Aritmética Binária

Pedroni – Capítulo 3

Prof. Odilson Tadeu Valle

Instituto Federal de Santa Catarina – IFSC
Campus São José
odilson@ifsc.edu.br

Conteúdo programático

1. Adição sem sinal
2. Adição e subtração com sinal
3. Princípios da divisão e multiplicação
4. Operações de deslocamento (SHIFT)
5. Princípios da divisão e multiplicação utilizando deslocamento
6. Multiplicação SEM sinal
7. Multiplicação COM sinal
8. Divisão SEM sinal
9. Divisão COM sinal

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Adição sem sinal

- A adição de números posicionais segue sempre a mesma regra: soma-se dígito a dígito, iniciando-se do menos significativo e, porta-se o vai um para o dígito mais a esquerda.
- Exemplos:

Decimal	Octal	Hexadecimal	Binário
1 1	1 1	0 0	1 0 1 1
77	77	77	1011
56	56	56	1011

Adição sem sinal

- A adição de números posicionais segue sempre a mesma regra: soma-se dígito a dígito, iniciando-se do menos significativo e, porta-se o vai um para o dígito mais a esquerda.
- Exemplos:

Decimal	Octal	Hexadecimal	Binário
1 1	1 1	0 0	1 0 1 1
77	77	77	1011
56	56	56	1011
133	155	CD	10110

Adição sem sinal

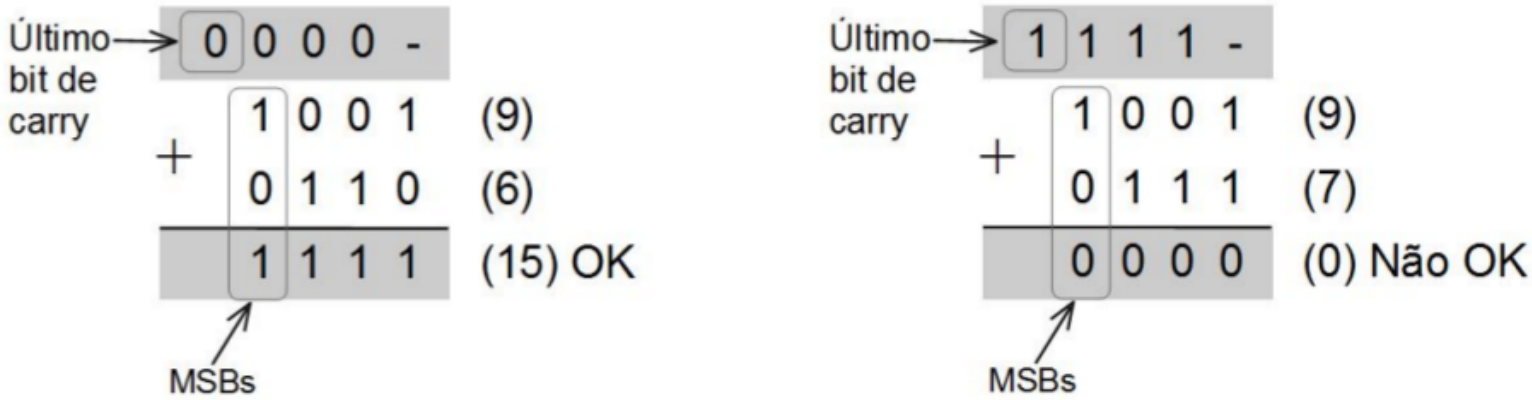
Ex:

$$\begin{array}{rl} 1101 & 4 \text{ bits} \\ +1011 & 4 \text{ bits} \\ \hline 11000 & 5 \text{ bits} \end{array}$$

- Normalmente computadores armazenam somente N bits, ou seja, pode ocorrer *overflow* (estouro) como é o caso do exemplo acima com N = 4.

Adição sem sinal com entrada de N bits e saída de N bits

- Como pode ocorrer *overflow* deve-se ter critérios para detecção automática caso esse erro ocorra:
  i) se o último bit do *carry* for 1 ==> ocorreu *overflow*
  ii) baseado nos operandos: quando ambos MSB são 1 ou quando são diferentes e o resultado de sua soma é zero
- Exemplo:



Adição sem sinal com entradas de N bits e saída de N+1 bits

- Nunca ocorre *overflow*!

```
  0 0 0 0 -
+ 1 0 0 1   (9)
  0 1 1 0   (6)
-----------
  0 1 1 1 1 (15) OK
```

```
  1 1 1 1 -
+ 1 0 0 1   (9)
  0 1 1 1   (7)
-----------
  1 0 0 0 0 (16) OK
```

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Subtração

- Exemplo:

$$
\begin{array}{l}
{\scriptstyle 0\text{-}1\text{-}1\ 0}\ <== \textit{borrow}\ (\text{empresta-um}) \\
\ \ 1001\ (+9) \\
\text{-}0111\ (\text{-}7) \\
\hline
00010\ (+2)
\end{array}
$$

- Regras operatórias:
  - Sinais iguais: soma e conserva o sinal.
  - Sinais diferentes: subtrai e conserva o sinal do maior módulo.
- 9 - 12?

