

Introdução à Ciência da Computação
Introdução à Engenharia de Computação

Aritmética Binária com Números com Sinal



Profa. Ana Pernas
Prof. Giovani Farias
Profa. Lisane Brisolara
Prof. Rafael Soares



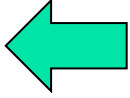
Aritmética Binária

- Soma
- Subtração
- Multiplicação
- Divisão

Todas as 4 operações
podem ser feitas via
Soma



Aritmética Binária

- **Soma**
- **Subtração**  **Subtração via soma**
- Multiplicação
- Divisão



Adição binária (revisão)

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ e vai-um}$$

$$1 + 1 + 1 = 1 \text{ e vai-um}$$



Aritmética com sinal

- **Usaremos C-2** para representar números com sinal
- **Faremos as 2 operações via soma**
 - Nros de **mesmo sinal** são somados
 - Nros de **sinais diferentes** são subtraídos



C-2 (Revisão)

- **Usaremos C-2** para representar Nros com sinal
 - **Nros positivos:** basta converter de decimal p/ binário e completar com zeros na esquerda até atingir o nro de bits adotado
 - **Nros negativos:** pegar a representação do nro binário positivo em C-2, inverter todos os bits (C-1) e somar 1.



Complemento de 2 (C-2)

Ex: Usando representação de **8 bits** (1 bit de sinal + 7 bits de módulo)

$+33_{(\text{base } 10)}$ e $-33_{(\text{base } 10)}$

+33: 00100001
 32 1

-33?: C-1 e soma 1

```
00100001
11011110
+ 1
-----
-33: 11011111
```



C-2 (Revisão)

Ex: Usando representação de **8 bits**

Se tivermos um nro negative em C-2 como saber qual o seu módulo?

11011111

Tem 2 formas:

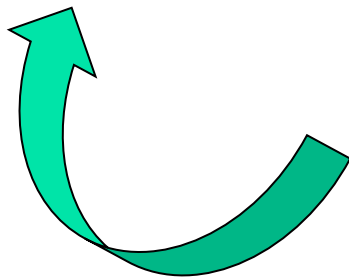
- 1) usando C-1 + 1 -> simetria
- 2) Usando pesos base 2, mas com bit de sinal com peso negativo

C-2 (Revisão)

Ex: Usando representação de **8 bits**

Se tivermos um nro negative em C-2 como saber qual o seu módulo?

11011111



-33

C-1 e soma 1

11011111
00100000
+ 1

+33: 00100001



C-2 (Revisão)

Ex: Usando representação de **8 bits**

Se tivermos um nro negative em C-2 como saber qual o seu módulo?

11011111

$$1+2+4+8+16+64+(-128) = \mathbf{-33}$$



Bit de sinal com
peso negativo

Somamos os pesos (potências de 2) , mas bit de sinal terá peso negativo



Soma de Nros com sinal

- Mesmo sinal
 - 2 nros positivos
 - 2 nros negativos
- Sinais diferentes
 - Subtração: $24-13 \Rightarrow +24+(-13)$



Soma de Nros com sinal

- Mesmo sinal: Nros **positivos**

Ex: 6 bits

$$+13 + (+13) = ?$$

Colocar operandos em C-2 com 6 bits

$$\begin{array}{r} \overset{1}{} \overset{1}{} \overset{1}{} \\ 001101 \\ + 001101 \\ \hline 011010 \end{array} \quad \rightarrow \quad 26_{10}$$



Soma de Nros com sinal

- Mesmo sinal: Nros **Negativos**

Ex: 6 bits

$$-13 + (-13) = ?$$

Colocar operandos em C-2 com 6 bits

+13:	001101
	110010
	+1
	<hr/>
-13:	110011

Pesos: $1+2+16+(-32) = -13$

Soma de Nros com sinal

- Mesmo sinal: Nros **Negativos**

Ex: 6 bits

$$-13 + (-13) = ?$$

Colocar operandos em C-2 com 6 bits

+13:	001101		
	110010		
	+1		
	<hr/>		
-13:	110011		
		Operando	
		↗	
			110011
			+ 110011
			<hr/>

Soma de Nros com sinal

- Mesmo sinal: Nros **Negativos**

Ex: 6 bits

$$-13 + (-13) = ?$$

Colocar operandos em C-2 com 6 bits

+13:	001101		
	110010		
	+1		
-13:	110011		
		Operando	
			ignorar
			(1) 1 1 1
			110011
		+	110011
			<u>100110</u>
			2+4+(-32) = - 26
			→ -26 ₁₀

Resultado é 1 nro negativo!



