# Binär subtraktion för heltal utan tecken, D = X – Y

**Exempel med två 8-bitars heltal X och Y:**

$X = 10101101_2 = 173_{10}$

$Y = 01100110_2 = 102_{10}$

```
 8 7 6 5 4 3 2 1 0     bitposition
 0 0 2 0 0 0 2 2 0     lån (borrow)
   1 0 1 0 1 1 0 1     X
  -0 1 1 0 0 1 1 0    -Y
   0 1 0 0 0 1 1 1 = D = 71(10)
```

"***Overflow***" vid subtraktion av heltal utan tecken inträffar om det krävs ett lån vid subtraktionen av siffrorna längst till vänster. Det innebär att resultatet skulle ha blivit negativt, dvs. utanför talområdet. Negativa tal finns ju inte för tal utan tecken.

Addition och subtraktion utförs olika och kräver därmed olika grindnät ( = kretsar).

Vi använder istället ***2-komplementaritmetik***, på samma sätt som vi tidigare använde ***10-komplementaritmetik*** och kan då utföra subtraktionen som en addition.

Byt ut X – Y mot $2^n + X – Y$, där n är antalet bitar vi arbetar med.

**Exempel med n = 8:**

$2^8 + X – Y = 256 + X – Y = 255 + 1 + X – Y = X + (11111111_2 – Y) + 1 = X + Y_{1k} + 1$

$11111111_2 – Y = Y_{1k}$ som kallas (siffervis) 1-komplement av Y.
Man bildar $Y_{1k}$ genom att invertera alla bitar i Y.

Med samma värden på X och Y som ovan får vi:

```
 8 7 6 5 4 3 2 1 0       bitposition
 1 0 1 1 1 0 0 1 1   1   minnessiffra (carry)
   1 0 1 0 1 1 0 1   X
  +1 0 0 1 1 0 0 1  +Y1k
 1 0 1 0 0 0 1 1 1 = D = 71(10)
```

Lägg märke till att vi har satt minnessiffran längst till höger till 1 ($c_0 = 1$) och att minnessiffran ut $c_8 = 1$. $c_8$ har ju vikten $2^8 = 256$, som vi lade till från början.
Vi får den korrekta skillnaden genom att stryka $c_8$.

*Overflow* vid subtraktion av tal utan tecken med 2-komplementaritmetik upptäcks genom att minnessiffran ($c_8$ i exemplet ovan) vid additionen av siffrorna längst till vänster blir 0. Eftersom $c_8 = 1$ i exemplet blir det inte *overflow* i detta fall.