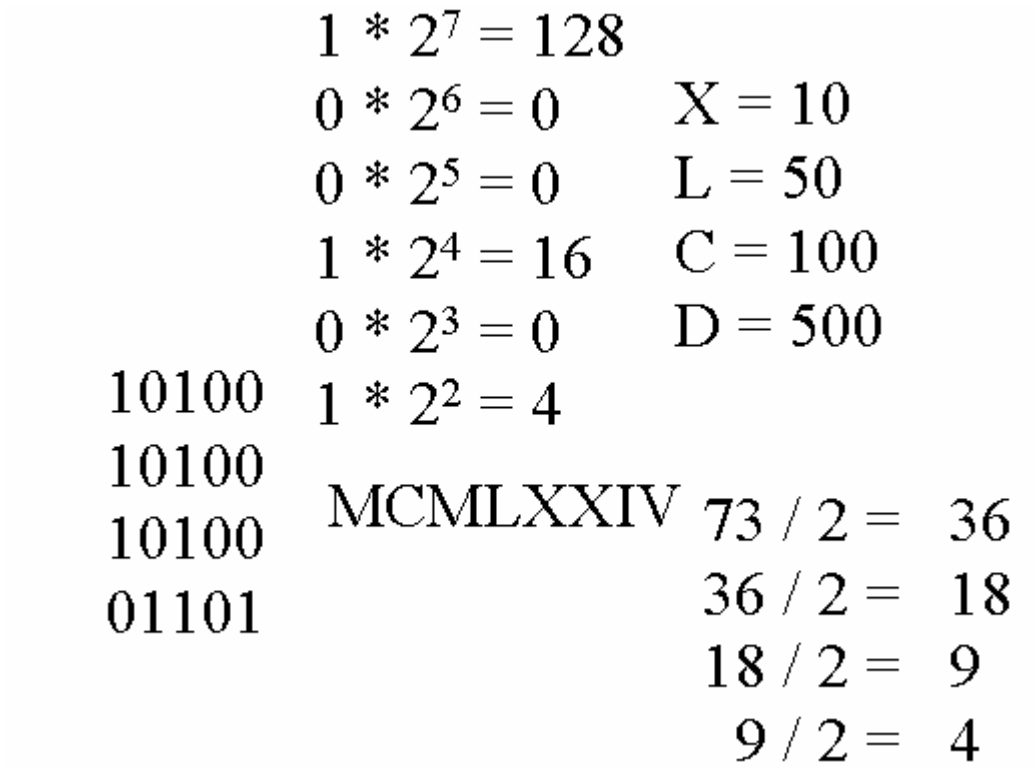
Talsystemer



Indhold

Indledning

Der findes to typer talsystemer, vores normale ti-talsystem er et positions-system, mens det romerske talsystem er et additivt talsystem.

Det romerske talsystem.

I det romerske talsystem betyder de enkelte cifre en bestemt værdi, mens det ikke betyder noget hvor de står.
De normalt anvendte tegn i det romerske talsystem er:
I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000

Det additive i det romerske talsystem fungerer på den måde at man simpelthen lægger de enkelte tegns værdi sammen, således at XXXII er tallet 32.
Der har dog lidt betydning hvordan tegnene står, nemlig hvis et mindre tegn står foran et større. Tallet 4 skrives som IV, mens 6 skrives som VI.

MCMLXXIV er således 1974.

I det romerske talsystem er der ikke et tegn for 0, da man ikke har brug for det, til at angive tal med.

Romertallene har den begrænsning at de ikke kan angive særligt store tal, da man normalt kun anvender tegn op til M=1000.
Nogen steder anvender man den notation, at man sætter en streg over, hvilket ganger tallet med 1000, så $\overline{X}$ betyder 10.000, men det betyder at man stadig ikke kan angive tal større end ca. 5.000.000.

Positions-talsystemer.

Når vi kigger på et tal i et positionsbestemt talsystem, så er det ikke blot tallet (cifferet) der betyder noget, men også hvor i tallet det er placeret. Tager vi vores daglige decimalsystem, så siger værdien af et ciffer 10 gange for hver position vi rykker til venstre, og det er netop grundtallet (10) også kaldet radix.

For et positionsbestemt talsystem er det sådan, at værdien af et givet ciffer er grundtallet gange større end samme ciffer i positionen umiddelbart til højre for cifferet.