Subtração

- Exemplo:

$$\begin{array}{r} {\scriptstyle 0\text{-}1\text{-}1\ 0} \ \texttt{<==}\ \textit{borrow}\ \text{(empresta-um)} \\ 1001\ (+9) \\ \texttt{-}0111\ (\texttt{-}7) \\ \hline 00010\ (+2) \end{array}$$

- Regras operatórias:
  - Sinais iguais: soma e conserva o sinal.
  - Sinais diferentes: subtrai e conserva o sinal do maior módulo.
- 9 - 12?
- 12 - 9 e troca o sinal do resultado.

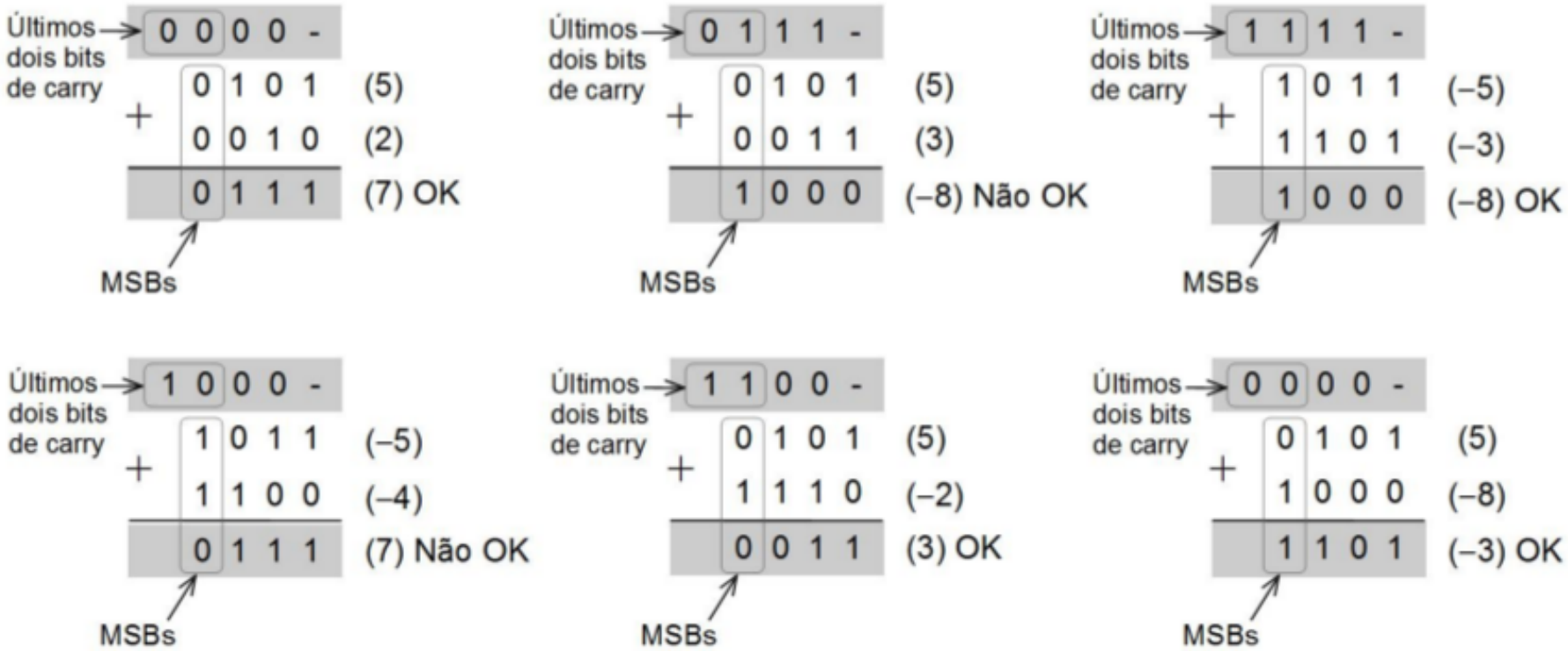
$$\begin{array}{r} 1100\ (12) \\ \texttt{-}1001\ (9) \\ \hline 0011\ (3)\ \texttt{==>}\ \texttt{-}3 = \texttt{-}0011 \end{array}$$

Subtração

- Na prática, ao implementar-se nos sistemas digitais, a subtração nos moldes anteriores NÃO é utilizada!
- Para executar subtração em circuitos lógicos, utiliza-se a representação em complemento de dois, procedendo-se a soma direta entre os vários elementos:
  - $a - b = a + (-b)$, sendo $-b$ o complemento de dois de b
- Exemplo: faça as operações aritméticas binárias em 8 bits dos equivalentes decimais:
  1. +7-5 =
  2. -22-48 =
  3. +125-78 =

Adição e subtração com sinal com entradas de N bits e saída de N bits

- Como detectar *overflow*:
  i) se os dois últimos bits de carry forem diferentes, então ocorreu *overflow*.
  ii) se os MSB dos operandos forem iguais e o MSB da soma for diferente deles, então ocorreu *overflow*.



