Det binære talsystem og lidt om, hvordan computeren virker

Det binære talsystem

Hvordan er det muligt at en computer kan udføre de mange forbløffende ting, den kan?
Havde romerne fået overtaget, så vi i dag brugte deres talsystem, så var computere aldrig blevet til virkelighed.

Romerne brugte bogstaver som I, V, X, L, C, D og M. De kendte ikke til tallet nul.
Hvis romerne skulle angive, at der havde været 605 tilskuere til er gladiatorkamp, ville de skrive: DCV. Havde der været 616, skulle der stå: DCXVI
Heldigvis fik araberne indflydelse. De indførte taltegnene 0 -9 og opfandt positionssystemet. På en overskuelig måde kan man nu skrive enhver værdi med blot 10 forskellige tegn.

605

Tallet sekshundredeogfem består således af 5 enere, 0 tiere og 6 hundreder.
Når man bevæger sig fra højre mod venstre vil værdien af taltegnet hele tiden forøges med en faktor på 10. Da romerne ikke kendte til eksistensen af nullet, var der derfor ikke muligt at lave et tilsvarende system.
At regne med romertallene var en besværlig sag.
Vi har alle i skolen lært at regne med arabertallene, og det fornemmes ganske enkelt, når man blot følger nogle simple regler.

Computere kan kun arbejde med 2 stadier f. eks. åben-lukket, lav eller høj spænding, nordpol eller sydpol.
En computer arbejder derfor med et talsystem, der kun indeholder 2 tegn: 0 eller 1.
Dette talsystem kaldes to-tal- systemet eller det binære talsystem. (bi her i betydningen 2).
Et tegn hedder i computerverdenen for en Bit, altså et ettal eller et nul.

I det følgende vil det binære talsystem blive gennemgået.
Det bedst kendte talsystem er ti-talsystemet. Det består af 10 taltegn/cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Næste tal i talrækken hedder ti, men der findes ikke noget tegn for ti, det må derfor opbygges af tegn fra de ti grundtegn.
Tallet ni kan også skrives: 9,00 eller 009. De har samme betydning som 9; men bytter vi om på cifrene f. eks. sådan: 900, får nullerne pludselig en væsentlig betydning. Det talsystem, vi anvender, kaldes et positionssystem. Cifrene får betydning ud fra den plads i tallet, hvor de står. Tallet ti opbygges derfor med cifrene 1 og 0 således: 10. Hundrede skrives således: 100, tusind: 1000 og attenhundredetooghalv-fems: 1892. Her står 1 på tusindernes plads, 8 på hundrernes, 9 på tiernes og 2 på enernes plads. Når man bevæger sig fra en position til den næste, bruges tallet ti til at forøge eller mindske cifferets værdi.

NB! I det følgende afsnit vil tal skrevet i to-talsystemet være vist i kursiv/skråskrift.

I det binære talsystem - to-talsystemet findes kun to cifre. Grundtegnene er: *0* og *1*. Som ved ti-talsystemet findes ikke noget tegn, der kan repræsentere det næste tal i talrækken, det må derfor på samme måde opbygges i positioner ved hjælp af grundtegnene. Næste tal må derfor være: *10*, der repræsenterer værdien 2. 3 er lig *11*, og 4 skrives som: *100* o. s. v.

Herunder er vist en omsætningstabel over de første 16 tal i talrækken:

0 =	*0*		
1 =	*1*	9 =	*1001*
2 =	*10*	10 =	*1010*
3 =	*11*	11 =	*1011*
4 =	*100*	12 =	*1100*
5 =	*101*	13 =	*1101*
6 =	*110*	14 =	*1110*
7 =	*111*	15 =	*1111*
8 =	*1000*	16 =	*10000*

Vi observerer, at det er nemt at se om et tal er lige eller ulige. Vi ser også, at der er to etcifrede tal, to tocifrede, fire trecifrede, otte firecifrede og ?? femcifrede.