Soma de Nros com sinal

- Sinais diferentes

Ex: 6 bits

$$+13 + (-13) = ?$$

Colocar operandos em C-2 com 6 bits

ignorar

①	1	1	1	1	1			
	0	0	1	1	0	1		+13
+	1	1	0	0	1	1		-13
<hr/>								
	0	0	0	0	0	0		

Sinais diferentes
somamos, mas o resultado
é de uma subtração!



Overflow

- Ao realizar operações aritméticas com operandos **com sinal**, podemos ter um estouro na capacidade de representação -> **overflow**
- Pois usamos um **limitado e fixo número de bits** na representação tanto de operandos quanto do resultado das operações matemáticas

Podemos não conseguir realizar uma operação porque os operandos não são representáveis com um dado nro de bits OU o resultado não é representável!

Lembrem-se das faixas de representação...



Overflow: Exemplo

$$65 + 67 = 132$$

- É possível realizar usando 8 bits?
 - Conseguimos representar os 2 operandos
 - Faixa do C-2 c/ 8 bits: de -128 até +127
 - Mas, **não conseguimos representar o resultado**
 - 132 excede a faixa de valores representáveis com 8 bits, cujo maior valor positivo é +127

Se aumentarmos para 9 bits a representação já conseguiríamos fazer a operação! No papel podemos fazer isso, mas se esta for uma limitação do processador, não tem jeito!



Overflow: Como identificar?

- A soma de 2 números positivos tem de dar outro número positivo
 - Se o bit de sinal der diferente de 0, neste caso, é porque ocorreu overflow
- A soma de 2 números negativos deve dar um número também negativo
- A soma de nros com sinal diferentes, não dá overflow pois o resultado será a subtração dos nros e portanto um nro menor que os operandos (e portanto representável).



Overflow: Exemplo (ilustração)

Ex: 8 bits

$$65 + 67 = +132$$

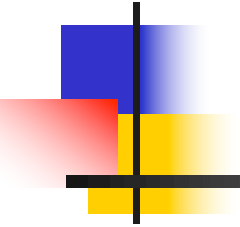
$$\begin{array}{r} \textcolor{green}{1} \textcolor{green}{1} \textcolor{green}{1} \\ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \\ + 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \end{array}$$



Overflow de
complemento de 2

Opa! Somamos **2** nros
positivos e a resposta foi um
nro **negativo**!

Mais exemplos de aritmética binária com sinal





Sinais iguais

Ex: $29 + (+7) = 36$

- 1) Converter operandos para binário
- 2) Colocá-los em C-2 (no ex. 6 bits)



Sinais iguais

Ex: 29+7

Usando c-2 com 6 bits:

011101 + (000111)

$$\begin{array}{r} \begin{array}{cccccc} & 1 & 1 & & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & \\ + & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & \end{array} & \begin{array}{r} 29 \\ +7 \\ \hline 36 \end{array} \end{array}$$

Estouro de representação:

36 não é represável com 6 bits (-32 até +31)



Sinais diferentes

$$29 - 7 = 22$$

$$+29 +(-7) = 22$$

Agora: Vamos fazer a subtração via soma em binário usando nros com sinal?

