



南京大學  
NANJING UNIVERSITY



# 整数加减运算

南京大学

计算机科学与技术系

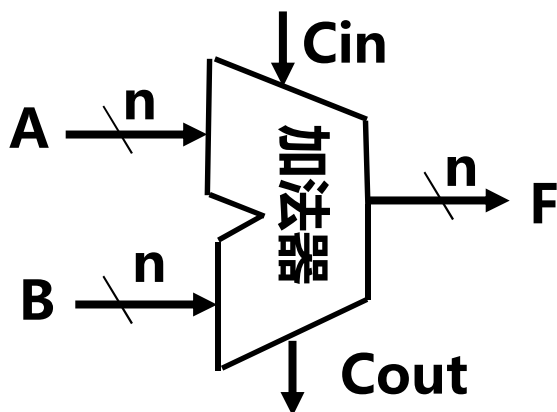
袁春风

email: [cfyuan@nju.edu.cn](mailto:cfyuan@nju.edu.cn)

2015.6

# 整数加、减运算

- C语言程序中的整数有
  - 带符号整数，如char、short、int、long型等
  - 无符号整数，如unsigned char、unsigned short、unsigned等
- 指针、地址等通常被说明为无符号整数，因而在进行指针或地址运算时，需要进行无符号整数的加、减运算
- 无符号整数和带符号整数的加、减运算电路完全一样，这个运算电路称为整数加减运算部件，基于带标志加法器实现
- 计算机中的加法器，因为只有n位，所以是一种模 $2^n$ 运算系统！



例：n=4，A=1001，B=1100

则：F=0101，Cout=1

# 回顾：整数加减运算部件

- 补码加减运算公式

$$[A+B]_{\text{补}} = [A]_{\text{补}} + [B]_{\text{补}} \pmod{2^n}$$

$$[A-B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2^n}$$

- 补码加减运算要点和运算部件

- 加、减法运算统一采用加法来处理
- 符号位(最高有效位MSB)和数值位一起参与运算
- 直接用Adder实现两个数的加运算（模运算系统）

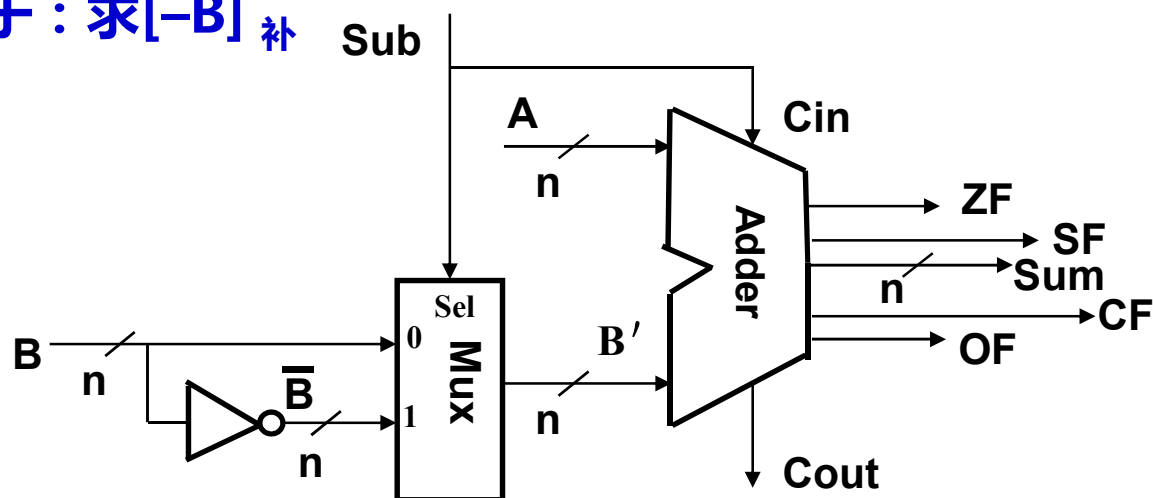
**问题：模是多少？** 运算结果高位丢弃，保留低 $n$ 位，相当于取模 $2^n$

