Talsystemer

Dette er et dokument om forskellige talsystemer. Løst og fast !!

Romertal

Først lidt om Romertal. Hvordan var de struktureret??

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

1'ere:	I = 1	II = 2	III = 3	IV = 4	V = 5	VI = 6	VII = 7	VIII = 8	IX = 9
10'ere:	X = 10	XX = 20	XXX = 30	XL = 40	L = 50	LX = 60	LXX = 70	LXXX = 80	XC = 90
100'ere:	C = 100	CC = 200	CCC = 300	CD = 400	D = 500	DC = 600	DCC = 700	DCCC = 800	CM = 900
1000'ere:	M = 1000	MM = 2000	MMM = 3000						

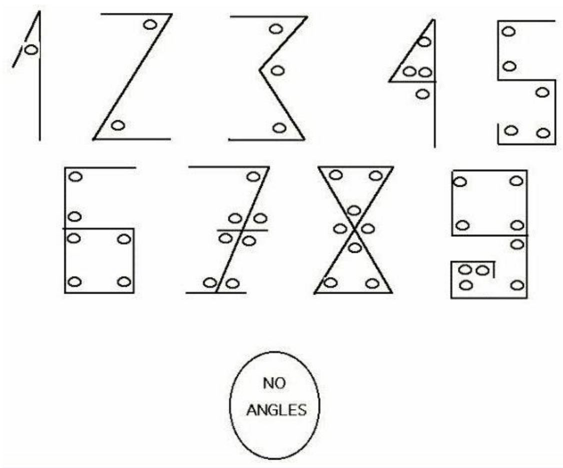
Regler:

Hvis et lille tal skrives foran et stort tal trækkes tallet fra: IV = 5 - 1 = 4
Hvis et lille tal skrives efter et stort tal lægges tallet til: VI = 5 + 1 = 6
Der må kun stå ét lille tal foran et større tal.

Hvordan er tallene skruet sammen?

Figuren viser de oprindelige tegn, - der havde et antal vinkler, svarende til deres værdi.

Arabisk for "nul" hedder "Ziffr"



http://message.snopes.com/showthread.php?t=49183

De danske talord [1]

[1] http://www.gorps.dk/artikler/talord.asp

Hvad betyder ”halvanden”? Kan man også sige ”Halvtredie”??

Hvad betyder egentlig ”halvfjerds” ?? Giver det mening, at det skal forstås som 3 plus det halve af den fjerde snes ?? Hvad med ”halvfems” ?? Er der andre forklaringer ??

Det næste er fundet på Nettet:
De danske talord er lidt specielle i forhold til andre sprogs talord. For det første nævner vi enerne før tierne (sådan er det dog også på tysk), og for det andet har vores tiere nogle – tilsyneladende – ulogiske navne. Der er dog mening i dem, hvis man kigger efter.

Talordene for 1½, 2½, 3½ osv.

Først skal vi lige kigge på ordet halvanden, som alle ved betyder 1½. Man kan sammenligne dette med vores måde at angive klokkeslet på, f.eks. halv to for kl. 1.30, halv tre for kl. 2.30 osv.

På helt tilsvarende måde kunne man tidligere sige halv*tredje* for tallet 2½, halv*fjerde* for tallet 3½, halv*femte* for tallet 4½ osv. Af en eller anden grund bruger vi ikke disse talord mere, men de har fået varig indflydelse på talordene for nogle af tierne (*halvtreds*, *halvfjerds* og *halvfems*).

De danske tier-talord

tal	dagligdags ord	det uforkortede tal-ord	forklaring
10	ti	ti	af indoeuropæisk *texun*, der betyder 'to hænder'
20	tyve	tyve	af urnordisk *twai teyjuz*, som betyder 'to tiere' (altså 2 gange 10 = 20)
30	tredive	tredive	af fællesgermansk *priz teyjuz*, som betyder 'tre tiere' (altså 3 gange 10 = 30)
40	fyrre	fyrretyve	af fællesgermansk *fedurai tey-juz*, som betyder 'fire tiere' (altså 4 gange 10 = 40)
50	halvtreds	halvtredsindstyve	halvtredje sinde tyve (altså 2½ gange 20 = 50)
60	tres	tresindstyve	tre sinde tyve (altså 3 gange 20 = 60)
70	halvfjerds	halvfjerdsindstyve	halvfjerde sinde tyve (altså 3½ gange 20 = 70)
80	firs	firsindstyve	fire sinde tyve (altså 4 gange 20 = 80)
90	halvfems	halvfemsindstyve	halvfemte sinde tyve (altså 4½ gange 20 = 90)

(*Sinde* betyder 'gange'. Vi kender det fra ordene *ingensinde* og *nogensinde*.)

