

Decimala talsystemet

- har basen 10
- använder tecknen 0-9
- positionen längst till höger, (entalspositionen) $10^0=1$ och kan vara 0-9 vilket ger värdet 0-9
- positionen längst näst längst till höger, (tiotalpositionen) $10^1=10$ och kan vara 0-9 vilket ger värdet 0, 10, 20, 30, 40, 50, 60, 70, 80 eller 90
- tredje positionen från höger, (hundratalpositionen) $10^2=100$ och kan vara 0-9 vilket ger värdet 0, 100, 200, 300, 400, 500, 600, 700, 800 eller 900
- 0-9000, 0-90000, 0-900000, osv
- **exempel:** talet 257 kan skrivas så här: $2 \times 100 + 5 \times 10 + 7 \times 1$

Binära talsystemet

- har basen 2
- använder tecknen 0-1
- positionen längst till höger $2^0=1$ och kan vara 0 eller 1 vilket ger värdet 0 eller 1
- nästa position $2^1=2$ och kan vara 0 eller 1 vilket ger värdet 0 eller 2
- nästa position $2^2=4$ och kan vara 0 eller 1 vilket ger värdet 0 eller 4
- nästa position $2^3=8$ och kan vara 0 eller 1 vilket ger värdet 0 eller 8
- 0/16, 0/32, 0/64, 0/128, osv
- **exempel:** talet b'100000001' (dec 257) kan skrivas så här:
 $1 \times 256 + 0 \times 128 + 0 \times 64 + 0 \times 32 + 0 \times 16 + 0 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1$

Hexadecimala talsystemet

- har basen 16
- använder tecknen 0-F
- positionen längst till höger $16^0=1$ och kan vara 0 eller 1 vilket ger värdet 0 eller 1
- nästa position $16^1=16$ och kan vara 0 eller 1 vilket ger värdet 0 eller 16
- nästa position $16^2=256$ och kan vara 0 eller 1 vilket ger värdet 0 eller 256
- nästa position $16^3=4096$ och kan vara 0 eller 1 vilket ger värdet 0 eller 4096
- 0/65536, 0/1048576, 0/16777216, 0/268435456, osv
- **exempel:** talet x'101' (dec 257) kan skrivas så här:
 $1 \times 256 + 0 \times 16 + 1 \times 1$

Användning av binära tal

- en dator kan endast använda 0 eller 1 som representerar ON eller OFF, ström eller inte ström, magnetism eller inte magnetism, osv.
- det som läses in från en skiva, knappas in från tangentbordet eller finns lagrat i minnet finns i denna binära form
- på disketten eller hårddisken är det magnetism eller inte magnetism och i minnet är en viss minnesposition ON eller OFF
- dessa binära tecken är ingenting av sig själva - det är hur de används eller tolkas som ger betydelsen
- ett A (stort) i EBCDIC-code ger detta binära tal: b'11000001' (dec 193, men datorn kan inte klara av det decimala talsystemet)
- nu skriver man nästan aldrig ut det binära talet utan använder en Hexadecimal representation istället, det blir enklare och framför allt kortare tal.

Se nästa bild --->

Användning av Hexadecimala tal 1/2

- som konstaterades på förra bilden använder man Hexadecimal representation när man ska skriva binära tal
- då delar man in det binära talet i grupper om fyra (4) bitar (bits)
- varje sådan grupp kallas half-byte och kan representeras med ett Hexadecimalt tal
- tecknet A (stort) i EBCDIC-code (11000001) som nämndes på förra bilden kan delas i två grupper om fyra bitar, b'1100' och b'0001'
- enligt det binära talsystemet får bitarna följande värde beroende på vilken position de har i gruppen, 8, 4, 2, 1 (se bild om binära talsystemet)
- varje grupp kan nu representeras av ett Hexadecimalt tal, första gruppen blir X 'C' och andra gruppen X '1'. Skriver man dem tillsammans blir det X 'C1' och det är så man t.ex anger tecknet för stora A i EBCDIC-kod

Användning av Hexadecimala tal 2/2

0 = 0000	A = 1010 (dec 10)
1 = 0001	B = 1101 (dec 11)
2 = 0010	C = 1100 (dec 12)
3 = 0011	D = 1101 (dec 13)
4 = 0100	E = 1110 (dec 14)
5 = 0101	F = 1111 (dec 15)
6 = 0110	
7 = 0111	
8 = 1000	
9 = 1001	

Exempel: X '3EFF'

- Expandera först till grupper om fyra bitar vilket ger:
 - b'0011 1110 1111 1111'
- Man behandlar alltid varje grupp om fyra bitar tillsammans
- X '3EFF' kan vara en adress i minnet eller vad som står på ett visst ställe i minnet, det kan också vara värdet i ett register som t.ex. talar om vilka egenskaper en skrivare har, osv.
- dessa värden läses av ett program, men i datorn blir det alltid 0:or och 1:or igen