

## 本节主题



# 加法和减法的实现

北京大学·慕课  
计算机组成  
制作人：陆俊林



# 二进制的加法



$$\begin{array}{r} 1101 \\ + 0110 \\ \hline 10011 \end{array}$$

被加数 A

加数 B

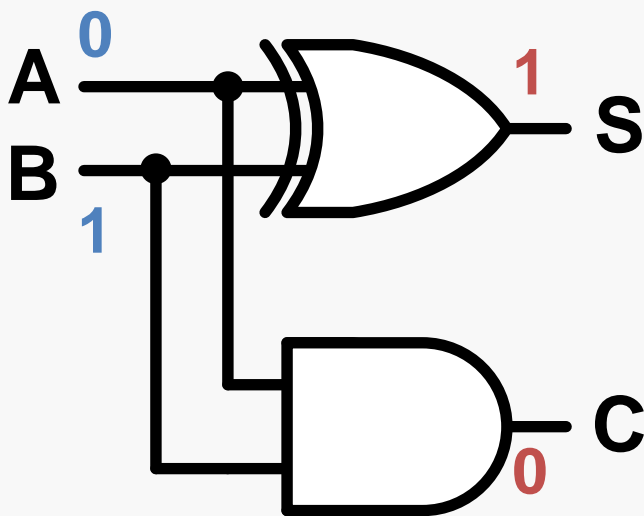
和 S

两个4-bit二进制数相加

1. 两个1-bit二进制数相加
2. 进位输入参与运算
3. 可以产生进位输出

# 半加器 ( Half Adder )

- 半加器的功能是将两个一位二进制数相加
  - 输入端口A、B
  - 输出端口S ( 和 )、C ( 进位 )

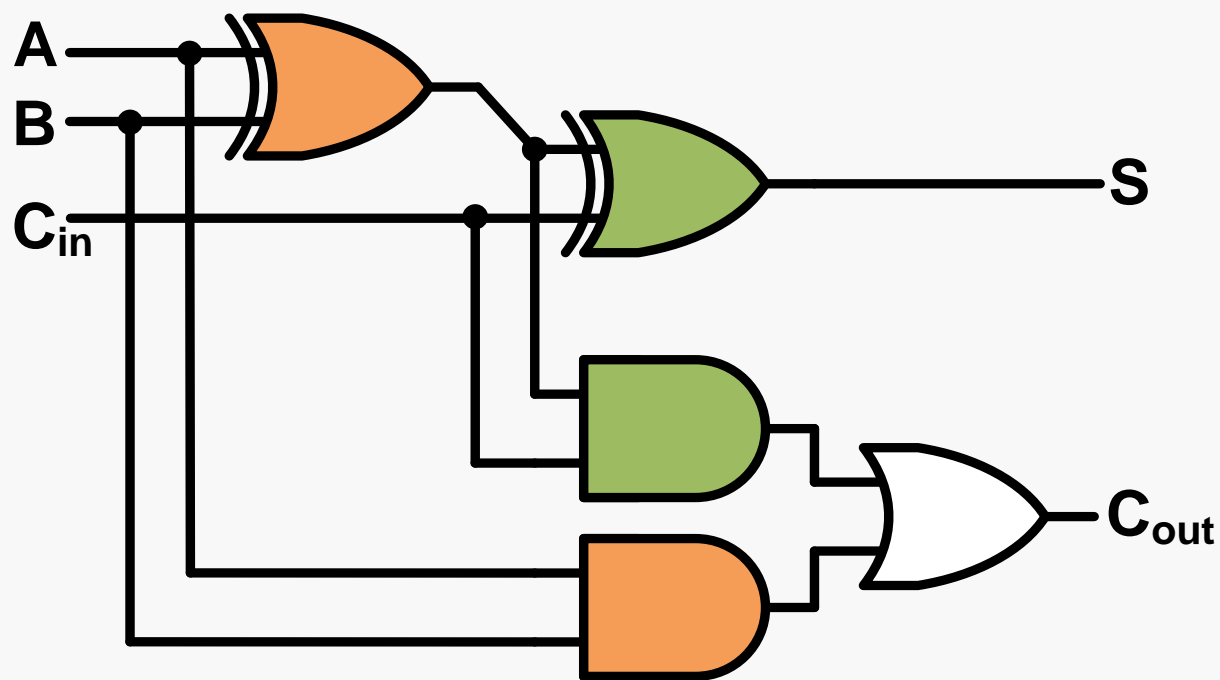


A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

# 全加器 ( Full Adder )

## 全加器由两个半加器构成

- 输入端口A、B、 $C_{in}$  ( 进位输入 )
- 输出端口S ( 和 )、 $C_{out}$  ( 进位输出 )