Hvis der kan være nogen tvivl om hvilket grundtal et tal har, så skriver man det normalt efter tallet, så man skriver 543 i ti-talsystemet som 543_{10}.

Af de positionsbestemte talsystemer der anvendes i dag skal nævnes følgende:

Talsystem	Grundtal	Symboler (cifre)
Binært	2	0, 1
Oktalt	8	0, 1, 2, 3, 4, 5, 6, 7
Decimalt	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimalt	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Positions-talsystemer, historisk set

Op gennem historien har der været anvendt forskellige talsystemer i de forskellige kulturer, babylonerne anvendte et talsystem baseret på grundtallet 60 i deres kileskrift, det kendes helt tilbage fra 1800 f.Kr. Mayaerne havde et talsystem med grundtallet 20, som de anvendte fra 400 e.Kr. Og vores eget 10-talsystem stammer oprindeligt fra Indien, hvor det blev brugt fra 500 e.Kr. dog først med anvendelsen af 0 som et ciffer fra omkring 700 e.Kr. Det der er enestående ved tegnet nul, "0", der gør det muligt at nedskrive, at en talklasse (enere, tiere, hundreder...) er tom. Hvis man ikke havde tegnet 0, så ville det være svært at skrive 102, alle andre forsøg på at skrive det i et positionssystem vil bare blive til, at man opfinder et andet tegn for 0, men med samme betydning.

Umiddelbart er det binære talsystem (2-talsystemet) nært knyttet sammen med computere, men det stammer helt tilbage fra før år 0, og har været anvendt i grene af matematikken før det blev oplagt at beskrive digital elektronik ved hjælp af det binære talsystem. Det er ikke kun inde i computeren de binære tal anvendes, det anvendes fx også ved lagring af data på medier som hulkort/-strimmel, Magnetbånd, fx DAT, Magnetstribe fx på id-kort, kreditkort etc., cd, DVD og Harddisk.

Ti-talsystemet

De tal vi anvender til daglig er ti-talsystemet, der er et positionssystem med grundtallet 10. Det betyder at de cifre vi anvender er fra 0 til 9, og at den plads cifferet står på betyder hvilken værdi det har. Vi tænker ikke særligt over det i det daglige at tallet 575 skal læses som 5 hundreder, 7 tiere og 5 enere, men for at få sammenhængen til de andre positionssystemer, kan det være en fordel lige at betragte tallet i forhold til grundtallet, så 3576 kan udtrykkes ved cifrene i tallet og grundtallet opløftet i en potens afhængigt af cifferets position, så $3576 = 3*10^3+5*10^2 + 7*10^1+6*10^0$.

Det binære talsystem.

Det binære talsystem har grundtallet 2, og der er derfor kun brug for 2 cifre, nemlig 0 og 1. Det binære talsystem er opbygget som titalssystemet. Ser vi på cifrene i et binært tal fra højre mod venstre, repræsenterer det første enerne, det næste toerne, herefter firerne, otterne osv. - hele tiden det dobbelte af det foregående. Man kan angive 0 eller 1 ved hver position, således at tallet "10" er tallet 2 i titals-systemet. "110" er således 6, fordi den yderste venstre position angiver 4, den næste 2.

I computersammenhæng kalder man et 1-tal for on (tændt) og 0 for off (slukket). Tallets værdi er summen af de tændte positioner. En sådan størrelse, en position som kun har to mulige tilstande, kaldes en bit. Dette er den mindste enhed der arbejdes med i computeren. Når man taler om en bit, siger man ofte at den er sat/slettet i stedet for tændt/slukket.

Tager vi et lidt større binært tal f.x. 10010010, og vil regne ud hvad tallet er, så kan man gøre det på samme måde som i 10-talsystemet, hvor man bare bruger 2 som grundtal i stedet for, altså regnes det som:
$1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 128 + 16 + 2 = 144$.