- Outros exemplos.

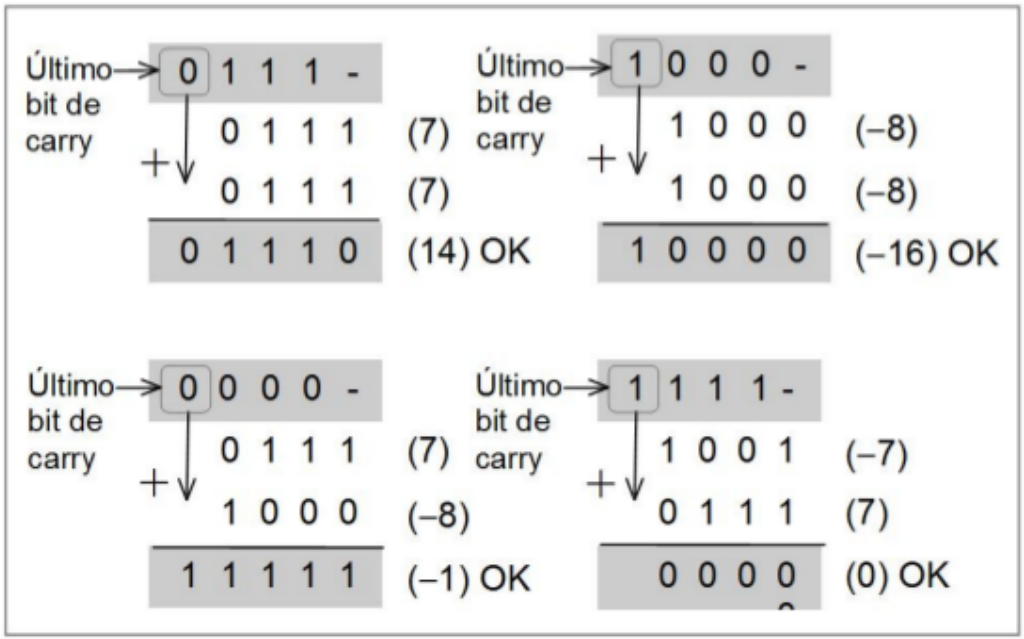
Adição e subtração com sinal com entradas de N bits e saída de N+1 bits

- Nunca ocorre *overflow*, MAS:

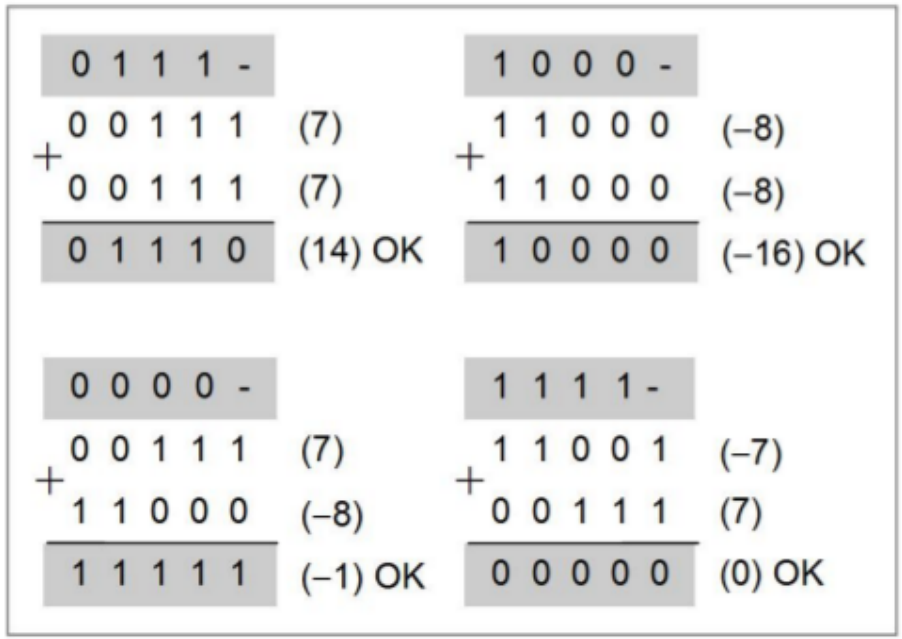
a) Se as entradas tiverem sinais diferentes (bit mais a esquerda), o MSB (N+1) do resultado, ou do carry, deve ser invertido;

b) Utiliza-se a chamada extensão do sinal.

a)



b)



Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Princípios do deslocamento de bits

- O que ocorre quando deslocamos 1 bit a direita de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 1100110 (102)?