- 实现减法的主要工作在于：求 $[-B]_{\text{补}}$

**问题：如何求 $[-B]_{\text{补}}$ ？**

$$[-B]_{\text{补}} = \overline{B} + 1$$

当Sub为1时，做减法  
当Sub为0时，做加法



# 所有运算电路的核心

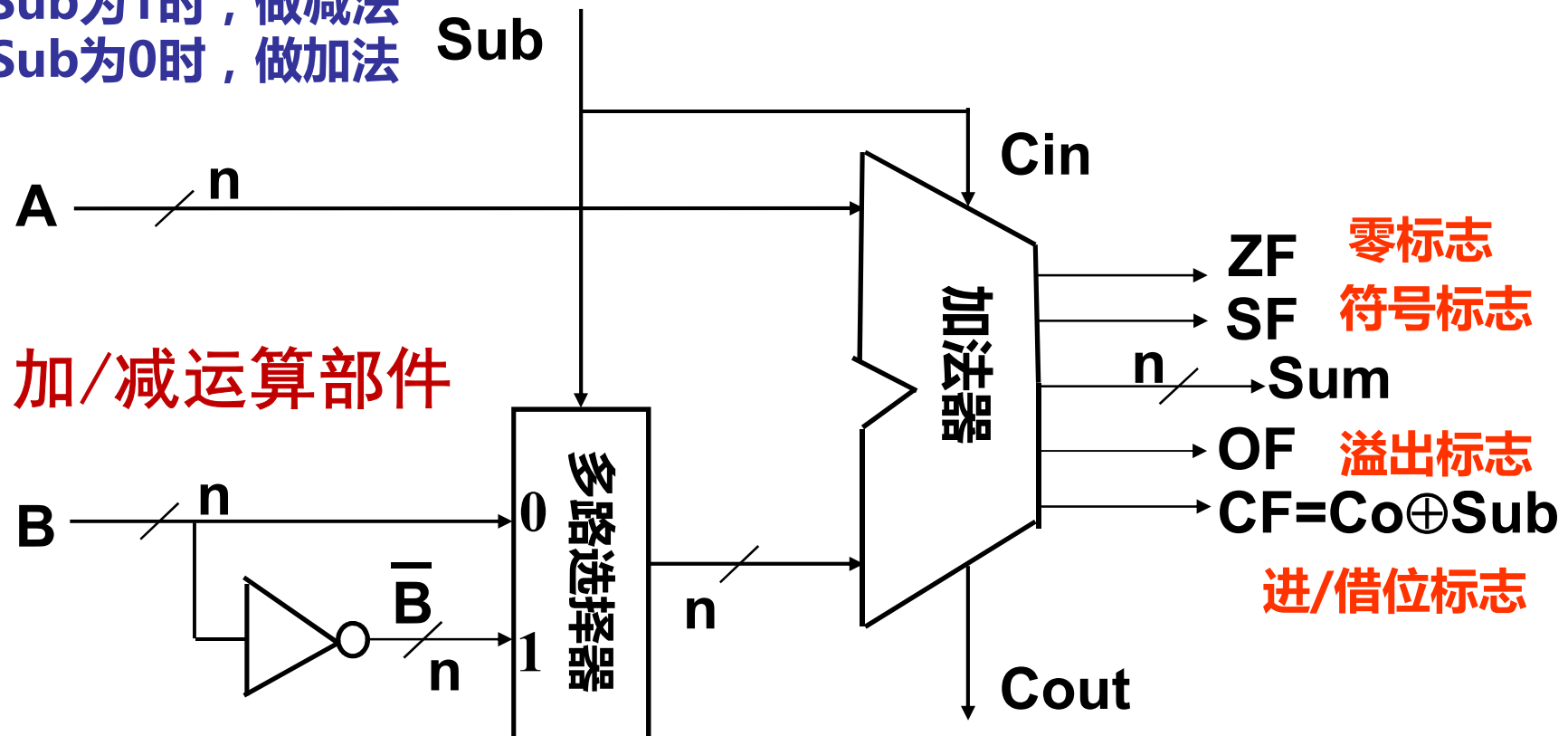


**重要认识1：**计算机中所有运算都基于加法器实现！

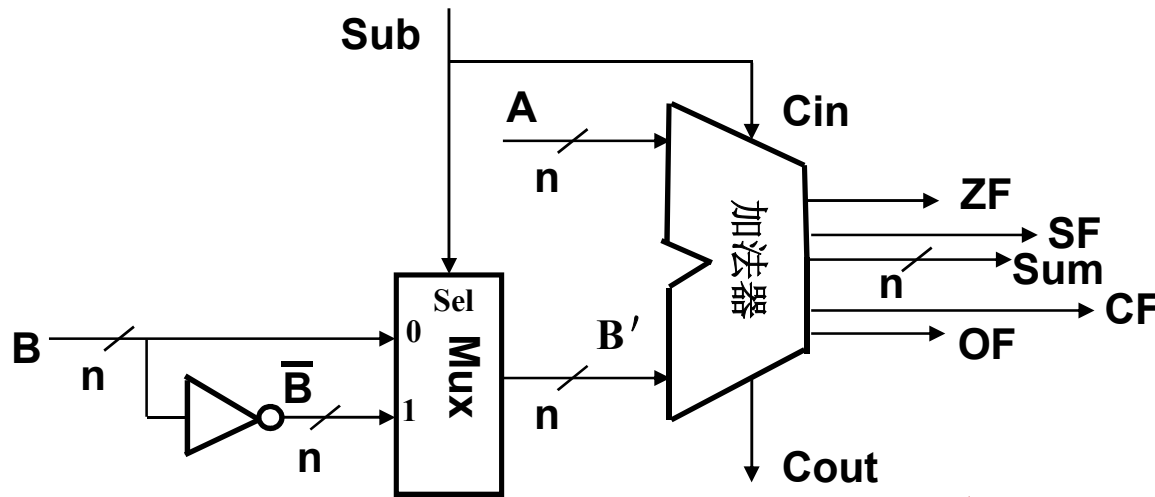
**重要认识2：**加法器不知道所运算的是带符号数还是无符号数。

**重要认识3：**加法器不判定对错，总是取低n位作为结果，并生成标志信息。

当Sub为1时，做减法  
当Sub为0时，做加法



# 条件标志位（条件码CC）



整数加/减运算部件

问题：为什么要生成并保存条件标志？

