Koder skjult i talsystemer

af Glenn Ganderup

Det binære talsystem

Computere benytter et specielt talsystem kaldet det binære talsystem (nogle gange omtalt som totalssystemet eller Base 2). Talsystemet ser noget anderledes ud end vores titalssystem (decimalsystemet), men egner sig rigtig godt til computere, fordi det kun består af nuller og ettaller. Disse værdier kan håndteres af elektroniske kredsløb via små komponenter, som enten kan være tændt (ladet op) eller slukket (afladet). Binære tal kan således både opbevares og transporteres rundt i computeren som elektroniske signaler.

Binære data kan også gemmes på en traditionel harddisk ved at magnetisere bittesmå områder på diskens overflade, så hvert område på disken kan repræsentere en nul- eller et-værdi, eller skrives direkte til flash memory (eksempelvis USB-sticks eller SSD-diske).

Det binære talsystem er opbygget således, at man læser værdier (0 eller 1) fra højre mod venstre. Den første (yderste højre) værdi repræsenterer enerne i vores titalssystem, den næste værdi toerne, den tredje firerne og så videre – hver plads mod venstre svarer til det dobbelte af den forrige værdi. Hver enkelt værdi kaldes en bit, og sammensætter man eksempelvis otte bit kan man således repræsentere et heltal mellem nul og 255 (eller i alt 256 værdier). Her er et par eksempler på, hvordan en binær værdi konverteres til en decimal værdi.

Position	7	6	5	4	3	2	1	0	
Decimal værdi	128	64	32	16	8	4	2	1	
Binær værdi	0	0	0	0	0	0	0	0	0 (0+0+0+0+0+0+0+0)
	0	0	0	0	0	0	0	1	1 (0+0+0+0+0+0+0+1)
	0	0	0	0	0	0	1	0	2 (0+0+0+0+0+0+2+0)
	0	0	0	0	0	0	1	1	3 (0+0+0+0+0+0+2+1)
	0	0	0	0	0	1	0	0	4 (0+0+0+0+0+4+0+0)
	0	0	0	0	0	1	0	1	5 (0+0+0+0+0+4+0+1)
	0	0	1	0	1	1	0	1	45 (0+0+32+0+8+4+0+1)
	0	1	1	0	0	1	0	0	100 (0+64+32+0+0+4+0+0)
	1	0	0	1	0	1	1	0	150 (128+0+0+16+0+4+2+0)
	1	1	0	0	1	0	0	0	200 (128+64+0+0+8+0+0+0)
	1	1	1	1	1	1	1	1	255 (128+64+32+16+8+4+2+1)

Binære værdier.

Bemærk at en decimal værdi kan beregnes ved at sammenlægge de decimale værdier for hver position, der indeholder en ener (tændt bit). Den binære værdi 101 kan således konverteres til 5 ved beregningen 4 + 0 + 1.

Som det fremgår, kan man altså ved hjælp af 8 bit repræsentere en værdi mellem 0 og 255, men man kan også vælge at bruge den yderste venstre bit (kaldes en fortegnsbit eller sign bit) til at vise, om der er tale om et positivt (værdien 0) eller negativt (værdien 1) tal. Således kan man bruge de 8 bit til at repræsentere en decimal værdi mellem -127 og 128. Hvis man benytter mere end otte bit,

kan man repræsentere et langt større heltal – eksempelvis kan man med 16 bit arbejde med heltal fra 0 til 65.535 eller -32.768 til 32.767.

Hvis man skal repræsentere reelle tal (kommatal) i det binære talsystem, bruger man typisk en standard kaldet floating-point (flydende komma). Ved hjælp af floating point kan man enten repræsentere meget store tal med få decimaler eller meget små tal med mange decimaler. Hvor præcist, man ønsker tallet, kan kontrolleres af antallet af bit, der benyttes, og det er faktisk en af de helt store faldgruber inden for programmering – hvis man benytter for få bit til at gemme et reelt tal, kan det få uheldige konsekvenser, fordi et tal kan risikere at blive afrundet af computeren.

Ligesom i vores titalssystem kan man foretage forskellige typer af beregninger som addition, subtraktion, multiplikation og division, men fordi det binære talsystem består af nuller og ettere, kan beregninger nemt og meget hurtigt klares af elektroniske kredsløb. Eksempelvis kan en binær addition ske ved at benytte menter som i titalssystemet:

Mente					1				
Binær værdi	0	0	0	0	1	0	1	0	10 (0+0+0+0+8+0+2+0)
Binær værdi	0	0	0	0	0	0	1	1	3 (0+0+0+0+0+0+2+1)
	0	0	0	0	1	1	0	1	13 (0+0+0+0+8+4+0+1)

Sammenlægning af binære værdier.

Reglerne ved binær addition er simple: 0+0=0, 1+0=1, 0+1=1 og 1+1=1 samt 1 i mente. Et andet eksempel på en simpel binær beregning er at multiplicere et tal med 2: her flyttes alle bit blot en plads til venstre. I efterfølgende eksempel ganges 10 med 2:

Binær værdi	0	0	0	0	1	0	1	0	10 (0+0+0+0+8+0+2+0)
Flyt 1 til venstre	0	0	0	1	0	1	0	0	20 (0+0+0+16+0+4+0+0)

Multiplikation af binære værdier.

Du behøver ikke sætte dig yderligere ind i binær matematik, men blot være bevidst om, at man med elektroniske kredsløb og det binære talsystem kan foretage beregninger – og at det med dagens computere kan foregå ufatteligt hurtigt. En af fordelene ved at benytte et nyere højniveau programmeringssprog er jo netop, at man ikke behøver tænke binært. Men derfor er det vigtigt at vide lidt om, hvad der sker under motorhjelmen.