Som om det kun var for at forvirre, er der altså to helt forskellige betydninger af ordet *tyve*: Det kan som i *fyrretyve* betyde 10, og så kan det som i *halvtredsindstyve*, *halvfjerdsindstyve*, *halvfemsindstyve* og i det dagligdags *tyve* betyde 20.

Bemærk, at den "mærkelige" forskel mellem *halvtreds* og *tres*, at der er d i det ene ord, men ikke i det andet, ikke er så mystisk, når man kender de uforkortede navne for tallene.

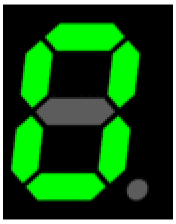
Hvorfor har vi mon 10-talsystemet ??

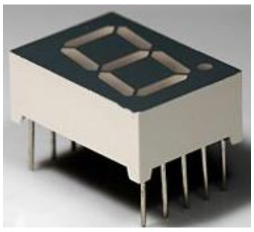
Digitale systemer:

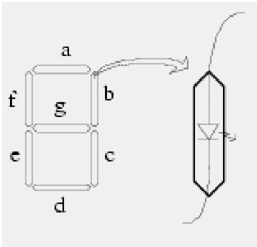
Tællere brugt i digitalteknik tæller normalt i det binære talsystem. Der findes dog nogle, fx 4017, der har 10 udgange, der skiftes til at være høj for hver tilført klock-puls.

Tællere der tæller efter det binære system kan enten tælle til 9 (decimal), og så starte forfra, eller til 15 (decimal).

De, der stopper ved 9, kaldes BCD-tællere, det står for Binary Coded Decimal. BCD-tællerne er ideelle til at sætte foran 7-segmenter, der jo netop kan vise tallene fra 0 og op til 9. Eksempler er 4029, 4518/20 og 40110.







Men hvordan virker så det binære talsystem?? Først ses lidt på vores kendte ti-talsystem:

Titalsystemet:

I det talsystem, vi er vokset op med, det decimale tal-system, har pladserne en bestemt værdi.

Fx tallet 923_D hvor D står for decimal, vides, at der er 9 hundreder, 2 tier og 3 enere. (Man kunne også skrive 923_{10}).

Tallet er sat sammen af 900 + 20 + 3 = 923.

Det kan også skrives som $9 \cdot 100 + 2 \cdot 10 + 3 \cdot 1$

Eller hvis der bruges potenser: $9 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

Pladsernes værdi svarer til potenser af ti, altså $10^n, \cdots, 10^2, 10^1, 10^0$

Man siger, at ti-talsystemet har grundtallet 10. Eller radix 10.

Det højeste tegn i ti-talsystemet er (10 minus 1), altså 9. Der er ti muligheder på hver plads, 0 til 9.

Tilsvarende regler findes for det binære talsystem:

Det største tegn på en plads er grundtallet minus 1

Det binære talsystem

Se evt: http://www.mathsisfun.com/binary-number-system.html

I det binære talsystem gælder de samme regler som i titalsystemet. Grundtallet er 2, og det største tegn er 2 minus 1 = 1. Der er altså to muligheder, 0 og 1.

Pladsernes værdi findes derfor som: $2^n, .., 2^3, 2^2, 2^1, 2^0$.

Pladsens værdi fra højre er også her grundtallet^0, grundtallet^1, grundtallet^2 osv

Et tal som fx 10011_B kan omregnes til titalsystemet.

$$1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 2 + 1 = 19$$

Omregning fra 10-talsystemet til binær:

Haves fx et tal på 58_d der ønskes omregnet til binær kan det gøres på flere måder: Først den logiske måde:

Der afsættes først et antal pladser til cifre i det binære tal. De er her angivet med et "X", og pladsernes værdi er angivet nedenunder.

Pladser	x	x	x	x	x	x	x
Pladsernes værdi	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Pladsernes værdi	64	32	16	8	4	2	1

Sættes et 1-tal på pladsen med værdien 64, har man brugt 64 af de 58, der er tilrådighed. Og det kan man jo ikke. Derfor et 0 på pladsen.

Der er råd til et "1"-tal på pladsen med værdien 32. Herved er der brugt 32 af de 58.
Resten er 26. Næste plads har værdien 16. Det er der også råd til, altså et "1"-tal på denne plads.
Resten er nu 26 – 16 = 10.