Hvis man vil omskrive et tal fra 10-talsystemet til 2-talsystemet, så kan man bruge følgende metode:

1) Husk på tallet der skal omskrives, som vi herefter kalder konverteringstallet.
2) Fortsæt indtil konverteringstallet er under 1
3) Er konverteringstallet lige
4) Så skriv et 0 i resultatet (start ved mindst betydende ciffer)
5) Ellers skriv 1 i resultatet
6) Divider koverteringstallet med 2 og smid resten væk
7) Fortsæt ved 3) hvis konverteringstallet er over 0

Hvis man anvender metoden på 180 vil det gøres på følgende måde:

Vi husker konverteringstallet på de 180
180 er lige, så der skal stå 0 bagerst i resultatet og konverteringstallet bliver til 90.
90 er lige, næste ciffer bliver 0, resultatet er nu 00 og konverteringstallet bliver til 45.
45 er ulige, næste ciffer bliver 1 resultatet nu 100 og konverteringstallet bliver til 22.
22 er lige, næste ciffer bliver 0 resultatet nu 0100 og konverteringstallet bliver til 11.
11 er ulige, næste ciffer bliver 1 resultatet nu 10100 og konverteringstallet bliver til 5.
5 er ulige, næste ciffer bliver 1 resultatet nu 110100 og konverteringstallet bliver til 2.
2 er lige, næste ciffer bliver 0 resultatet nu 0110100 og konverteringstallet bliver til 1.
1 er ulige, næste ciffer bliver 1 resultatet nu 10110100 og konverteringstallet bliver til 0.
Vi er færdige, altså bliver resultatet nu 10110100

Omskrevet til binære tal bliver 180 altså til 10110100

I en computer er det meste organiseret i bytes, der i dag normalt er en samling på 8 bits, som kan angive en af $2^8 = 256$ tilstande, og som tal vil den kunne indeholde fra 0 til 255 (00000000 til 11111111), eller hvis man lader den mest betydende bit være et fortegn, så vil den kunne indeholde fra -128 til 127. 8 bits kan altså tilsammen svare til 256 forskellige tilstandsformer eller værdier. Typisk vil man også kunne have en enkelt karakter liggende i en byte (hver karakter har sin egen ASCII talværdi).
En byte er den lagerenhed i en computer, som bruges til at rumme information svarende til ét tegn. I byte-adresserbare computere er en byte tillige den mindste adresserbare enhed. Næsten alle computere er i dag byte-adresserbare. Byte er en omskrivning af bite (engelsk for bid). Omskrivningen er foretaget for at undgå forveksling med bit. Oprindelig bestod en byte af op til 6 bits, men i dag består den næsten altid af 8 bits. Andre størrelser som 9 og 12 bits har også været brugt.

Det hexadecimale talsystem

Når man arbejder med tal, så er det ikke altid at man kan klare sig med 0-255, så derfor er det nødvendigt at bruge flere bytes sammen, og så bliver det straks sværere at overskue alle de nuller og etter, derfor har man valgt at anvende endnu et talsystem, med grundtallet 16.

Fordelen ved et grundtal på 16 er, at det dækker præcis 4 bit ved at 4 bit i binære tal er fra 0000 til 1111. de første 10 cifre i det hexadecimale talsystem er taget fra 10-talsystemet og de resterende 6 er A til F

Decimalt	Binært	Hex-ciffer
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Fordelen ved at det er 4 bit er, at to hex-cifre er netop en byte. En byte kan indeholde fra 00 til FF, hvilket svarer til 0 – 255.

Omregning fra Hex-tal til decimaltal foregår på samme måde som ved omregning mellem andre talsystemer, her med grundtallet 16.
F.x. tallet 9C kan omregnes som følger $9*16^1 + 12*16^0 = 144 + 12 = 156$.

Omregning fra decimaltal til hex-tal er lidt anderledes end at omregne til binær tal, men det følger samme principper

1) Husk på tallet der skal omskrive, som vi herefter kalder konverteringtallet.
2) Fortsæt indtil konverteringstallet er 0
3) divider med grundtallet 16, som heltalsdivision
4) Resten fra heltalsdivisionen er det ciffer der skal bruges.
5) Første gang er mindst betydende ciffer, herefter sættes de foran
6) Sæt konverteringstallet til resultatet fra heltalsdivisionen
7) Fortsæt ved 2)

Anvender man metoden på decimaltallet 20100 vil det være som følger:

Vi husker konverteringstallet på de 20100
20100 / 16 = 1256 rest 4, konverteringstallet bliver 1256 vi skriver 4 bagerst i resultatet.
1256 / 16 = 78 rest 8, konverteringstallet bliver 78, og der står 84 i resultatet
78 / 16 = 4 rest 14, altså E konverteringstallet bliver 4, og der står E84 i resultatet
4/ 16 = 0 rest 4, konverteringstallet bliver 0 og der står 4E84 i resultatet
Da vi er kommet til 0, så er vi færdige med resultatet $20100_{10} = 4E84_{16}$.
Det oktale talsystem