为了在分支指令（条件转移指令）中被用作是否转移执行的条件！

问题：OF = ? ZF = ?  
SF = ? CF = ?

```
if (i > j) {  
    ...  
}
```

还记得如何得到各个标志位吗？

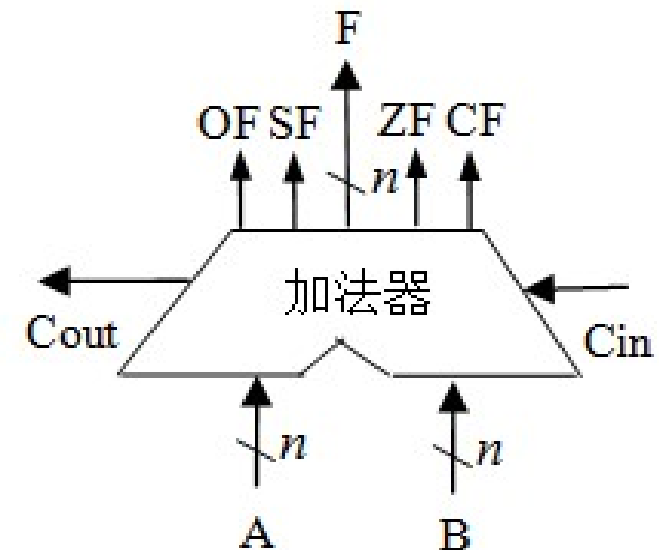
**OF**：若A与B' 同号但与Sum不同号，则1；否则0。 **SF**：sum符号

**ZF**：如Sum为0，则1，否则0。 **CF**： $Cout \oplus sub$

- 零标志**ZF**、溢出标志**OF**、进/借位标志**CF**、符号标志**SF**称为条件标志。
- **条件标志（Flag）**在运算电路中产生，被记录到专门的寄存器中
- 存放标志的寄存器通常称为**程序/状态字寄存器**或**标志寄存器**。每个标志对应标志寄存器中的一个标志位。如，IA-32中的**EFLAGS寄存器**