Næste plads har værdien 8. Denne kan også give et ”1”-tal, og resten er nu 10 – 8 = 2.

På pladsen med værdien 4, er der ikke råd. Derfor et 0 her. På værdi 2- pladsen er der netop råd til et ”1”-tal. Resten er nu 0, og derfor et 0 på sidste plads.

Skemaet ser nu således ud:

Pladser	x	x	x	x	x	x	x
Systematisk Værdi	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Pladsernes værdi	64	32	16	8	4	2	1
Der er ”råd til”	0	1	1	1	0	1	0

Altså tallet 58d kan omregnes til 111010b.

Det antal bit, der skal bruges, kan findes af følgende:

$$\text{Antal Bit} = \frac{\log_{10}(\text{Decimaltal})}{\log_{10}(2)}$$

Det fundne tal skal rundes op !!

Divisionsmetoden:

	Decimaltal	Rest	
	58		
Divideret med 2 =	29	0	LSB
”	14	1	
”	7	0	
”	3	1	
”	1	1	
”	0	1	MSB

Det fundne binære tal læses nedefra. Altså 111010b

På tilsvarende måde er det muligt at omregne fra binær til decimal:

Binær tal	1	0	0	1	1	0	1_2		
Værdi	64	32	16	8	4	2	1		
Ialt	64	+ 0	+ 0	+ 8	+ 4	+ 0	+ 1	=	77_{10}

Eller på multiplikationsmåden:

MSB						LSB
1	0	0	1	1	0	1
	+					
1 x 2 =	2	+				
	2 x 2 =	4	+			
		4 x 2 =	8	+		
			9 x 2 =	18	+	
				19 x 2 =	38	
					38 x 2 =	76

77

Der startes med mest betydende bit, MSB.

Det første 1-tal ganges med 2. Resultatet adderes med næste bit, og ganges igen med 2 osv.

<u>Hexadecimale talsystem, eller 16-talsystemet.</u>

Binary				HEX
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**2**
0	0	1	1	**3**
0	1	0	0	**4**
0	1	0	1	**5**
0	1	1	0	**6**
0	1	1	1	**7**
1	0	0	0	**8**
1	0	0	1	**9**
1	0	1	0	**A**
1	0	1	1	**B**
1	1	0	0	**C**
1	1	0	1	**D**
1	1	1	0	**E**
1	1	1	1	**F**

Why is hex important?

<u>One hex digit</u> can be used as <u>*shorthand*</u> to represent <u>four binary digits</u>

<u>Two hex digits</u> can be used as <u>*shorthand*</u> to represent <u>eight binary digits</u> or one byte

Også 16-talsystemet er opbygget som de andre talsystemer. Grundtallet er 16, og det største tegn har værdien 15.

Imidlertid er der ikke nogle naturlige tegn for værdierne 10 op til 15. Her bruges i stedet A, B, C, D, E og F

Binær tal				Decimal tal	Hex tal
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

Som i det binære talsystem kan omregnes på en ”logisk” måde

58.277$_d$ ønskes omsat til Hex

Først kan beregnes, hvor mange pladser, der skal bruges: $N = \frac{\log_{10}(58277)}{\log_{10}(16)}$ = 3,95 ~ 4 pladser
Disse afsættes, og deres værdier anføres:

Pladser	x	x	x	x
	16^3	16^2	16^1	16^0
Pladsernes værdi	4096	256	16	1
Der er ”råd til”				

Først undersøges, hvor mange der er råd til på 16^3 – pladsen.

58277 divideret med 4096 er 14 rest 933

Når de 14 gange 4096 er brugt, er der 933 tilbage, og af dem kan der placeres :

933 divideret med 256 er 3, resten er 165 - og så fremdeles:

165 delt med 16 er 10, og resten er 5.

Det ser på skemaform således ud !!

Pladser	x	x	x	x
	16^3	16^2	16^1	16^0
Pladsernes værdi	4096	256	16	1
Der er ”råd til”	14 = E	3	10 = A	5

Altså er 58277$_D$ lig med E3A5h

På divisionsmåden fås:

	Decimaltal	Rest	
	58.277		
Divideret med 16 =	3642	5	LSB
”	227	10=A	
”	14	3	
”	0	14=E	MSB

Det fundne Hexadecimale tal læses nedefra. Altså 58.277$_d$ = E3A5h.

Fra binær til Hex og omvendt:

Et binært tal kan opdeles i grupper af 4 bit fra højre. Hver gruppe kan udtrykkes med et hex-tal.

Eks: 1100 1001 0011b. konverteres til
C 9 3 Hex.

