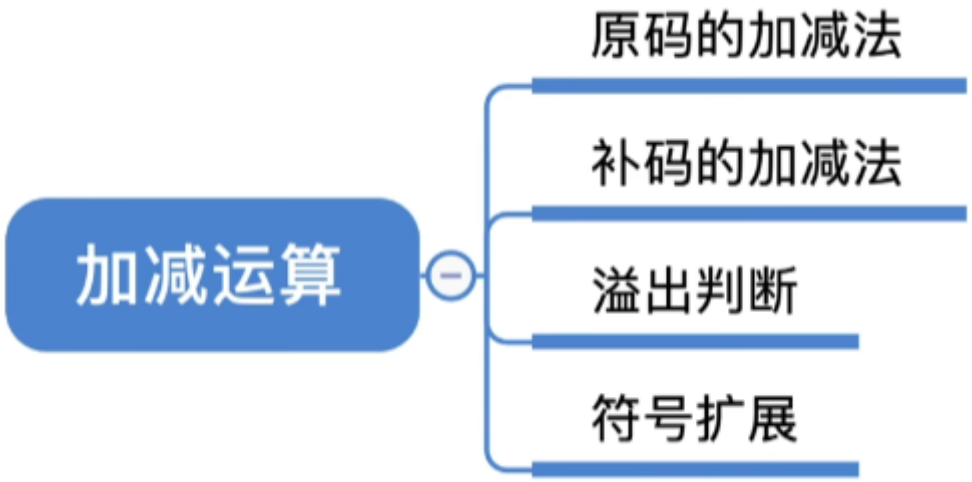
加减运算和溢出判断



原码的加减运算

原码表示的有符号数

```
  14        00001110
 -14   +    10001110
   0        10011100
```

☹

加法器直接对原码进行加法运算，可能出错

```
   00001110
+  10001110
```

用减法器实现

```
   00001110
-  00001110
   00000000
```

☺

原码的加法运算：

正+正 →绝对值做加法，结果为正
负+负 →绝对值做加法，结果为负
} 可能会溢出
正+负 →绝对值大的减绝对值小的，符号同绝对值大的数
负+正 →绝对值大的减绝对值小的，符号同绝对值大的数

原码的减法运算，“减数”符号取反，转变为加法：

正-负→正+正
负-正→负+负
正-正→正+负
负+正→负-负

补码的加减运算

设机器字长为8位（含1位符号位），$A=15$，$B=-24$，求$[A+B]_{补}$和$[A-B]_{补}$

	原码	补码
$A=+1111$	→ 0,0001111	→ 0,0001111
$B=-11000$	→ 1,0011000	→ 1,1101000

负数补→原：①数值位取反+1；②负数补码中，最右边的1及其右边同原码。最右边的1的左边同反码

$[A+B]_{补}=[A]_{补}+[B]_{补}=0,0001111+1,1101000=1,1110111$

原码：1,0001001　真值-9

$[A-B]_{补}=[A]_{补}+[-B]_{补}=0,0001111+0,0011000=0,0100111$　真值+39

$[-B]_{补}$：　$[B]_{补}$连同符号位一起取反加1

$C=124$，求$[A+C]_{补}$和$[B-C]_{补}$

溢出

$[A+C]_{补}=0,0001111+0,1111100=1,0001011$　真值-117

$[B-C]_{补}=1,1101000+1,0000100=0,1101100$　真值+108

对于补码来说，无论加法还是减法，最后都会转变成加法，由加法器实现运算，符号位也参与运算

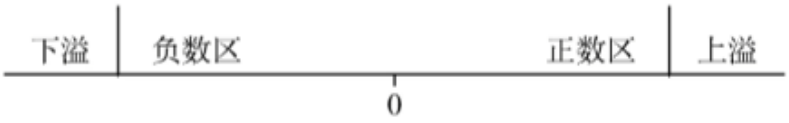
溢出判断

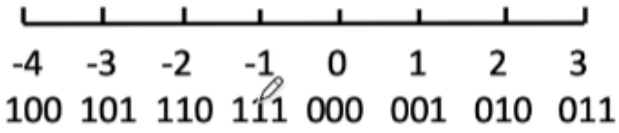
设机器字长为8位（含1位符号位），$A=15$，$B=-24$，求$[A+B]_{补}$和$[A-B]_{补}$

