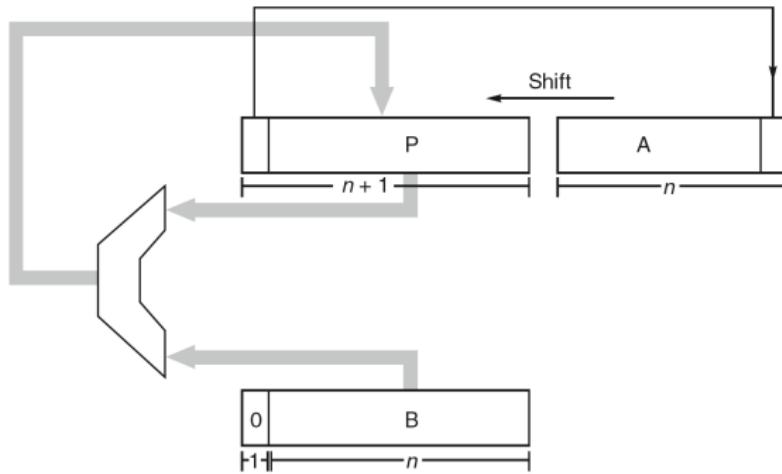


Algoritmo da Divisão

Multiplicação

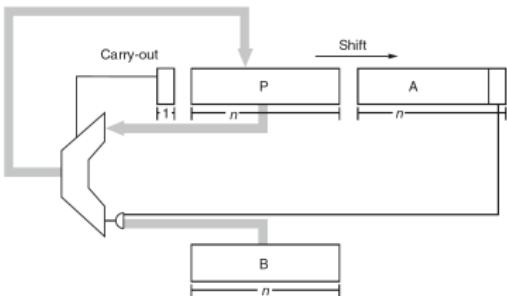
Divisão

Versão refinada do hardware da divisão



O registrador P contém o Resto, A o Quociente e B o Divisor. A é inicializado com o Dividendo. Antes de começar, o algoritmo deve verificar se o divisor é zero.

Restoring Division Algorithm



- O Quociente $a_{n-1}a_{n-2}\dots a_0$ e o divisor $b_{n-1}b_{n-2}\dots b_0$ são colocados nos registradores A e B.
- O registrador A é inicializado com o Dividendo.
- Cada passo da divisão tem 2 partes:
 - (i) Desloque o par de registradores PA 1 bit para a esquerda
 - (ii) Execute $P = P - B$
 - (iii) Se P resultante é negativo, então torne 0 o bit menos significativo de A, senão 1
 - (iv) Se P resultante é negativo, restabeleça P fazendo $P = P + B$

Exemplo de Divisão - Restoring

Multiplicação

Divisão

Execução de $14_{10} \div 3_{10} = 1110_2 \div 0011_2 = 4_{10} = 0100_2$ com resto $2_{10} = 0010_2$

P	A	
00000	1110	Divide $14 = 1110_2$ by $3 = 11_2$. B always contains 0011_2 .
00001	110	step 1(i): shift.
<u>-00011</u>		step 1(ii): subtract.
-00010	1100	step 1(iii): result is negative, set quotient bit to 0.
00001	1100	step 1(iv): restore.
00011	100	step 2(i): shift.
<u>-00011</u>		step 2(ii): subtract.
00000	1001	step 2(iii): result is nonnegative, set quotient bit to 1.
00001	001	step 3(i): shift.
<u>-00011</u>		step 3(ii): subtract.
-00010	0010	step 3(iii): result is negative, set quotient bit to 0.
00001	0010	step 3(iv): restore.
00010	010	step 4(i): shift.
<u>-00011</u>		step 4(ii): subtract.
-00001	0100	step 4(iii): result is negative, set quotient bit to 0.
00010	0100	step 4(iv): restore. The quotient is 0100_2 and the remainder is 00010_2 .

Non-Restoring Division Algorithm

Multiplicação

Divisão

- O Quociente $a_{n-1}a_{n-2}\dots a_0$ e o divisor $b_{n-1}b_{n-2}\dots b_0$ são colocados nos registradores A e B.
- O registrador P é inicializado com o Dividendo.
- Se P é negativo
 - (i) Desloque o par de registradores PA 1 bit para a esquerda
 - (ii) Execute $P = P + B$
- Senão
 - (i) Desloque o par de registradores PA 1 bit para a esquerda
 - (ii) Execute $P = P - B$
 - (iii) Se P é negativo, configure o bit menos significativo de A para 0, senão 1.
- No final, se P é negativo, deve-se executar $P = P + B$. P conterá o Resto.

Exemplo de Divisão - Non Restoring

Multiplicação

Divisão

Execução de $14_{10} \div 3_{10} = 1110_2 \div 0011_2 = 4_{10} = 0100_2$ com resto $2_{10} = 0010_2$

00000	1110	Divide $14 = 1110_2$ by $3 = 11_2$. B always contains 0011_2 .
00001	110	step 1(i-b): shift.
+11101		step 1(ii-b): subtract b (add two's complement).
11110	1100	step 1(iii): P is negative, so set quotient bit to 0.
11101	100	step 2(i-a): shift.
+00011		step 2(ii-a): add b.
00000	1001	step 2(iii): P is nonnegative, so set quotient bit to 1.
00001	001	step 3(i-b): shift.
+11101		step 3(ii-b): subtract b.
11110	0010	step 3(iii): P is negative, so set quotient bit to 0.
11100	010	step 4(i-a): shift.
+00011		step 4(ii-a): add b.
11111	0100	step 4(iii): P is negative, so set quotient bit to 0.
+00011		Remainder is negative, so do final restore step.
00010		The quotient is 0100_2 and the remainder is 0010_2 .

Signed Division

Multiplicação

Divisão

Em geral, a divisão com sinal é feita convertendo os operandos para números não negativos e tornar o quociente negativo se os sinais do dividendo e do divisor não coincidirem. Apesar deste método de divisão ser mais lento, a divisão é menos frequente que a multiplicação e, portanto, causa menos impacto que a multiplicação.

Leitura Recomendada

Multiplicação

Divisão

- Leia as seções 3.3 (*Multiplication*), 3.4 (*Division*), A.5 (*Constructing a Basic Arithmetic Logic Unit*) e A.6 (*Faster Addition: Carry Lookahead*) do livro *Computer Organization and Design - The Hardware/Software Interface ARM Edition*, de Hennessy e Patterson.
- Leia o Apêndice J (*Computer Arithmetic*) do livro *Computer Architecture - A Quantitative Approach*, de Hennessy e Patterson. Este Apêndice está disponível em https://www.elsevier.com/__data/assets/file/0006/795732/Hennessy_References_Appendices_V1.zip. Veja descrição em <https://www.elsevier.com/books-and-journals/book-companion/9780128119051>.

Referências

Multiplicação

Divisão

-  D. Patterson and J. Hennessy.
Computer Organization and Design ARM Edition: The Hardware Software Interface.
The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science, 2016.
 -  J. Wakerly.
Digital Design: Principles and Practices.
Pearson Education, Incorporated, 5 edition, 2018.
- [1] [2]

Obrigado!



Universidade de São Paulo



DEPARTAMENTO DE ENGENHARIA DE
COMPUTAÇÃO E SISTEMAS DIGITAIS

PCS