Princípios do deslocamento de bits

- O que ocorre quando deslocamos 1 bit a direita de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 1100110 (102)? $==>$ 0110011 (51) $\surd$
  2. 1100111 (103)?

Princípios do deslocamento de bits

- O que ocorre quando deslocamos 1 bit a direita de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 1100110 (102)? ==> 0110011 (51) $\surd$
  2. 1100111 (103)? ==> 0110011 (51) $\approx$
  3. 110?
  4. 111?

  O que ocorre quando deslocamos 1 bit a esquerda de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 01100110 (102)?

Princípios do deslocamento de bits

- O que ocorre quando deslocamos 1 bit a direita de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 1100110 (102)? ==> 0110011 (51) $\surd$
  2. 1100111 (103)? ==> 0110011 (51) $\approx$
  3. 110?
  4. 111?

  O que ocorre quando deslocamos 1 bit a esquerda de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 01100110 (102)? ==> 11001100 (204) $\surd$
  2. 01100111 (103)?

Princípios do deslocamento de bits

- O que ocorre quando deslocamos 1 bit a direita de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 1100110 (102)? ==> 0110011 (51) $\surd$
  2. 1100111 (103)? ==> 0110011 (51) $\approx$
  3. 110?
  4. 111?

  O que ocorre quando deslocamos 1 bit a esquerda de uma determinada representação binária, mantendo-se o número de bits? Ex:
  1. 01100110 (102)? ==> 11001100 (204) $\surd$
  2. 01100111 (103)? ==> 11001110 (206) $\surd$
  3. 110?
  4. 111?

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

SHIFT lógico

- A palavra binária é deslocada um certo número de posições para a direita ou para a esquerda, as posições vazias são preenchida com zeros. Ex:
  1. 10111 SRL2 = 00101 (SRL: *Shift Right Logic n positions*)
  2. 10100 SRL2 = 00101
  3. 11101 SLL1 = 11010 (SLL: *Shift Left Logic n positions*)

SHIFT Aritmético

- A palavra binária é deslocada um certo número de posições para a direita ou para a esquerda.
- Quando deslocado para a direita, as posições vazias são preenchidas com o valor do bit original da extrema esquerda (bit de sinal).
- Quando deslocado para a esquerda são preenchidas com o valor do bit da extrema direita, seguindo o padrão do VHDL. Ex:
  1. 10101 SRA2 = 11101 (SRA: *Shift Right Arithmetic n positions*)
  2. 10101 SLA1 = 01011 (SLA: *Shift Left Arithmetic n positions*)

SHIFT Circulação (rotação)

- Semelhante ao SHIFT lógico, com a diferença que as posições vazias são preenchidas com os bits removidos. Ex:
  1. 11001 ROR2 = 01110 (ROR: *Rotate Right n positions*)
  2. 11001 ROL1 = 10011 (ROL: *Rotate Left n positions*)

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Divisão usando SHIFT lógico para números sem sinal

- Regra: Um deslocamento lógico de uma posição para a direita resulta na divisão do número por 2, duas posições por 4 e assim sucessivamente.
- Exemplos:
  1. 101110 ==> (1 posição) ==> 010111, está correto?
  2. 110101 ==> (1 posição) ==> 011010, está correto?
  3. 101110 ==> (2 posição) ==> 001011, está correto?
  4. 110101 ==> (2 posição) ==> 001101, está correto?

Divisão usando SHIFT aritmético para números com sinal

1. 01110 ==> 00111, está correto?
2. 00111 ==> 00011, está correto?
3. 10000 ==> 11000, está correto?
4. 11001 ==> 11100, está correto?

Multiplicação usando SHIFT lógico

- Regra: Um deslocamento lógico de uma posição para a esquerda resulta na multiplicação do número por 2, duas posições por 4 e assim sucessivamente.
- Exemplos:
  - Sem sinal:
    1. 01110 (14) ==> 11100 (28) $\surd$
    2. 11010 (26) ==> 10100 (20) $\times$
  - Com sinal
    1. 00111 (+7) ==> 01110 (+14) $\surd$
    2. 01110 (+14) ==> 11100 (-4) $\times$
    3. 11000 (-8) ==> 10000 (-16) $\surd$
    4. 10000 (-16) ==> 00000 (0) $\times$

       Observe que quando os primeiros bits, do número original e do resultado, forem iguais o resultado é correto e incorreto em caso contrário.