$C=124$，求$[A+C]_{补}$和$[B-C]_{补}$

$[A+C]_{补}=0,0001111+0,1111100=1,0001011$　真值-117

$[B-C]_{补}=1,1101000+1,0000100=0,1101100$　真值+108





只有“正数+正数”才会上溢 —— 正+正=负

只有“负数+负数”才会下溢 —— 负+负=正

逻辑表达式

与：如ABC，表示A与B与C
仅当A、B、C均为1时，ABC为1
A、B、C中有一个或多个为0，则ABC为0

或：如A+B+C，表示A或B或C
仅当A、B、C均为0时，A+B+C为0
A、B、C中有一个或多个为1，则A+B+C为1

非：如 $\overline{A}$，表示A非
若A为1，则 $\overline{A}$ 为0
若A为0，则 $\overline{A}$ 为1

方法一：采用一位符号位
设A的符号为A_S，B的符号为B_S，运算结果的符号为S_S，则溢出逻辑表达式为

$$V=A_S B_S \overline{S_S}+\overline{A_S}\,\overline{B_S} S_S$$

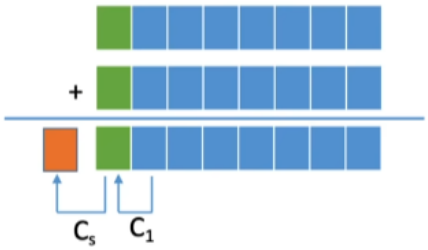
若V=0，表示无溢出；
若V=1，表示有溢出。

A_S为1且B_S为1且S_S为0　　A_S为0且B_S为0且S_S为1

设机器字长为8位（含1位符号位），$A=15$，$B=-24$，求$[A+B]_{补}$和$[A-B]_{补}$

$C=124$，求$[A+C]_{补}$和$[B-C]_{补}$

$[A+C]_{补}=0,0001111+0,1111100=1,0001011$　　真值-117
$[B-C]_{补}=1,1101000+1,0000100=0,1101100$　　真值+108



方法二：采用一位符号位，根据数据位进位情况判断溢出

	符号位的进位C_S	最高数值位的进位C_1
上溢	0	1
下溢	1	0

即：C_S与C_1不同时有溢出
处理“不同”的逻辑符号：异或⊕
溢出逻辑判断表达式为$V=C_S \oplus C_1$
若V=0，表示无溢出；V=1，表示有溢出。

异或逻辑：不同为1，相同为0
0⊕0 = 0
0⊕1 = 1
1⊕0 = 1
1⊕1 = 0

方法三：采用双符号位
正数符号为00，负数符号为11

$[A+C]_{补}=00,0001111+00,1111100=01,0001011$　　上溢
$[B-C]_{补}=11,1101000+11,0000100=10,1101100$　　下溢

记两个符号位为$S_{S1}S_{S2}$，则$V=S_{S1}\oplus S_{S2}$
若V=0，表示无溢出；若V=1，表示有溢出。

$[A+B]_{补}=00,0001111+11,1101000=11,1110111$
$[A-B]_{补}=00,0001111+00,0011000=00,0100111$

实际存储时只存储1个符号位，运算时会复制一个符号位

双符号位补码又称：模4补码
单符号位补码又称：模2补码

符号扩展

int->long，短数据->长数据。多出来的那些位应该怎么填补？

8位->16位 正整数（原、反、补码的表示都一样）

int→long，短数据→长数据。多出来的那些位应该怎么填补？

Eg：8位→16位

正整数（原、反、补码的表示都一样）

0,1011010 → 0,00000000 1011010

正小数（原、反、补码的表示都一样）

0.1011010 → 0.1011010 00000000

	负整数：		负小数：	
原码：	1,1011010	→ 1,00000000 1011010	1.1011010	→ 1.1011010 00000000
反码：	1,0100101	→ 1,11111111 0100101	1.0100101	→ 1.0100101 11111111
补码：	1,0100110	→ 1,11111111 0100110	1.0100110	→ 1.0100110 00000000

定点整数的符号扩展：
在原符号位和数值位中间添加新位，正数都添0；负数原码添0，负数反、补码添1

定点小数的符号扩展：
在原符号位和数值位后面添加新位，正数都添0；负数原、补码添0，负数反码添1