Ja, det kan være lidt svært at se; men der må vel være et system, hvorefter man kan bestemme antallet?
Antal etcifrede fås ved at beregne 2 -0 = 2
Tocifrede: 4 - 2 = 2
Trecifrede: 8 - 4 = 4
Firecifrede: 16 - 8 = 8
Femcifrede: ?? -16 = __

Vi må altså have bestemt det næste tal, hvor der skal benyttes en ny position.

I ti-talsystemet vokser antallet af cifre ved 10, 100, 1000, 10000 o. s. v. Hvis vi igen ser på oversigten, kan man iagttage, at antallet af cifre/positioner forøges med 1 ved: 2, 4, 8 og 16

10^1 =	10	2^1 =	2
10^2 =	100	2^2 =	4
10^3 =	1000	2^3 =	8
10^4 =	10000	2^4 =	16

Af oversigten ses, at hver gang en ny potens af grundtallet nås, så øges antallet af cifre med 1. Det tal, vi leder efter må derfor være 2^5, som er lig 32.
Antal fem-cifrede tal må derfor være: 32 -16 = 16

Hvor vi i ti-talsystemet taler om enere, tiere, hundreder o. s. v., kan vi i to-talsystemet altså tale om: enere, toere, firere, ottere o. s. v. Dette har betydning for tolkningen af et binært tals størrelse.

Det binære tal: *101101* må således have værdien:
En 1'er, nul 2'er, en 4'er, en 8'er, nul 16'er og en 32'er der kan udregnes til at være:
1 + 0 + 4 + 8 + 0 +32 =45.
101101 er altså lig 45

Hvad er det største tal, der kan skrives med otte cifre i det binære talsystem?

11111111 = 128 + 64 + 32 + 16 + 8 + 4 + 2+ 1 = 255.
Hvis vi medregner nul, vil der altså kunne laves 256 forskellige tal med op til 8 cifrede binære tal.

255 kan også skrives:
11111111 = $2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

2^3 betyder 2 * 2 * 2.

I ti-talsystemet ændres værdien fra position til position med 10. Ved at iagttage denne opstilling ses det, at positionsændringer i det binære system sker med faktoren 2.

Vi har ovenover set en metode til at omregne værdien af et binært tal til ti-talsystemet.
Nu skal vi se, hvorledes man kan regne modsat. Her kan tallene 1, 2, 4, 8, 16, 32, 64.... være nyttige.

Vi prøver med tallet 29. Fra talrækken her over ses, at 32 er for stort; men 16 er indeholdt i 29, der må altså sættes en markering - et ettal - i den position, der markerer 16'erne. Når 16 trækkes ud af 29 er der 13 til rest. 8 er indeholdt i 13, altså markeres 8-positionen. Resten bliver 5 (13 -8). Der skal således også markeres i 4'rne. Resten bliver nu 1. Den indeholder ingen 2'er. Da der ingen 2'er er, markeres det med et nul. 1 markeres med et ettal i 1'er positionen.

29 = 16 + 8 + 4 +0 + 1 29 = *11101*

Hvis man skal omsætte tallet: 1892 på denne måde, bliver det lidt omstændelig og måske også lidt vanskelig. Vi vil derfor se på en anden metode til at omsætte fra ti-talsystemet til to-tal-systemet.
29 kan deles med 2. Det bliver 14 med en rest på 1
14 deles med 2. Det bliver 7 med resten 0

```
29 : 2 = 14          rest 1
14 : 2 =  7          rest 0
 7 : 2 =  3          rest 1
 3 : 2 =  1          rest 1
 1 : 2 =  0          rest 1
```

Her er resterne interessante. Prøv at se på dem nedefra og opad: *11101*, det giver samme resultat som vist ovenfor.