Sinais diferentes

- 1) Converter para binário os operandos
- 2) Colocá-los em c-2 (no ex. 6 bits)

000111 → 111000 → 111001 -7_{10}

$$\begin{array}{r} \begin{array}{cccccc} & 1 & 1 & 1 & & 1 \\ & 0 & 1 & 1 & 1 & 0 & 1 \\ + & 1 & 1 & 1 & 0 & 0 & 1 \\ \hline \end{array} & \begin{array}{r} 29 \\ -7 \\ \hline 22 \end{array} \\ \textcircled{1} & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Bit de transporte é
ignorado



Resumo: Soma com nros com sinal

- Números de mesmo sinal soma
- Números com sinais diferentes subtrai

Exemplos com 4 bits:

Num de mesmo sinal -> **Soma**

$$+5 + 2 = +7$$

Num de sinal diferente -> **Subtrai**

$$+5 + (-2) = +3$$



Resumo: Soma com nros com sinal

- Números de mesmo sinal soma
- Números com sinais diferentes subtrai

Exemplos com 4 bits:

Num de mesmo sinal -> **Soma**

$$+5 + 2 = +7$$

sinal

$$\begin{array}{r} \boxed{0}101 \\ + \boxed{0}010 \\ \hline 0111 \end{array}$$

Num de sinal diferente -> **Subtrai**

$$+5 + (-2) = +3$$

$$\begin{array}{r} \overset{1}{\boxed{0}}101 \\ + \overset{1}{\boxed{1}}110 \quad -2 \text{ em C-2} \\ \hline 0011 \end{array}$$

Ilustração das operações de soma e subtração via soma



Exercícios

Fazer operações de **soma** em binário com números em C-2 de 8 bits

$$(-9) + (+8) =$$

$$1 + (-1) =$$

$$(-1) + (-1) =$$

$$(127) + (-128) =$$



Exercícios

$$(-9) + (+8) = -1$$

+8: 00001000

+9: 00001001

-9: 11110110+1= 11110111

$$\begin{array}{r} 00001000 \\ + 11110111 \\ \hline 11111111 \end{array}$$



Exercícios

$$(-9) + (+8) = -1$$

+8: 00001000

+9: 00001001

-9: 11110110+1= 11110111

$$\begin{array}{r} 00001000 \\ + 11110111 \\ \hline 11111111 \end{array}$$

Verificação: 11111111 ?

00000001 = +1

Entao:

11111111 = -1

Pelos pesos:

$$1+2+4+8+16+32+64+(-128)=-1$$



Exercícios

$$(1) + (-1) =$$

+1: 0000001

-1: 1111111

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 00000001 \\ + 11111111 \\ \hline \textcircled{1} 00000000 \end{array}$$

Bit de transporte é
ignorado



Exercícios

$$(-1) + (-1) =$$

+1: 0000001

-1: 11111111

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 11111111 \\ +\ 11111111 \\ \hline 11111110 \end{array}$$



Exercícios

$$(-1) + (-1) =$$

+1: 0000001

-1: 11111111

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 11111111 \\ +\ 11111111 \\ \hline 11111110 \end{array}$$

Verificação: 11111110 ?

$$00000001 + 1 = 00000010 = +2$$

$$\text{Então: } 11111110 = -2$$

Pelos pesos:

$$2 + 4 + 8 + 16 + 32 + 64 + (-128) = -2$$



Exercícios

$$(127) + (-128) =$$

+127: 01111111

-128: 10000000

$$\begin{array}{r} 01111111 \\ +10000000 \\ \hline 11111111 \end{array}$$

OBS: +128 não é representável em C-2 com 8bits, pois em binário 128 seria 1000 0000 o que teria módulo invadindo o bit de sinal!



Exercícios

$$(127) + (-128) =$$

$$+127: 01111111$$

$$-128: 10000000$$

$$\begin{array}{r} 01111111 \\ +10000000 \\ \hline 11111111 \end{array}$$

Verificação: 11111111 ?

$$00000000 + 1 = 00000001 = +1$$

Então: 11111111 = -1

Pelos pesos:

$$1 + 2 + 4 + 8 + 16 + 32 + 64 + (-128) = -1$$

OBS: +128 não é representável em C-2 com 8bits, pois em binário 128 seria 1000 0000 o que teria módulo invadindo o bit de sinal!