机器码、真值、原码、反码以及补码

阐述机器底层为什么使用补码进行加减运算

1、机器码

正数最高位用`0`表示，负数最高位用`1`表示

2、真值

`0000 0001`的真值是`+1`

`1000 0001`的真值是`-1`，而不是`129`

3、原码

原码是符号位加上真值的的绝对值，即第一位表示符号，其余位表示值（原码也是机器码本身）

`[+1]`的原码是`[0000 0001]`

`[-1]`的原码是`[1000 0001]`

> 假设机器内部使用原码参与加减运算:
>
> 期望结果：`[+1] + [-1] = [0]`
>
> 实际情况：`[0000 0001]原 + [1000 0001]原 = [1000 0010]原 = [-2]`
>
> 使用原码后机器的取值范围：`[1111 1111, 0111 1111]原 = [-127, 127]`

4、反码

正数的反码是其本身（原码）；负数的反码符号位不变，其余各位取反

`[0000 0001]`的反码是`[0000 0001]`

`[1000 0001]`的反码是`[1111 1110]`

> 假设机器内部使用反码参与加减运算:
>
> 期望结果：`[+1] + [-1] = [0]`
>
> 实际情况：`[0000 0001]反 + [1111 1110]反 = [1111 1111]反 = [1000 0000]原 = [-0]`
>
> 会出现`[0000 0000]原`与`[1000 0000]原`两个编码表示`0`

5、补码

正数的补码是其本身；负数的补码符号位不变，其余各位取反，再加`1`（负数的补码+1）

`[0000 0001]`的补码是`[0000 0001]`

`[1000 0001]`的补码是`[1111 1111]`

> 假设机器内部使用反码参与加减运算:
>
> 期望结果：`[+1] + [-1] = [0]`
>
> 实际情况：`[0000 0001]补 + [1111 1111]补 = [0000 0000]补 = [0000 0000]反 = [0000 0000]原 = [0]`

> `[-1] + [-127] = [-128] = [1000 0001]原 + [1111 1111]原 = [1111 1111]补 + [1000 0001]补 = [1000 000]补`
>
> 补码消除了存在两个编码表示为`0`的情况，且使用补码后机器的取值范围：`[-128, 127]`
>
> 注意：`[1000 0000]补`推算出`[0000 0000]原`是不正确的，`-128`并没有反码与原码表示

6、右移位`>>`、`>>>`及左移位`<<

1、`>>`移位：带符号右移位（每移右移一位最高位始终用`1`补充）

`>>`右移公式（`x`表示当前数，`n`表示左移位数）

$$Y = \lfloor X * 2^{-n} \rfloor$$

`-5 >> 1 = -3`

```
[-5] = [1000 0101]原 = [1111 1010]反 = [1111 1011]补
[1111 1011]补 >> 1 = [1111 1101]补
[1111 1101]补 + [1111 1111]补 = [1111 1100]反 = [1000 0011]原 = [-3]
```

`-5 >> 2 = -2`

```
[-5] = [1000 0101]原 = [1111 1010]反 = [1111 1011]补
[1111 1011]补 >> 2 = [1111 1110]补
[1111 1110]补 + [1111 1111]补 = [1111 1101]反 = [1000 0010]原 = [-2]
```

`-5 >> 3 = -1`

```
[-5] = [1000 0101]原 = [1111 1010]反 = [1111 1011]补
[1111 1011]补 >> 3 = [1111 1111]补
[1111 1111]补 + [1111 1111]补 = [1111 1110]反 = [1000 0001]原 = [-1]
```

2、`>>>`移位：不带符号右移

`-5 >>> 1 = 125`

```
[-5] = [1000 0101]原 = [1111 1010]反 = [1111 1011]补
[1111 1011]补 >>> 1 = [0111 1101]补
[0111 1101]补 = [0111 1101]反 = [0111 1101]原 = [125]
```

`-5 >>> 2 = 62`

```
[-5] = [1000 0101]原 = [1111 1010]反 = [1111 1011]补
[1111 1011]补 >>> 2 = [0011 1110]补
[0011 1110]补 = [0011 1110]反 = [0011 1110]原 = [62]
```

`-5 >>> 3 = 31`

```
[-5] = [1000 0101]原 = [1111 1010]反 = [1111 1011]补
[1111 1011]补 >>> 3 = [0001 1111]补
[0001 1111]补 = [0001 1111]反 = [0001 1111]原 = [31]
```

3、`<<`左移公式（`X`表示当前数，`n`表示左移位数）

$$Y = X * 2^n$$