Det hexadecimale talsystem

I visse situationer vil man gerne kunne vise en binær værdi på en anden måde end ettere og nuller – det gælder eksempelvis visse basale programmeringssprog som eksempelvis maskinsprog (Assembler). Hvis man skulle angive en 32 bit værdi, ville koden blive alt for kompleks og uoverskuelig. Derfor bruger man ofte i stedet det hexadecimale talsystem (kaldes også base 16).

I det hexadecimale talsystem har man bibeholdt værdierne 0-9, mens man har erstattet 10, 11, 12, 13, 14 og 15 med A, B, C, D, E, F:

Base 2 Binært	Base 10 Decimal	Base 16 Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Det hexadecimale talsystem.

Systemet fungerer ved at opdele den binære værdi i grupper af fire bit, der kan have værdien 0 (0000 binært) til 15 (1111 binært), og erstatte hver gruppe med en hexadecimal værdi (fra 0 til F). Eksempelvis kan decimalværdien 125 angives som 7D:

Binært	0111	1101	Binært	0001	1110	1101	1000
Hexadecimalt	7	D	Hexadecimalt	1	E	D	8

I moderne programmeringssprog støder man tit på hexadecimale værdier og de har typisk et foran- eller bagvedstillet specielt tegn som eksempelvis #1ED8, $1ED8 eller 1ED8h. Hexadecimale værdier benyttes også mange andre steder som eksempelvis i en URL, hvor specielle tegn angives i hexadecimal eller til at angive farver i HTML og CSS.

Tegntabeller

Computere arbejder med binære tal og kan derfor ikke som udgangspunk håndtere tegn og bogstaver. Derfor opfandt man allerede i 1960'erne en tegntabel kaldet ASCII (American Standard Code for Information Interchange) som stadig (dog i en udvidet version) benyttes som standard for, hvordan man kan repræsentere tegn med tal. I tabellen findes 128 tegn (7 bit) bestående af kontrolkarakterer, bogstaver, tal og specialtegn. Der er ingen grund til at nævne hele tabellen her - den kan hurtig findes på nettet ved en søgning – men de mest interessante elementer i tabellen er som følger:

Base 2 Binært	Base 10 Decimal	Base 16 Hexadecimal	Værdi
00000000	0	0	Null
00001000	8	8	Back
00001001	9	9	Tab
00001010	10	A	Line feed
00001100	12	C	Form feed
00001101	13	D	Carriage return
00110000	48	30	0
00110001	49	31	1
00110010	50	32	2
00110011	51	33	3
01000001	65	41	A
01000010	66	42	B
01000011	67	43	C
01000100	68	44	D
01100001	97	61	a
01100010	98	62	b
01100011	99	63	c
01100100	100	64	d

For at spare lidt plads i tabellen er der kun nævnt de første fire tegn i en typisk række (tal, store bogstaver og små bogstaver) – tegnene fortsætter i den forventede rækkefølge…

ACSII er oprindelig fra USA, og man kan derfor kun finde bogstaverne fra a-z. Derfor findes der flere andre standarder, som udvider ACSII. Den mest kendte er Unicode, hvor alle bogstaver fra alle sprog i verden er repræsenteret, men da der findes en del tegn er Unicode suppleret med forskellige typer af indkodninger for at spare mest mulig plads. De mest kendte herhjemme er UTF8 (8 bit) og UTF16 (16 bit). UTF er en forkortelse for Unicode Transfer Format. For yderligere information om Unicode og UTF henvises til http://unicode.org.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	^{n}
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	^{2}
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Konklusion: En computers informationer består af ettaller og nuller, i daglig tale også kaldt bit. Otte bit bliver til en Byte. Når du nu har den grundlæggende forståelse, for hvordan ASCII tegntabellen hænger sammen med de mest gænge talsystemer, så kan vi begynde at kigge på hvordan disse data kan bruges til at skjule et koordinat.

Vi starter med en af de nemme opgaver, en cache med koordinatet synligt i binært format, nemlig Talsystem #2 Binær/Grundtal 2 http://coord.info/GC33W85
Prøv om du kan løse den uden hjælpemidler, da det vil give dig den bedste træning.

Talsystem #1 – Romertal http://coord.info/GC33W8G
Talsystem #3 – Oktal/Grundtal 8 http://coord.info/GC33W7Z
Talsystem #4 – Urnemarkskultur http://coord.info/GC33W7N
Talsystem #5 – Hex/Grundtal http://coord.info/GC33W7A
Talsystem #6 – Vigesimal/Grundtal 20 http://coord.info/GC33W73
Talsystem #7 – NDOM 6-talssystem/Grundtal 6 http://coord.info/GC33W6M
Talsystem #8 – Seksagesimal/Grundtal 60 http://coord.info/GC33W67

Nåede du hertil, så er du klar til næste skridt, og det er alle de muligheder der ligger i kryptering af ovenstående værdier, f.eks. rotationer, farvekoder, mystiske fonte med mere.

Et godt værktøj til disse forskellig artede opgaver er GCC app'en til android, jeg ved ikke hvad den evt. måtte hedde til Iphone.

Andre nyttige værktøjer kan findes på:

http://www.convertworld.com/da/tal/ En super god konverteringsside.

http://cryptii.com/select/binary Kryptering imellem forskellige formater

http://en.wikipedia.org/wiki/List_of_numbers_in_various_languages Tal på mange sprog

http://dk-titan.dk/ DK Titans side, fyldt med værktøjer

https://www.google.dk Verdens bedste søgemaskine

http://www.angio.net/pi/bigpi.cgi Hjælpeprogram til pi

http://ezgif.com/split Split en .GIF fil op i enkelte frames