Her vises endnu et eksempel. Lad os tage 1892.

```
1892 : 2 = 946        rest 0      1'ere
 946 : 2 = 473        rest 0      2'ere
 473 : 2 = 236        rest 1      4'ere
 236 : 2 = 118        rest 0      8'ere
 118 : 2 =  59        rest 0     16'ere
  59 : 2 =  29        rest 1     32'ere
  29 : 2 =  14        rest 1     64'ere
  14 : 2 =   7        rest 0    128'ere
   7 : 2 =   3        rest 1    256'ere
   3 : 2 =   1        rest 1    512'ere
   1 : 2 =   0        rest 1   1024'ere
```

1892 = 1024 + 512 + 256 + 0 +64 + 32 + 0 +0 +4+0 +0

1892 er altså et 11-cifret tal med cifrerne: *11101100100*

Vigtige værdier i det binære system er altså:

0, 1, 2, 4, 8, 16, 31, 64, 128, 226, 512, 1024, ……

Hver gang et af disse tal overskrides, skal der bruges en ny position.

Lidt om, hvorledes computeren anvender det binære talsystem

Når der arbejdes med det binære talsystem, hører man ofte ordene bit og byte nævnt. Ordet bit kommer fra den engelske betegnelse: BInary digiT, der betyder binært et-cifret tal.

En Bit er altså et enten - eller. 0 eller 1. En lukket eller åben situation. En høj eller en lav spænding. Lys eller mørke. Nordpol eller sydpol.

Man opererer også med begrebet Byte. En Byte er en talværdi fra 0 til 255. Når den skrives binært, skal den indeholde 8 cifre.

En bit er altså et taltegn. Det kan enten være et nul eller et ettal, andre muligheder findes ikke. Tallet 17 = *10001* er et 5-cifret binært tal. Det indeholder således 5 bit

I computere anvendes ofte 8-bit tal. Tallet *10001* kan skrives således: *00010001*. Det er stadig et 5-cifret tal; men det består nu af 8 bit. Et tal med 8 bit kaldes en byte. I computere sendes og gemmes alle mulige tegn ofte i bytes. Det vil sige, at tegnene - bogstaver, tal m. m. - er omsat til talkoder i et 8-bit mønster.

Når du taster et A på tastaturet, så ses det på skærmen. På de gamle skrivemaskiner var der svingarme, som via et farvebånd afsatte et tegn på papiret. På et moderne tastatur sender tasterne en 8 ciftet talkode – en Byte - ind i computeren. Her tolkes koden, og resultatet vises som et tegn på skærmen.

Der findes 256 forskellige tal, når der kan anvendes 8 cifre i tallene, hvis vi medtager nul. Det betyder, at datamater, der gemmer tegn i bytes, kan operere med 256 forskellige tegn. De forskellige tegn med tilhørende talkoder er for størstedelen fastlagt internationalt i en såkaldt ASCII-kode tabel.
I det danske alfabet er der 28 forskellige bogstaver. Nej, der er i virkeligheder 56, da store og små bogstaver har forskellig talkode. Der er 10 regnetegn, hertil kommer parenteser, anførselstegn, komma, kolon, plus, minus m.m. Altså ca. 100 forskellige tegn.
Tastaturet har godt 60 forskellige taster, der via Shift-tasten og Alt Gr-tasten rummer de meste brugte tegn. Men der stadig langt til alle 256 muligheder er opbrugt.

På de sidste sider har jeg lavet en oversigt over alle de tegn, man kan anvende via tastaturet.

For at bruge de mindre kendte tegn, skal man kunne et lille trick. Det får du her.

Forudsætningen er, at tastaturet har et taltastatur til højre på tastaturet.

Du vil måske skrive kvadratmeter. Normalt skrives det sådan: m²; men hvor finder man det lille total?
Hold Alt-tasten nede og skriv 253 via taltastaturet.
Du har lavet en skrivelse, som du har copyright til. Hold Alt-tasten nede og tast 184. ©.

Metoden er altså

1. at kende tegnets kode (se oversigten på de sidste sider).
2. holde Alt-tasten nede
3. taste koden på taltastaturet.