# n位带标志加法器

- n位加法器无法用于两个n位带符号整数（补码）相加，无法判断是否溢出
- 程序中经常需要比较大小，通过（在加法器中）做减法得到的标志信息来判断



带标志加法器符号

溢出标志OF :

$$OF = C_n \oplus C_{n-1}$$

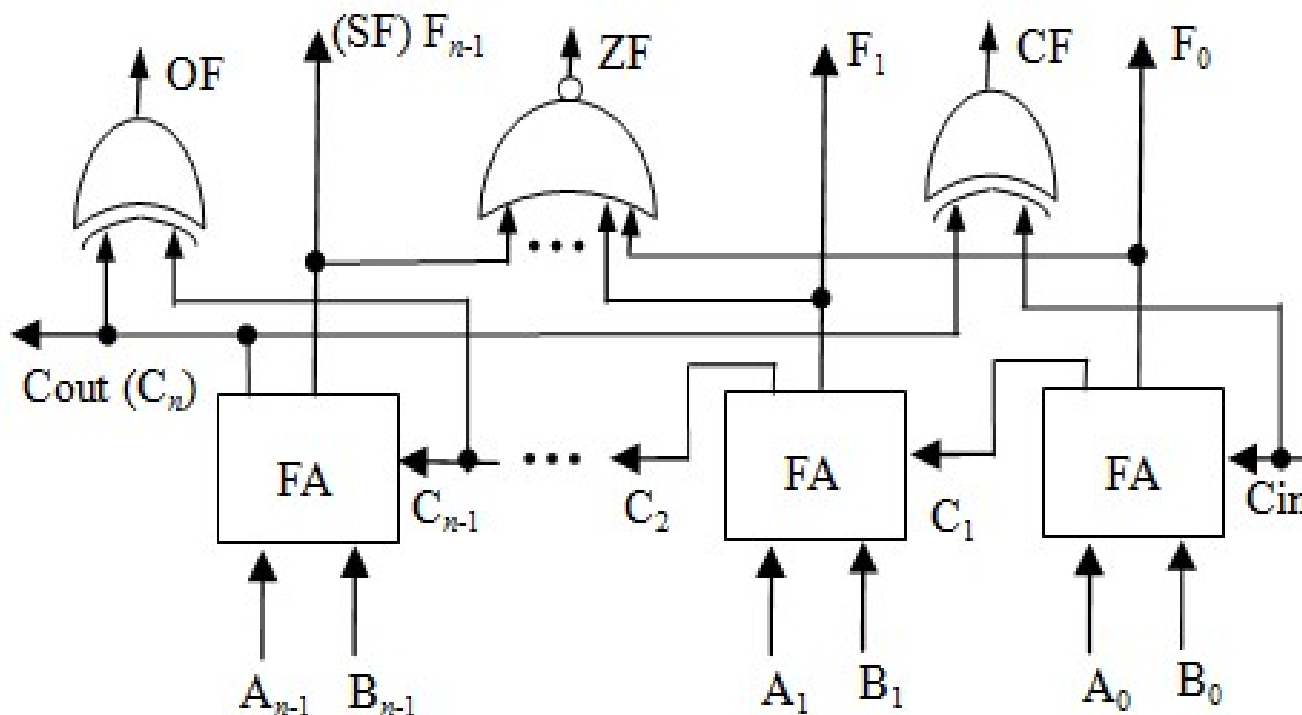
符号标志SF :

$$SF = F_{n-1}$$

零标志ZF=1当且仅当F=0 ;

进位/借位标志CF :