Multiplicação usando SHIFT e uma saída mais larga

- Na prática, dobra-se o número de bits (2N) do resultado, garantindo assim a não mais ocorrência do *overflow* com até N deslocamentos.
- Números sem sinal com 4 deslocamentos, SHIFT lógico, ex:
  1. 0011 (3) = 0000 0011 ==> (4 pos) 0011 0000 (48)
  2. 1000 (8) ==> 0000 1000 (8) ==> 1000 0000 (128)
- Números sinalizados com 4 deslocamentos, SHIFT aritmético, ex:
  1. 0011 (+3) ==> 0000 0011 (+3) ==> 0011 0000 (+48)
  2. 1000 (-8) ==> 1111 1000 (-8) ==> 1000 0000 (-128)
  3. 1111 (-1) ==> 1111 1111 (-1) ==> 1111 0000 (-16)

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

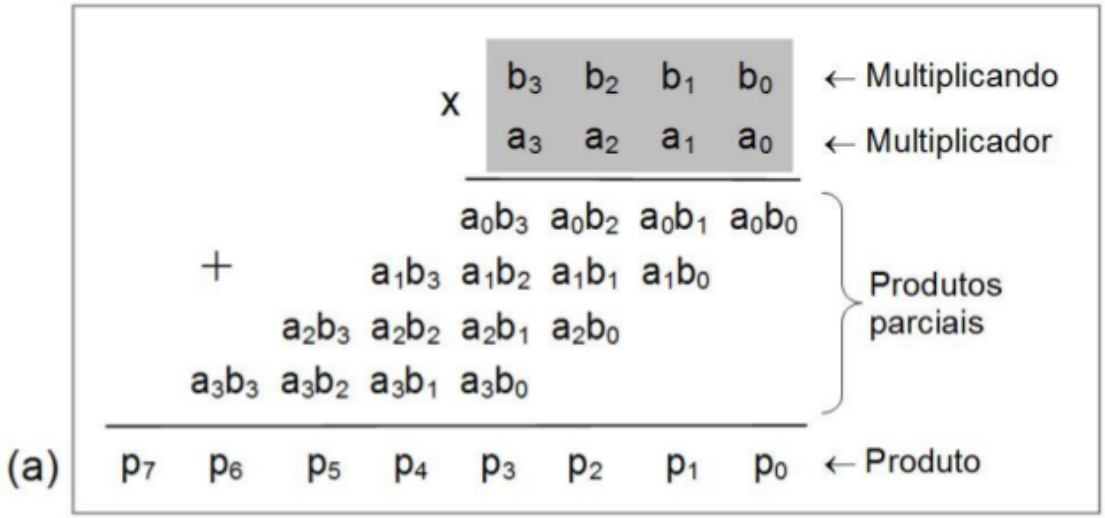
7 Multiplicação COM sinal

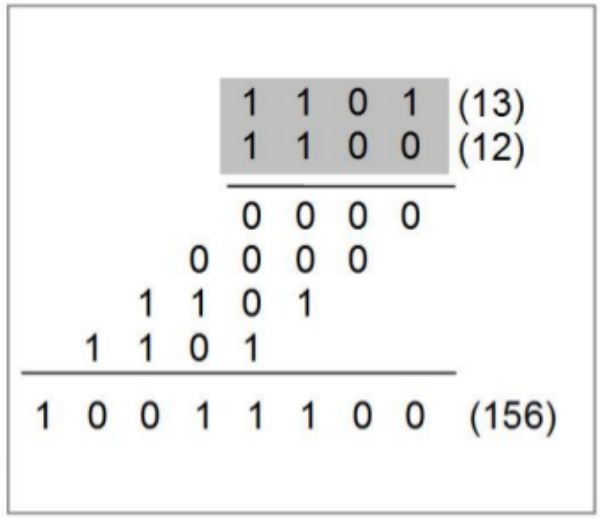
8 Divisão SEM sinal

9 Divisão COM sinal

Multiplicação sem sinal

- N bits para as entradas e 2N bits para as saídas, prevenindo a ocorrência de *overflow*. Ex:





- Qualquer algoritmo de multiplicação é derivado do algoritmo utilizado para a resolução do exemplo anterior.
- Quando faz-se uma implementação no hardware, do zero, utiliza-se esse algoritmo.

Multiplicação sem sinal

- Por outro lado a multiplicação pode ser efetuada usando somente operações de adição e deslocamento, abordagem empregada em computadores, já que, ele tem uma ALU (*Arithmetic Logic Unit*) que faz adições enquanto a unidade de controle pode facilmente deslocar os registradores de dados conforme o necessário.
- Exemplo, 1001×1100: 4 bits $\equiv$ 4 iterações (está correto?)