A	B	$C_{in}$	$C_{out}$	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

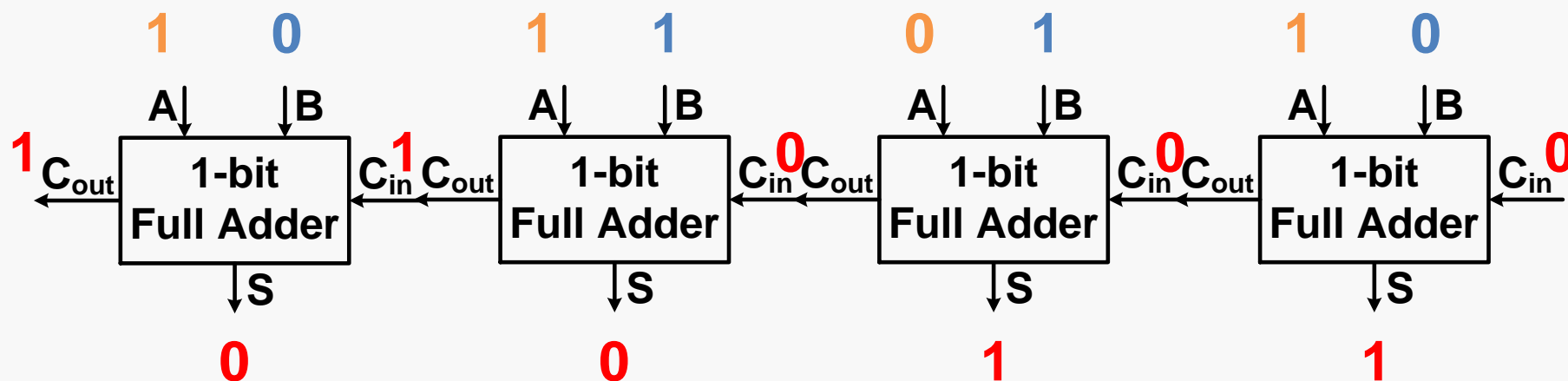
# 4-bit加法器

$$\begin{array}{r} \phantom{+} 1101 \\ + 0110 \\ \hline 10011 \end{array}$$

被加数 A  
加数 B  
进位C  
和 S

两个4-bit二进制数相加

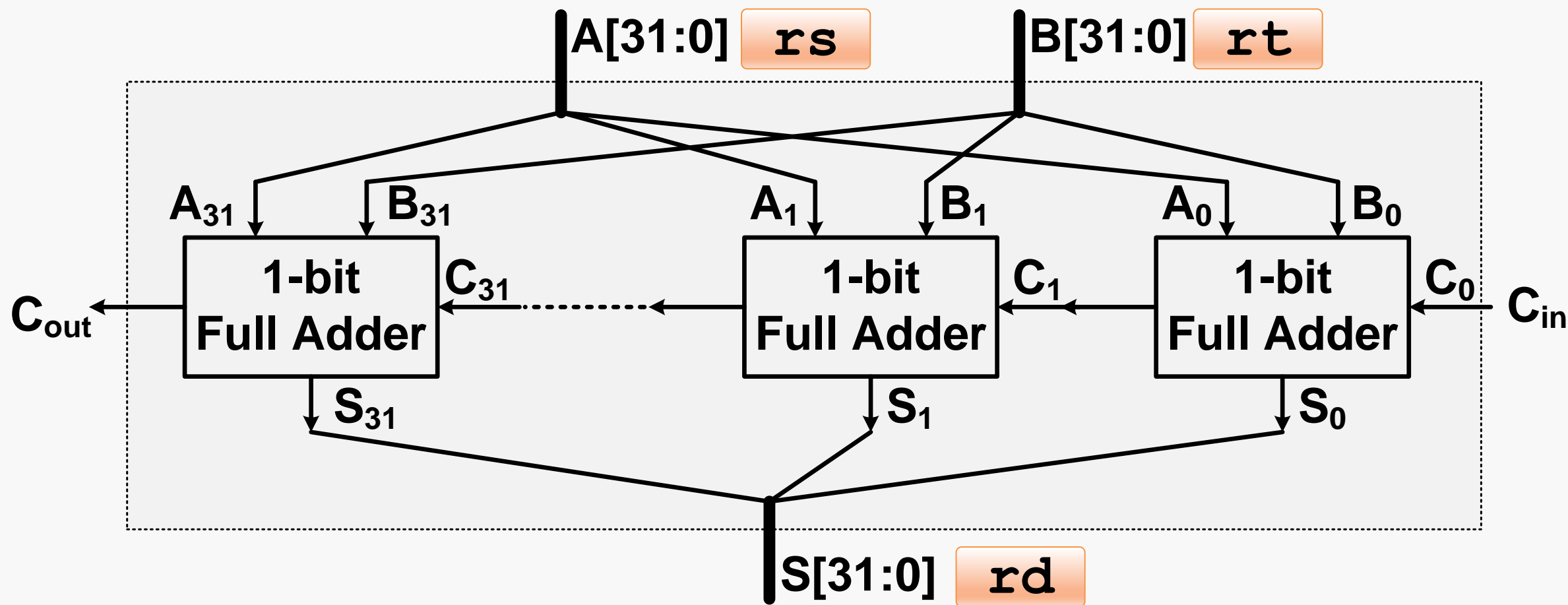
由四个全加器构成的4-bit加法器



# 加法运算的实现示例

▶ `add rd, rs, rt`

▶ `addu rd, rs, rt`



# 检查加法运算结果是否溢出



## “溢出” ( overflow )

- 运算结果超出了正常的表示范围

## “溢出” 仅针对有符号数运算

- 两个正数相加，结果为负数
- 两个负数相加，结果为正数