Det oktale talsystem

Det oktale talsystem har grundtallet 8, og er lavet ud fra samme princip som det hexadecimale talsystem, bare at det kun dækker 3 bit ad gangen, hvilket nok er grunden til at det ikke er så udbredt i dag, da de fleste systemer arbejder med 8 bit ad gangen, og det går dårligt op i en gruppering med 3 bit, men for fuldstændighedens skyld skal det alligevel omtales.

543_8 omregnes til decimaltal som følger:
$5 * 82 + 4 * 81 + 3 * 80 = 5 * 64 + 4 * 8 + 3 * 1 = 320 + 32 + 3 = 355_{10}$

Skal man konvertere til Oktale tal, så bruger man samme princip som ved hexadecimale tal, blot med grundtallet 8. Her illustreret med 683_{10}.

Vi husker konverteringstallet på de 683
683 / 8 = 85 rest 3, konverteringstallet bliver 85 vi skriver 3 bagerst i resultatet.
85 / 8 = 10 rest 5, konverteringstallet bliver 10, og der står 53 i resultatet
10 / 8 = 1 rest 2, konverteringstallet bliver 1, og der står 253 i resultatet
1 / 8 = 0 rest 1, konverteringstallet bliver 0, og der står 1253 i resultatet
Da der står 0 i konverteringstallet, så er vi færdige, $683_{10} = 1253_8$.

Oversigt over de forskellige talsystemer

Binært	Oktalt	Decimalt	Hexadecimalt
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10
10001	21	17	11
10010	22	18	12
111110	76	62	3E
111111	77	63	3F
1000000	100	64	40
1000001	101	65	41
1100011	143	99	63
1100100	144	100	64
1100101	145	101	65
1100110	146	102	66
1111110	176	126	7E
1111111	177	127	7F
10000000	200	128	80
10000001	201	129	81
11111110	376	254	FE
11111111	377	255	FF
100000000	400	256	100
100000001	401	257	101

Konverteringsmetoder mellem talsystemer

Alt efter hvad man skal konvertere, så er der forskellige metoder der skal benyttes. Det kan sammenfattes i 3 forskellige metoder:

Potensmetoden – når man skal konvertere fra et tal i et eller andet talsystem over til decimalsystemet.
Divisionsmetoden – når man skal konvertere fra decimalsystemet til et tal i et andet talsystem.
Den direkte metode – når man skal konvertere mellem talsystemer der har et grundtal med en potens af 2, altså det binære, det oktale og det hexadecimale talsystem.

Potensmetoden

Tallet konverteres ciffer for ciffer, ud fra grundtallet.
Hvert enkelt ciffer ganges med grundtallet opløftet i den potens der svarer til den position cifferet har i tallet. Alle produkterne summeres.

Her vises et par eksempler:

10010110_2 med grundtallet 2 (binært) konverteres til decimaltal.

$1 * 2^7 = 128$
$0 * 2^6 = 0$
$0 * 2^5 = 0$
$1 * 2^4 = 16$
$0 * 2^3 = 0$
$1 * 2^2 = 4$
$1 * 2^1 = 2$
$0 * 2^0 = 0$
$10010110_2 = 150_{10}$

$2EC5_{16}$ med grundtallet 16 (hexadecimalt) konverteres til decimaltal.

$2 * 16^3 = 8192$
E: $14 * 16^2 = 3584$
C: $12 * 16^1 = 192$
$5 * 16^0 = 5$
$2EC5_{16} = 11973_{10}$

Det er her **vigtigt** at huske på, at det ciffer der står længst til højre i tallet (mindst betydende ciffer) **ALTID** skal ganges med grundtallet i 0. potens, altså med 1.
Resten af cifrene skal så ganges med grundtallet i den potens alt efter hvor cifferet står.
Så inden man starter med at konvertere, så tæl fra 0 for cifferet til højre, og op til det mest betydende ciffer.

Divisionsmetoden

Tallet i det andet talsystem opbygges ciffer for ciffer ud fra tallet der skal konverteres og grundtallet.