Iteração	Procedimento	Multiplicando	Produto Esquerda Direita
0	Inicialização (ProdDireita recebe Multiplicador)	1001	0000 110 **0**
1	Bit=0 → Sem operação Deslocamento lógico p/ a direita		0000 1100 0000 011 **0**
2	Bit=0 → Sem operação Deslocamento lógico p/ a direita		0000 0110 0000 001 **1**
3	Bit=1 → ProdEsquerda + Multiplicando Deslocamento lógico p/ a direita		1001 0011 0100 100 **1**
4	Bit=1 → ProdEsquerda + Multiplicando Deslocamento lógico p/ a direita		1101 1001 0110 1100

- Em resumo: se o bit de produto da extrema direita for 1, o multiplicando é adicionado a metade esquerda do produto, em seguida o registrador de produto é deslocado uma posição para a direita, entretanto, se o bit for 0, deve ocorrer somente a operação de deslocamento ($0 \times$ *qualquer coisa* $= 0$).

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

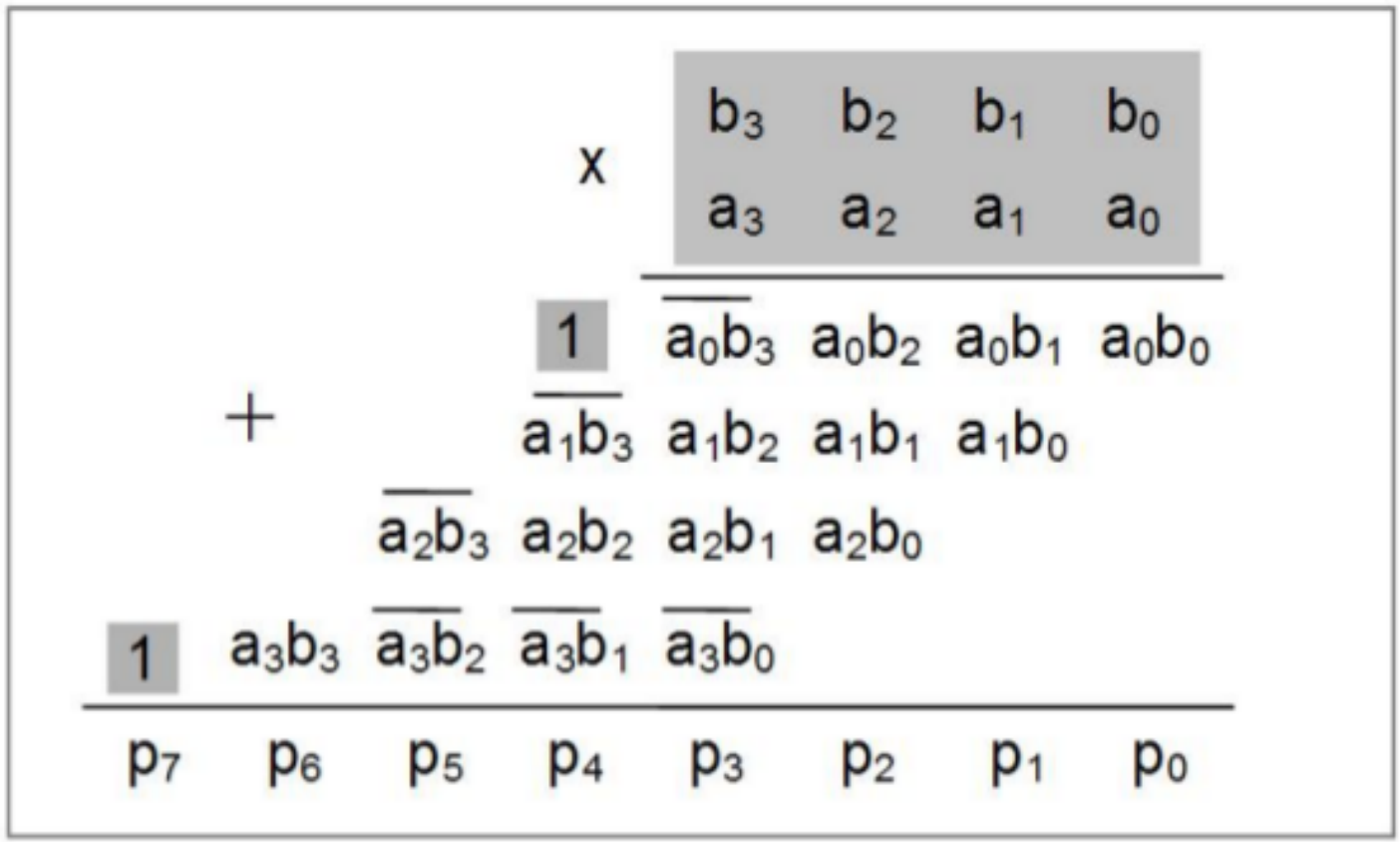
7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Multiplicação com sinal