$$\begin{array}{r} 0011 \\ + 0101 \\ \hline 1000 \end{array}$$

无符号数 :  $3 + 5 = 8$

有符号数 :  $3 + 5 = (-8)$

# “进位” 和 “溢出” 示例



## 注意区分 “进位” 和 “溢出”

- 有 “溢出” 时，不一定有 “进位”
- 有 “进位” 时，不一定有 “溢出”

无符号数： $3 + 5 = 8$

有符号数： $3 + 5 = (-8)$

$$\begin{array}{r} 0011 \\ + 0101 \\ \hline 01000 \end{array}$$

有 “溢出”，无 “进位”

无符号数： $14 + 12 = 10$

有符号数： $(-2) + (-4) = (-6)$

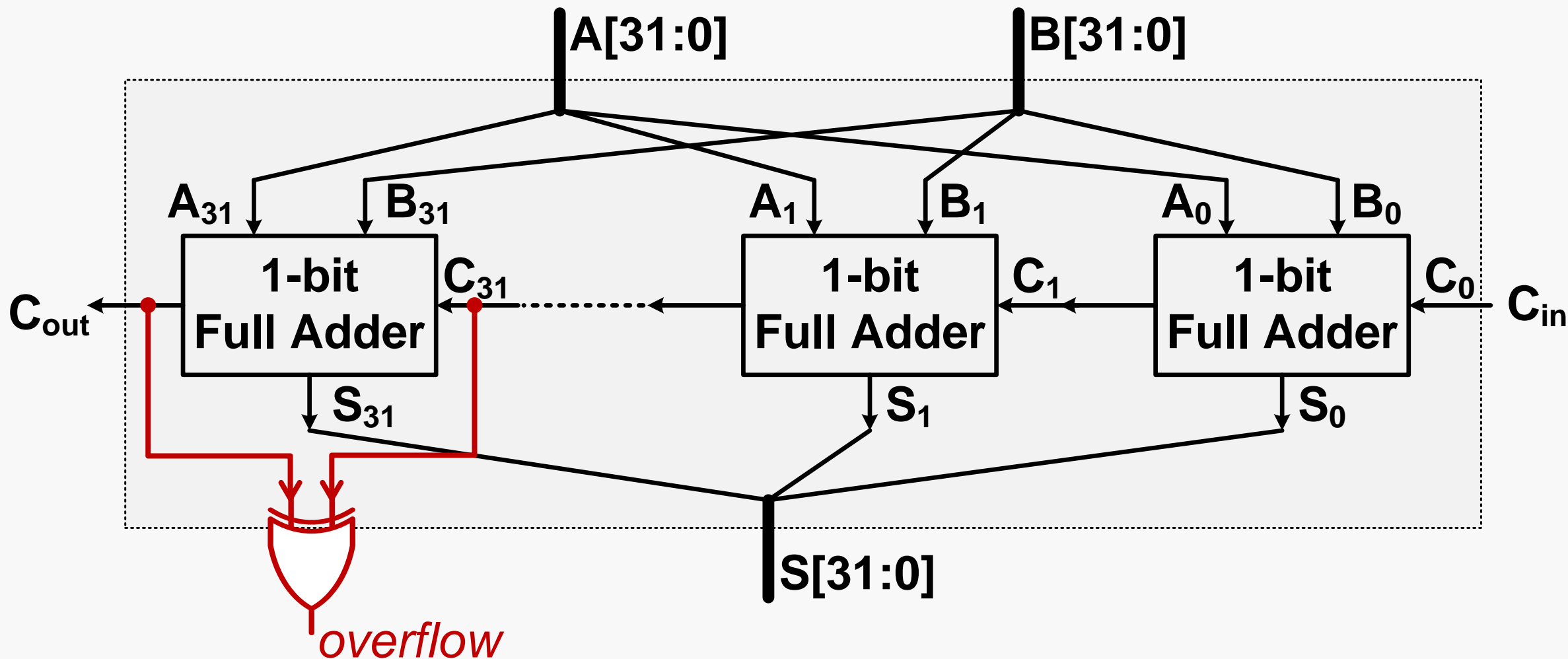
$$\begin{array}{r} 1110 \\ + 1100 \\ \hline 11010 \end{array}$$