$$CF = Cout \oplus Cin$$



带标志加法器的逻辑电路

# 整数加法举例

做加法时，主要判断是否溢出

无符号加溢出条件：CF=1

带符号加溢出条件：OF=1

若 $n=8$ ，计算 $107+46=?$

$$107_{10} = 0110\ 1011_2$$

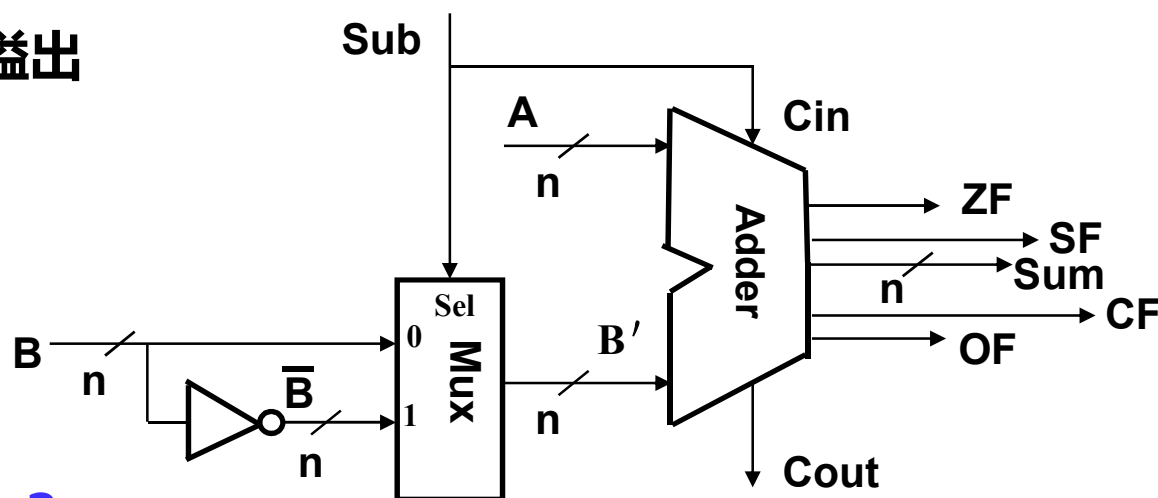
$$46_{10} = 0010\ 1110_2$$

$$\boxed{0}1001\ 1001$$

进位是真正的符号：+153

无符号：sum=153，因为CF=0，故未发生溢出，结果正确！

带符号：sum= -103，因为OF=1，故发生溢出，结果错误！



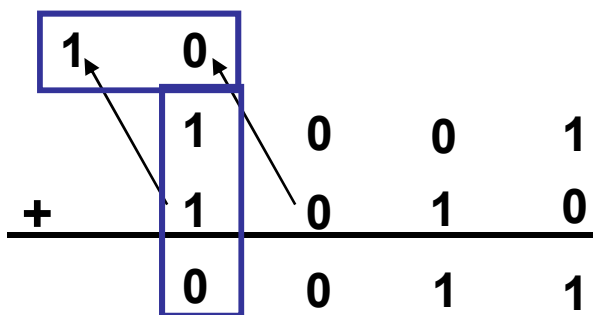
整数加/减运算部件

两个正数相加，结果为负数，  
故溢出！即OF=1

溢出标志OF=1、零标志ZF=0、  
符号标志SF=1、进位标志CF=0

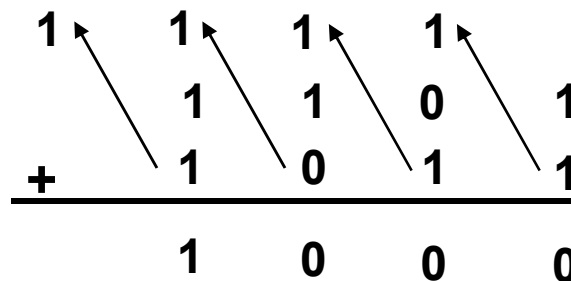
# 整数减法举例

$$\begin{aligned} -7 - 6 &= -7 + (-6) = +3 \text{ x} \\ 9 - 6 &= 3 \checkmark \end{aligned}$$



OF=1、ZF=0  
SF=0、借位CF=0

$$\begin{aligned} -3 - 5 &= -3 + (-5) = -8 \checkmark \\ 13 - 5 &= 8 \checkmark \end{aligned}$$



OF=0、ZF=0、  
SF=1、借位CF=0

带符号 (1) 最高位和次高位的进位不同  
溢出：(2) 和的符号位和加数的符号位不同

无符号减溢出：差为负数，即借位CF=1

做减法以比较大小，规则：  
Unsigned: CF=0时，大于  
Signed : OF=SF时，大于

验证：9>6，故CF=0；13>5，故CF=0

验证：-7<6，故OF≠SF  
-3<5，故OF≠SF



# 整数减法举例

unsigned int x=134;

unsigned int y=246;

int m=x;

int n=y;