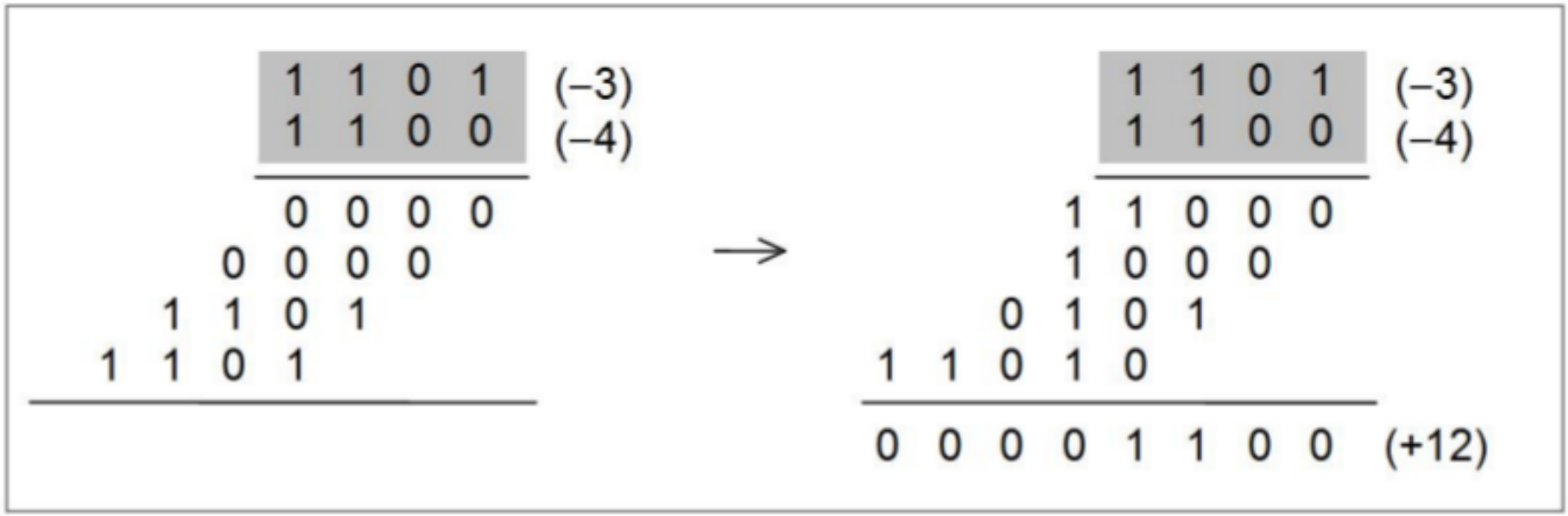
- Algoritmos para a multiplicação com sinal também são derivados do algoritmo básico.
- Na figura a seguir é ilustrado o multiplicador Baugh-Wooley modificado.



- Observe que foram inseridos dois 1's e alguns valores são invertidos: tratamento do sinal.

Multiplicação com sinal

- Por exemplo, vamos empregar o algoritmo acima para multiplicar 1101 e 1100:



Algoritmo de Booth

- Algoritmo baseado em ALU, utilizando adição/subtração e shift aritmético. 01110×10010.
- 5 bits nos operandos leva a necessidade de 5 iterações e 10 bits de saída:

Iteração	Procedimento	Multiplicando	Produto Esquerda	Produto Direito	Produto Extra
0	Inicialização (ProdDireita recebe Multiplicador)	10010	00000	0111 0	0
1	Bits=00 → Sem operação Deslocamento aritmético p/ a direita		00000 00000	0111 0 0011 1	0 0
2	Bits=10 → ProdEsquerda – Multiplicando Deslocamento aritmético p/ a direita		01110 00111	0011 1 0001 1	0 1
3	Bits=11 → Sem operação Deslocamento aritmético p/ a direita		00111 00011	0001 1 1000 1	1 1
4	Bits=11 → Sem operação Deslocamento aritmético p/ a direita		00011 00001	1000 1 1100 0	1 1
5	Bits=01 → ProdEsquerda + Multiplicando Deslocamento aritmético p/ a direita		10011 11001	11000 11100	

- Em resumo: se os bits da extrema direita forem 10, o multiplicando deve ser subtraído da metade esquerda do registrador de produto e o resultado deve ser deslocado aritmeticamente para a direita. Se os bits forem 01 então o multiplicando deve ser somado à metade esquerda do registrador de produto e o resultado deslocado aritmeticamente para a direita. Por fim, se os bits forem 00 ou 11, então deve ocorrer somente o deslocamento aritmético para a direita.