有 “进位”，无 “溢出”



# “溢出” 的检查方法

- “最高位的进位输入” 不等于 “最高位的进位输出”



# 对“溢出”的处理方式：MIPS



## 提供两类不同的指令分别处理

(1) 将操作数看做有符号数，发生“溢出”时产生异常

- `add rd,rs,rt`                      #  $R[rd] = R[rs] + R[rt]$
- `addi rt,rs,imm`                    #  $R[rt] = R[rs] + \text{SignExtImm}$

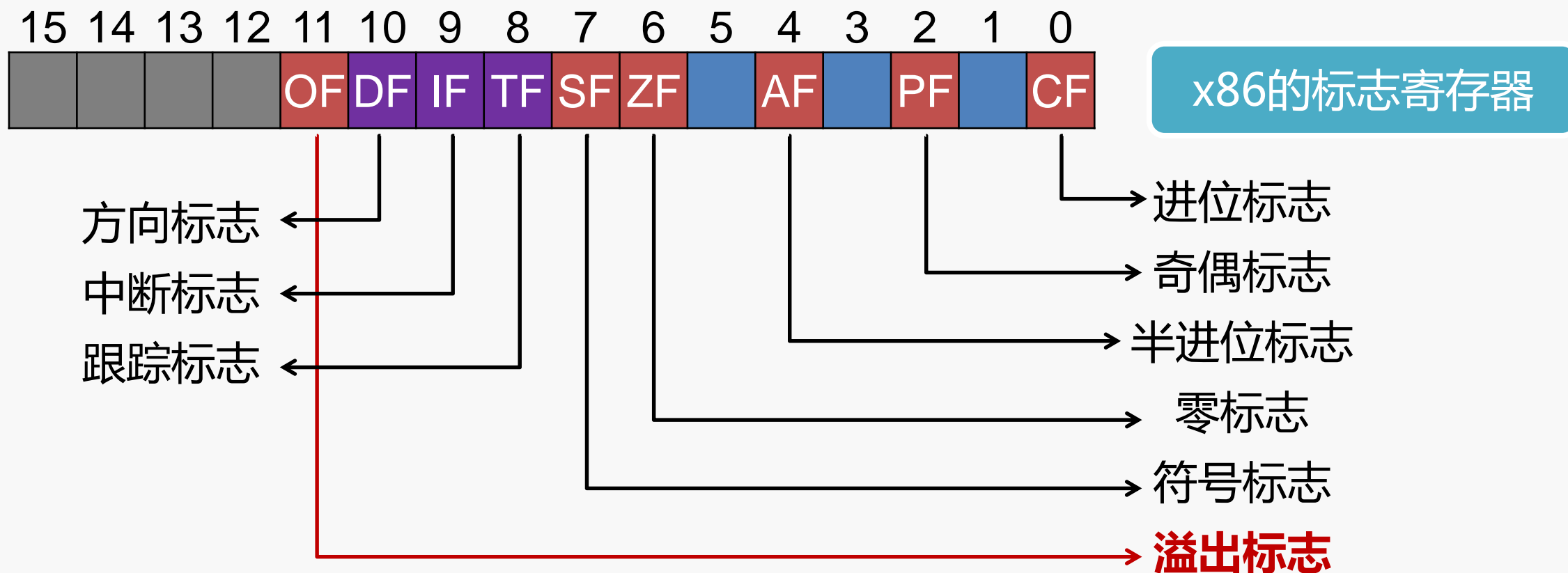
(2) 将操作数看做无符号数，不处理“溢出”