I ASCII tabellen vil det lille l findes på plads nr. 108, medens ettallet findes på plads nr. 49. Det forklarer det problem, som mange begyndere har stødt på, når de fra alm. skrivemaskiner begynder at anvende tastaturet på en PC: l og 1, og 0 og o er ikke det samme tegn. Så lo og 10 er ikke, som i gamle dage synonymer.

Det store A findes på plads nr. 65, medens det lille a vil findes på plads nr. 97.

Lad os se på bitmønstrerne for A og a og for T og t:

A ¸ 65 = 64 + 1	*01000001*
a ¸ 97 = 64 + 32 + 1	*01100001*
T ¸ 84 = 64 + 16 + 4	*01010100*
t ¸ 116 = 64 + 32 + 16 + 4	*01110100*

Dette viser tilsvarende, at der er forskel på store og små bogstaver. Læg også mærke til, at der er et mønster i disse forskelle: alle små bogstaver har en værdi, der er 32 større end det tilsvarende bogstav skrevet med stort. Det er derfor relativt nemt at omsætte små bogstaver til store eller omvendt. Til gengæld vil en sortering ofte give problemer, hvis der ikke tages højde for dette forhold.

Bemærk også, at æ, ø og å står langt nede i rækken. Hvis du er på ferie i udlandet, vil du lede forgæves efter disse tegn. De anvendes jo kun i Norge og Danmark, så man har måttet finde en alternativ placering for disse tegn.

En ret kendt og udbredt hjemmecomputer i fordums tid hed: Comodore 64. Det er ikke tilfældigt. Tallet 64 siger noget om maskinens lagerstørrelse. En computers lagerkapacitet måles i KB, der er en forkortelse af KiloBytes. Kilo betyder i almindelighed 1000, det gør det ikke her, ikke helt i hvert fald. I det binære system er et kilo - af gode grunde - lig med 1024. 64 i navnet betyder, at maskinen havde en hukommelse (RAM) på 64 KB. Ved en simpel beregning kan man let finde det antal tegn, som maskinen kan rumme i hukommelsen på samme tid: 64 * 1024 = 65536.
Sagt på en anden måde: gennemsnittet af tegn pr. side i dette materiale er ca. 3000 tegn.
Computerens hukommelse kunne således indeholde ca. 22 sider af dette materiale.
Moderne computeres lagerkapacitet er formidabelt set i forhold til fordums tid. Nu måles hukommelsen i GB!!

På harddisken, som kort er beskrevet tidligere, gemmes programmer, filer m.m. Hvordan mon?

Overfladen af den runde skive er belagt med et materiale, som kan magnetiseres. Her kan altså gemmes nordpole og sydpole, som kan identificeres som nuller og ettaller. Når f. eks. et A gemmes så laves der altså en nordpol, en sydpol, 5 nordpole og en sydpol i form af små magnetpletter på skiven.

På en DVD eller en CD brændes der små eller dybe huller i pladen, disse vil så kunne identificeres igen som nuller eller ettaller.

Ovenstående var måske svært. Til slut en bemærkning, som er helt uforståelig.

En harddisk er i dag typisk på 300 GB, altså kan den indeholde 300.000.000.000 bogstaver og andre tegn. Dette tal skal så ganges med 8 for at få antallet af magnetpletter.
Hvordan Søren har teknikkerne kunnet finde plads til dette astronomiske tal på en lille rund plade? Det er vist kun nørder, der kender svaret på det.

Hvis du vil kende min mening, så er Det binære talsystem simpelthen genialt!!

Lyst til at lege med de binære tal?

Hvis ovenstående sider ikke har taget modet fra dig, så kunne du måske have lyst til arbejde lidt videre med det binære talsystem.

Man regner med de binære tal på samme måde, som vi gør i ti-talsystemet. Det er nemmere, men lidt mere tid/plads krævende.

Addition:

```
 1                    1 1 1 1
213                  11010101
118                   1110110
---                  --------
331                 101001011
```

Udregningerne kræver måske nogen kommentarer?