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Divisão sem sinal

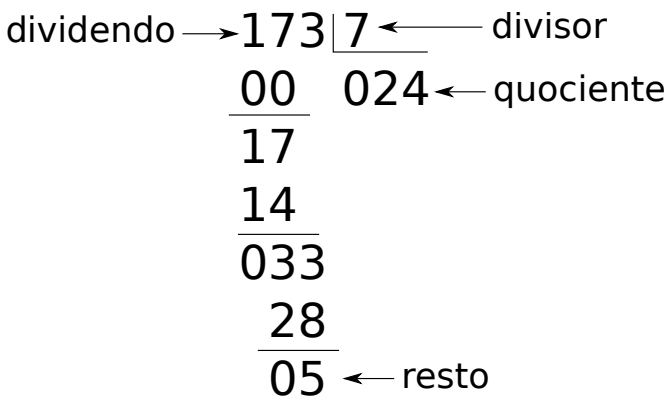
- Algoritmo básico, 1101 (13) dividido por 0101 (5) (para relembrar):

decimal (173/7) binário (13/5)

Divisão sem sinal

- Algoritmo básico, 1101 (13) dividido por 0101 (5) (para relembrar):

decimal (173/7)



binário (13/5)

Divisão sem sinal

- Algoritmo básico, 1101 (13) dividido por 0101 (5) (para relembrar):

decimal (173/7)

```
dividendo → 173 | 7   ← divisor
            00  | 024 ← quociente
            17
            14
            033
             28
             05 ← resto
```

binário (13/5)

```
dividendo → 0001101 | 0101 ← divisor
            0000    | 0010 ← quociente
             0011
             0000
              0110
              0101
               0011 ← resto
```

- Observações:
  - Se forem usados N bits para representar as entradas, então, novamente, serão necessários 2N bits para representar as saídas: 4 para o quociente e 4 para o resto.
  - O dividendo deve ser incrementado com 0's a esquerda até possuir um total de 2N-1 bits

Divisão sem sinal

- Do ponto de vista de hardware, é mais difícil implementar divisores dedicadas do que multiplicadores dedicados.
- Por essa razão, a abordagem mais comum é empregar algoritmos baseados em ALU para a divisão, os quais utilizam operações de adição, subtração e deslocamento.
- Figura: 1101(13)/0101(5). Deve-se ter N iterações e o registrador de resto 2N bits.

Iteração	Procedimento	Divisor	Resto Esquerda Direita
0	Inicialização (RestoDireita recebe Dividendo) Deslocamento p/ a esquerda com entrada de '0'	0101	0000 1101 0001 1010
1	RestoEsquerda – Divisor Bit=1 → RestoEsquerda + Divisor Bit=1 → Desloc. p/ a esquerda com entrada de'0'		1 100 1010 0001 1010 0011 0100
2	RestoEsquerda – Divisor Bit=1 → RestoEsquerda + Divisor Bit=1 → Desloc. p/ a esquerda com entrada de'0'		1 110 0100 0011 0100 0110 1000
3	RestoEsquerda – Divisor Bit=0 → Sem operação Bit=0 → Desloc. p/ a esquerda com entrada de'1'		0 001 1000 0001 1000 0011 0001
4	RestoEsquerda – Divisor Bit=1 → RestoEsquerda + Divisor Bit=1 → Desloc. p/ a esquerda com entrada de'0'		1 110 0001 0011 0001 0110 0010
	Deslocamento RestoEsquerda p/ direita com '0'		0011 0010

Conteúdo programático

1 Adição sem sinal

2 Adição e subtração com sinal

3 Princípios da divisão e multiplicação

4 Operações de deslocamento (SHIFT)

5 Princípios da divisão e multiplicação utilizando deslocamento

6 Multiplicação SEM sinal

7 Multiplicação COM sinal

8 Divisão SEM sinal

9 Divisão COM sinal

Divisão com sinal

- A divisão com sinal é normalmente efetuada como se os números não tivessem sinal.
- Isso implica que, em primeiro lugar, os números negativos devem passar por complemento de dois. Além disso, se o dividendo e o divisor tiverem sinais diferentes, então o quociente deve ser negativado (complemento de dois) e o resto também.
- Ex: $1001(-7)/0011(+3)$

Divisão com sinal

- A divisão com sinal é normalmente efetuada como se os números não tivessem sinal.
- Isso implica que, em primeiro lugar, os números negativos devem passar por complemento de dois. Além disso, se o dividendo e o divisor tiverem sinais diferentes, então o quociente deve ser negativado (complemento de dois) e o resto também.
- Ex: $1001(-7)/0011(+3)$

Complemento de dois de 1001 é 0111

```
0000111 | 0011
0000      0010 ← quociente
 0001
 0000
  0011
  0011
   0001 ← resto
```

complemento de dois de 0010 é 1110 (-2)
complemento de dois de 0001 é 1111 (-1)

- Operações com ponto flutuante: seções 3.8, 3.9 e 3.10 do Pedroni.