- `addu rd,rs,rt`                      #  $R[rd] = R[rs] + R[rt]$
- `addiu rt,rs,imm`                    #  $R[rt] = R[rs] + \text{SignExtImm}$

# 对“溢出”的处理方式：x86

## ❶ 溢出标志OF ( Overflow Flag )

- 如果把操作数看做有符号数，运算结果是否发生溢出
- 若发生溢出，则自动设置OF=1；否则，OF=0



# 减法运算

## ❶ 减法运算均可转换为加法运算

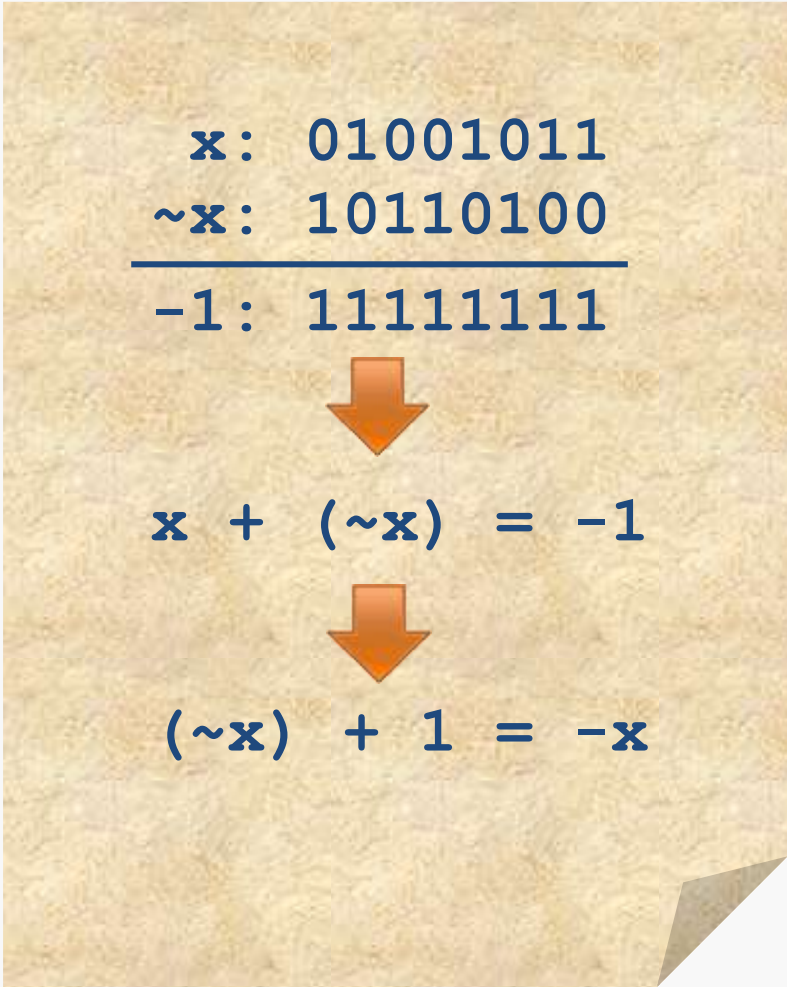
- $A - B = A + (-B)$

## ❷ 补码表示的二进制数的相反数

- 转换规则：按位取反，末位加一

## ❸ 在加法器的基础上实现减法器

- $A + (-B) = A + (\sim B + 1)$



The diagram illustrates the process of finding the negative of a binary number. It starts with a binary number  $x$  (01001011) and its bit-wise complement  $\sim x$  (10110100). A horizontal line separates these from the next step, which shows  $\sim x$  plus a carry-in of 1 (represented as -1: 11111111). An orange arrow points down to the next equation,  $x + (\sim x) = -1$ . Another orange arrow points down to the final result,  $(\sim x) + 1 = -x$ .