3 + 8 = 11. Eneren skrives, og tieren sættes op 'på loftet', det lærer man allerede i 1. eller 2. klasse. På nøjagtig samme måde arbejdes der i det binære system. *1 + 0 = 1* det giver ingen problemer. *0 + 1 = 1* heller ikke. *1 + 1 = 10* nullet skrives og menten en sættes 'på loftet'. Altså ingen forskel i beregningsmetoden, hvis man blot kan sin additionstabel. Den er yderst simpel og vises herunder:

Et udsnit af en additionstabel for 10-talssystemet:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

En fuldstændig additionstabel for det binære talsystem:

+	0	1
0	0	1
1	1	10

Subtraktion.

Computere kan ikke trække to tal fra hinanden, (at låne er alt for kompliceret selv for en computer), de kan heller ikke gange eller dele efter de metoder, vi lærte i skolen. Ikke desto mindre er de et fantastisk redskab til alt, hvad der vedrører tal. Ergo må de benytte andre metoder end dem, vi anvender.

Som antydet tidligere kan computere lægge tal sammen. Nu skal vi se en metode, hvor subtraktion udføres ved addition.

Vi indfører et begreb, der hedder komplementære tal. Vi kender begrebet fra komplementære farver.

I 10-talsystemet findes disse komplementære tal:

0 - 9; 1 – 8; 2 – 7; 3 – 6; 4 - 5.

2 er eksempelvis komplement til 7, og 5 er komplement til 4.
Man bemærker, at summen af komplement tallene altid giver sidste tal i talrækken, altså 9.

Lad os starte med at subtrahere to tal ved at anvende addition.

```
 17
- 7
---
 10
```

Dette simple subtraktionsstykke vil vi nu beregne ved at anvende komplement tal.

```
 17
-07
---
 10
```

Regnestykket omskrives som vist herover. Bemærk nullet foran 7.

07 erstattes med de tilsvarende komplement tal, og minustegnet ændres til plus, det var jo hensigten at udregne stykket udelukkende ved at anvende addition.

```
 1
 17
+92
---
 09
```

Som det ses, bliver der en mente på 1. Den fanges på vej ned og placeres på enernes plads under 9, hvor den adderes.

```
 17
+92
---
 09
+ 1
---
 10
```

Hokus Pokus: resultatet er det samme. Det var besværligt, måske uoverskueligt; men målet blev nået: at subtrahere udelukkende ved brug af addition.

Vi skal lidt senere gøre det samme i det binære talsystem, hvor der vil vise sig indlysende fordele ved denne beregningsmetode.

Inden da vises endnu et eksempel:

```
  560                  (1)(1)
- 234                    560
-----                  + 765
  326                  -----
                         325
                       +   1
                       -----
                         326
```

Nu prøver vi det samme i det binære talsystem.
Her er komplement tallene til at overse: 0 - 1.
Summen giver 1, det højeste tal i talrækken.

```
 17                10001
 -7              - 00111
 --                -----
 10                 1010
```

-00111 omsættes til: *+11000*, vi får dette additionsstykke:

```
(1)                (1)
 17                10001
+92              + 11000
---                -----
 09                01001
  1                    1
---                -----
 10                 1010 = en toer og en otter
```

NB! Den sidste mente tillægges enerne!

Hvor er så fordelen ved denne 'besværlige' regne metode?
Den ligger helt klart i det binære talsystem. At ændre et binært tal til dets komplement tal er den nemmeste sag i verden: tallet gennemløbes ciffer for ciffer, og alle nuller ændres til ettaller, medens alle ettaller ændres til nuller. Herefter kan man regne alle subtraktioner, hvis man ved, hvordan tal adderes. Denne metode udnytter mange maskiner, der skal behandle mange tal, og de gør det lyn hurtigt.

Multiplikation.