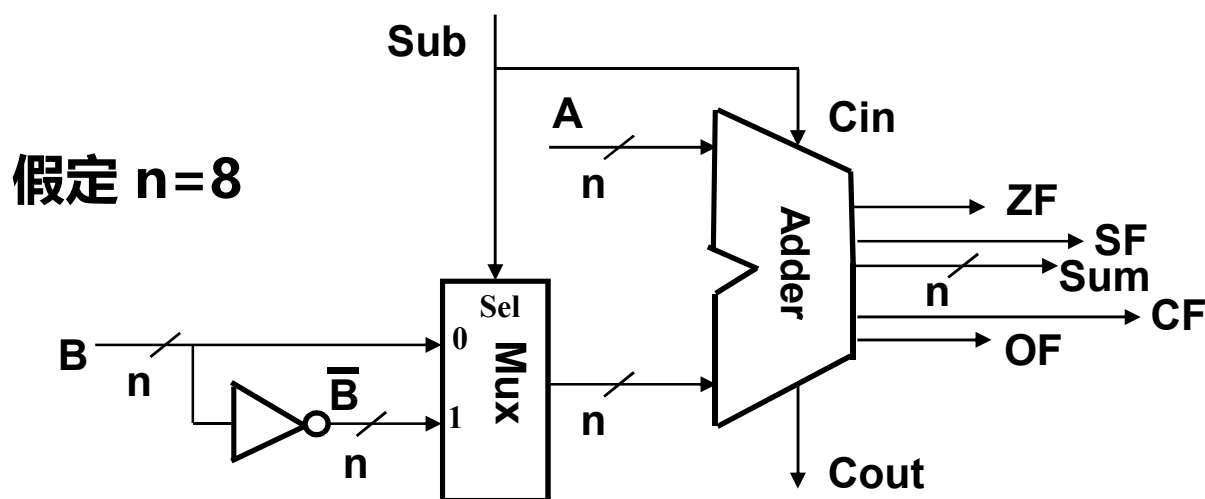
unsigned int **z1=x-y;**

unsigned int **z2=x+y;**

int **k1=m-n;**

int **k2=m+n;**

无符号和带符号加减运算都用该部件执行



x和m的机器数一样：1000 0110，y和n的机器数一样：1111 0110

z1和k1的机器数一样：1001 0000，**CF=1**，**OF=0**，**SF=1**

z1的值为144 ( =134-246+256， $x-y<0$  )，**k1**的值为-112。

无符号减公式：

$$\text{result} = \begin{cases} x-y & (x-y > 0) \\ x-y+2^n & (x-y < 0) \end{cases}$$

带符号减公式：

$$\text{result} = \begin{cases} x-y-2^n & (2^{n-1} \leq x-y) & \text{正溢出} \\ x-y & (-2^{n-1} \leq x-y < 2^{n-1}) & \text{正常} \\ x-y+2^n & (x-y < -2^{n-1}) & \text{负溢出} \end{cases}$$

# 整数加法举例

unsigned int x=134;

unsigned int y=246;

int m=x;

int n=y;

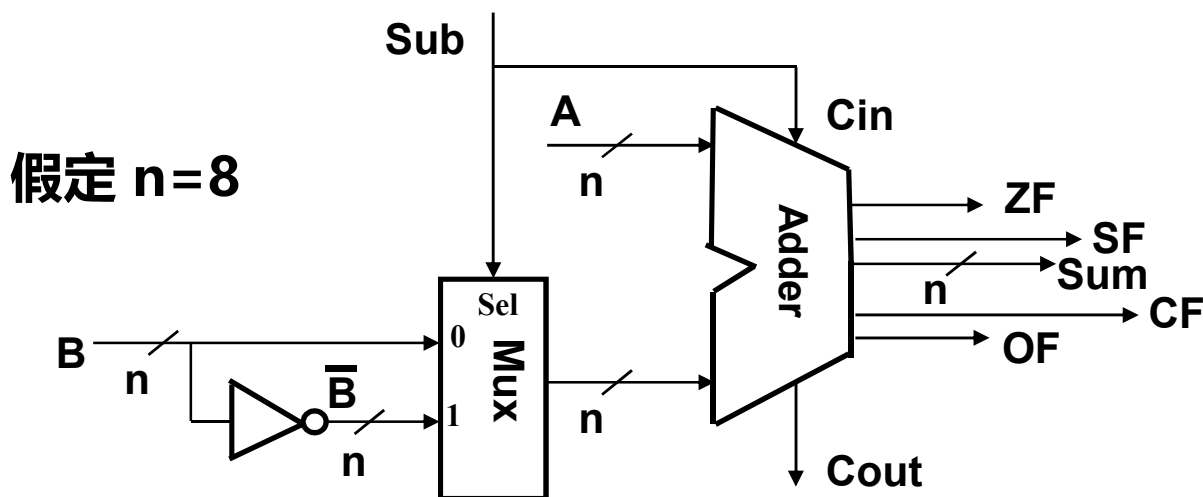
unsigned int z1=x-y;