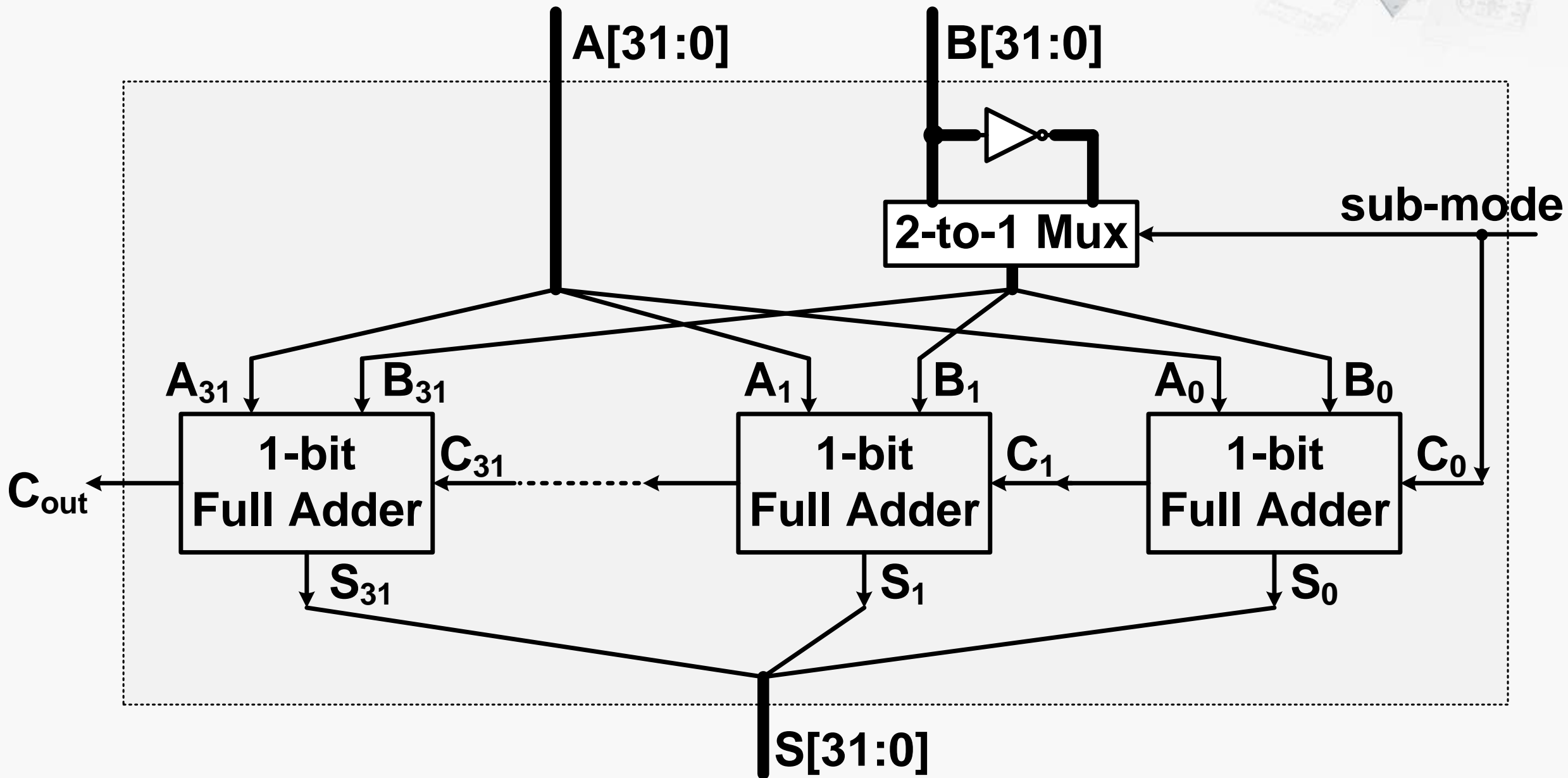
$$\begin{array}{r} x: 01001011 \\ \sim x: 10110100 \\ \hline -1: 11111111 \end{array}$$

$$x + (\sim x) = -1$$

$$(\sim x) + 1 = -x$$

# 减法运算的实现示例

$$A - B = A + (\sim B + 1)$$



## 本节小结



# 加法和减法的实现

北京大学·慕课  
计算机组成  
制作人：陆俊林