17 * 7 = 17 + 17 + 17 + 17 + 17 + 17 + 17 = 119

Dette er en af de besværlige måder at løse problemet på.

Normal udregning sker ved hjælp af multiplikationstabeler:

```
17 * 7
------
    49
+   70
------
   119
```

I det binære talsystem kan man også multiplicere.

Gangetabellen er ret enkel:

```
* | 0 | 1
---------
0 | 0 | 0
---------
1 | 0 | 1
```

Det er beregningerne også:

```
 10001 * 111
------------
       10001
      100010
     1000100
------------
     1110111
```

1110111 = 64 + 32 + 16 +0 + 4 + 2 +1 = 119

Additionsmetoden kunne også anvendes her:

```
     1 1
   10001
   10001
   10001
   10001
   10001
   10001
   10001
  -------
  1110111
```

Mange regnemaskiner/computere benytter netop additionsmetoden. Disse maskiner er fabelagtig hurtige, men kan jo kun udføre de processer, som de er programmeret til. Det kan være vanskeligt at lære dem at multiplicere, hvorimod additionsprocessen er ret enkel.

Vi kan måske også være lige glade med, hvordan den bærer sig ad, blot den regner rigtigt. Denne lille vejledning er da også blot tænkt anvendt til de, der måtter være interesseret i at kende lidt til computerens mystiske og forunderlige verden.

Division.

Her vises kun et enkelt eksempel.

```
119 : 17 = 7
```

For oversigtens skyld vises udregninger efter en gammelkendt metode:

```
                    1110111 : 10001 = 111
Går op en gang    - 10001..
                    ------..
Træk fra             11001.
Går op en gang       10001.
                     -----.
Træk fra              10001
Går op en gang        10001
                      -----
Træk fra                  0
```

Dette er, som vi jo alle ved, en vanskelig proces. Den kunne afløses af en proces, der benytter subtraktion:

119 – 17 = 102
102 – 17 = 85
 85 – 17 = 68
 68 – 17 = 51
 51 - 17 = 34
 34 – 17 = 17
 17 – 17 = 0

Hvor mange regnestykker blev det til? 7!

119 delt med 17 er altså 7.

Det synes igen at være en besværlig måde at regne på; men da regnemaskiner som sagt er lynende hurtige, vil en metode som denne, hvor subtraktion kan erstattes af addition med komplement tal, reducere regnemaskinens beregninger til kun at skulle omfatte additioner.

Alle 4 grundlæggende regningsarter: addition, subtraktion, multiplikation og division kan altså i princippet udføres ene og alene ved hjælp af addition. Det betyder samtidig, at det program, der styrer processen, kan gøres ganske primitivt.

Dette afsnit skulle gerne give læseren et lille indblik i det binære talsystems 'mysterier', der jo ikke ligger langt fra det system, vi næsten føler, vi er født med at kunne forstå. Desuden skulle det gerne give forklaringen på, hvorfor dette system er udvalgt som det bedste kommunikationsmiddel vedr. computere.

Kode		Kode		Kode	
32	Mellemrum	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	

Kode		Kode		Kode		Kode	
128	Ç	160	á	192	Š	224	Ó
129	ü	161	í	193	‹	225	ß
130	é	162	ó	194	Œ	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	-	228	õ
133	à	165	Ñ	197	Ž	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202		234	Û
139	ï	171	½	203	‘	235	Ù
140	î	172	¼	204	’	236	ý
141	ì	173	¡	205	“	237	Ý
142	Ä	174	«	206	”	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	€	208	ð	240	-
145	æ	177	▒	209	Ð	241	±
146	Æ	178	‚	210	Ê	242	œ
147	ô	179	‖	211	Ë	243	¾
148	ö	180	„	212	È	244	¶
149	ò	181	Á	213	•	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	—	249	¨
154	Ü	186	†	218	˜	250	·
155	ø	187	‡	219	™	251	¹
156	£	188	ˆ	220	š	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	ž
159	*ƒ*	191	‰	223	›	255	Ÿ