unsigned int z2=x+y;

int k1=m-n;

int k2=m+n;

无符号和带符号加减运算都用该部件执行



x和m的机器数一样：1000 0110，y和n的机器数一样：1111 0110

z2和k2的机器数一样：0111 1100，CF=1，OF=1，SF=0

z2的值为124 ( =134+246-256，x+y>256 )

k2的值为124 ( =134+246-256，m+n>128，即正溢出 )

带符号加公式：

无符号加公式：

$$\text{result} = \begin{cases} x+y & (x+y < 2^n) \\ x+y-2^n & (2^n \leq x+y < 2^{n+1}) \end{cases}$$

$$\text{result} = \begin{cases} x+y-2^n & (2^{n-1} \leq x+y) & \text{正溢出} \\ x+y & (-2^{n-1} \leq x+y < 2^{n-1}) & \text{正常} \\ x+y+2^n & (x+y < -2^{n-1}) & \text{负溢出} \end{cases}$$

# 无符号整数加法溢出判断程序

如何用程序判断一个**无符号数相加**没有发生溢出



$$\text{result} = \begin{cases} x+y & (x+y < 2^n) \\ x+y-2^n & (2^n \leq x+y < 2^{n+1}) \end{cases}$$

发生溢出时，一定满足  $\text{result} < x$  and  $\text{result} < y$   
否则，若  $x+y-2^n \geq x$ ，则  $y \geq 2^n$ ，这是不可能的！

**/\* Determine whether arguments can be added without overflow \*/**

```
int uadd_ok(unsigned x, unsigned y)  
{  
    unsigned sum = x+y;  
    return sum >= x;  
}
```

# 带符号整数加法溢出判断程序

如何用程序判断一个带符号整数相加没有发生溢出



$$\text{result} = \begin{cases} x+y-2^n & (2^{n-1} \leq x+y) & \text{正溢出} \\ x+y & (-2^{n-1} \leq x+y < 2^{n-1}) & \text{正常} \\ x+y+2^n & (x+y < -2^{n-1}) & \text{负溢出} \end{cases}$$

**/\* Determine whether arguments can be added without overflow \*/**

```
int tadd_ok(int x, int y) {  
    int sum = x+y;  
    int neg_over = x < 0 && y < 0 && sum >= 0;  
    int pos_over = x >= 0 && y >= 0 && sum < 0;  
    return !neg_over && !pos_over;  
}
```

# 带符号整数减法溢出判断程序

以下程序检查带符号整数相减是否溢出有没有问题？



$$\text{result} = \begin{cases} x+y-2^n & (2^{n-1} \leq x+y) & \text{正溢出} \\ x+y & (-2^{n-1} \leq x+y < 2^{n-1}) & \text{正常} \\ x+y+2^n & (x+y < -2^{n-1}) & \text{负溢出} \end{cases} \quad \text{带符号整数加}$$

$$\text{result} = \begin{cases} x-y-2^n & (2^{n-1} \leq x-y) & \text{正溢出} \\ x-y & (-2^{n-1} \leq x-y < 2^{n-1}) & \text{正常} \\ x-y+2^n & (x-y < -2^{n-1}) & \text{负溢出} \end{cases} \quad \text{带符号整数减}$$

**/\* Determine whether arguments can be subtracted without overflow \*/**

**/\* WARNING: This code is buggy. \*/**

```
int tsub_ok(int x, int y) {  
    return tadd_ok(x, -y);  
}
```

当  $x=0$  ,  $y=0x80000000$  时 , 该函数判断错误

带符号减的溢出判断函数如何实现呢？

无符号减的溢出判断函数又如何实现呢？