Tallet der skal konverteres, divideres med grundtallet, og den rest der bliver ved divisionen bliver det mindst betydende ciffer (helt til højre). Resultatet af divisionen divideres igen med grundtallet, og den nye rest bliver til næste ciffer, mens resultatet gemmes til beregningen af næste ciffer. Sådan fortsættes indtil resultatet bliver 0.

22653_{10} skal konverteres til hexadecimal

22653 / 16 = 1415 med rest 13: D
1415 / 16 = 88 med rest 7
88 / 16 = 5 med rest 8
5 / 16 = 0 med rest 5

Resultatet sættes sammen til $587D_{16}$.
Altså $22653_{10} = 587D_{16}$.

147_{10} skal konverteres til binært

147 / 2 = 73 med rest 1
73 / 2 = 36 med rest 1
36 / 2 = 18 med rest 0
18 / 2 = 9 med rest 0
9 / 2 = 4 med rest 1
4 / 2 = 2 med rest 0
2 / 2 = 1 med rest 0
1 / 2 = 0 med rest 1

Resultatet sættes samen til 10010011_2.
Altså $147_{10} = 10010011_2$.

Det kan umiddelbart være svært at se hvad der sker, men hvis man gerne vil have **FORSTÅELSEN** for det, så kan vi kigge på et eksempel vi kan overskue.
Vi kan illustrere hvad det er der sker, ved at konvertere et tal i 10-talsystemet til det samme tal i 10-talsystemet. (det har kun det formål at øge forståelsen).

2805_{10} konverterer vi til 10-talsystemet:

2805 / 10 = 280 med rest 5
280 / 10 = 28 med rest 0
28 / 10 = 2 med rest 8
2 / 10 = 0 med rest 2

Det vi kan observere er, at hver gang vi dividerer, så er det cifferet til højre i det vi har tilbage, der kommer ud som rest, og resultatet af divisionen indeholder det der endnu ikke er konverteret. Vi

skal lige lægge mærke til to ting: ved 280 / 10 = 28 får vi rest 0 – denne rest er lige så vigtig, da 0 også er et vigtigt ciffer – ved 28 / 10 = 2 rest 8, der er resultatet af divisionen 2 (heltallet af 2,8).

Den direkte metode

Den direkte metode går ud på, at man kan konvertere ciffer for ciffer til binært, hvis det er tal med grundtal der selv er en potens af 2, altså oktal ($8 = 2^3$) eller hexadecimalt ($16 = 2^4$).

Ind til man har lært cifrene som binære mønstre, så kan det være en fordel at anvende en tabel, som f.x. tabellen på side 5.

Hvis man vil konvertere fra hexadecimalt til oktalt, så gør man det ved at konvertere det hexadecimale tal til binært, og så gruppere de i 3 bit ad gangen (fra mindst betydende), og så konvertere til oktal. Det kan lyde lidt omstændeligt, men illustreres bedst ved et på eksempler:

$587D_{16}$ skal konverteres til oktal:

$587D_{16} = 0011\ 1000\ 0111\ 1101_2 = 011\ 100\ 001\ 111\ 101_2 = 34175_8$.

175432_8 skal konverteres til hexadecimal:

$175432_8 = 001\ 111\ 101\ 100\ 011\ 010_2 = 1111\ 1011\ 0001\ 1010_2 = FB1A_{16}$.

Anvendelser

Meget af anvendelserne af talsystemerne sker, når man udvikler programmer og andet, men det stikker hovedet op nogen andre steder.

Hvis man ser en farvekoden der hedder #00FF00, så kan man betragte den som en fast kode, som man kan finde i et program, men rent faktisk er der tale om 3 tal.

Det hedder en RGB farvekode, og hvis man kigger på den, så er det #RRGGBB, som altså er 3 hexadecimale tal. Vender vi tilbage til #00FF00, så er RR (den røde farve) 00, GG (den grønne farve) er FF og BB (den blå farve) er 00. Det betyder at den angivne farve ikke indeholder noget rød (den er 0), men indeholder $FF_{16} = 255_{10}$ eller fuldt blus på grøn og ikke noget blå, så det er farvekoden for helt klar grøn.

Med den viden kan man faktisk justere på sine farver, ikke bare ved at prøve sig frem, hvordan de ser ud, men ud fra deres aktuelle indhold af de tre dele rød, grøn og blå.