Fra Hex til binær bruges samme metode. Hver Hex-ciffer skrives med 4 bit.

Eks. $39A_{hex}$ bliver til 0011 1001 1010 binær

Forskellige andre koder.

Eksempler på forskellige binære koder til forskellige formål. Flg. Er nogle eksempler:

DEC	8421	2421	5421	XS-3	10-Gray
0	0000	0000	0000	0011	0000
1	0001	0001	0001	0100	0001
2	0010	0010	0010	0101	0011
3	0011	0011	0011	0110	0010
4	0100	0100	0100	0111	0110
5	0101	1011	1000	1000	1110
6	0110	1100	1001	1001	1010
7	0111	1101	1010	1010	1011
8	1000	1110	1011	1011	1001
9	1001	1111	1100	1100	1000

Gray-kode:

Der findes forskellige koder til forskellige formål.

Gray-koden er speciel, idet den er opbygget således, at der fra ét trin til det næste kun er ét bit, der ændres.

Opbygningen ses til højre herfor.

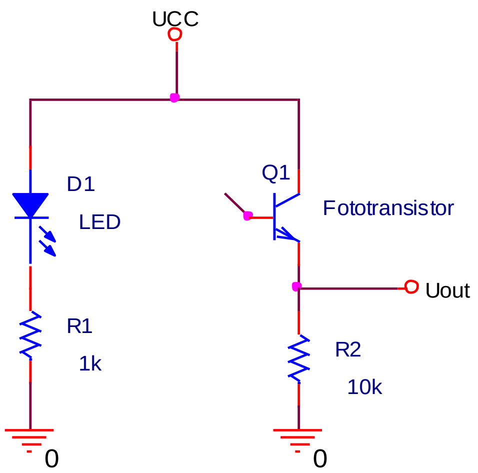
				0
Der spejles				1
Og sættes et 1 foran			1	1
Der spejles igen			1	0
Og sættes et 1 foran		1	1	0
		1	1	1
		1	0	1
Og igen		1	0	0
	1	1	0	0
	1	1	0	1
	1	1	1	1
	1	1	1	0
	1	0	1	0
	1	0	1	1
	1	0	0	1
	1	0	0	0

Gray-koden er også den kode, der bruges i Karnaugh-kort !!

Lad os antage, der skal bygges en vindretningsviser. For at opnå så lille friktion som muligt, laves en optisk løsning. På vejrhanen monteres en kodeskive. Over skiven placeres infrarøde lysdioder, og under skiven nogle fototransistorer.

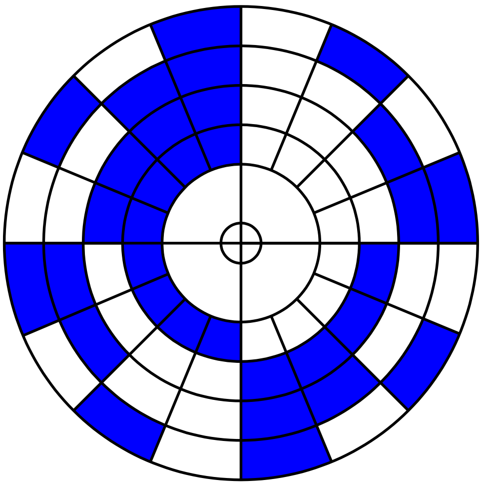
Flg. diagram-eksempel kan bruges.

Der skal bruges 4 fototransistorer til 4 bit.



Først ses på en 4 bit binær kodehjul.

Man kan forestille sig, at der er huller i skiven, hvor den er blå. Dvs. når skiven er drejet således, at lyset kan gå gennem et hul, leder fototransistoren under, og der kommer en kode ud. Koden for de forskellige positioner ses til højre for hjulet.



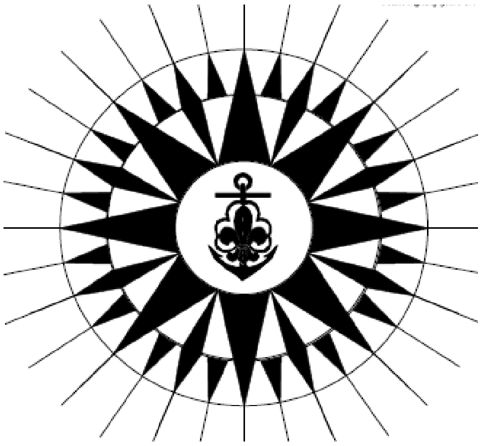
Kode fra kodehjul .

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Mest betydende bit er inderst.

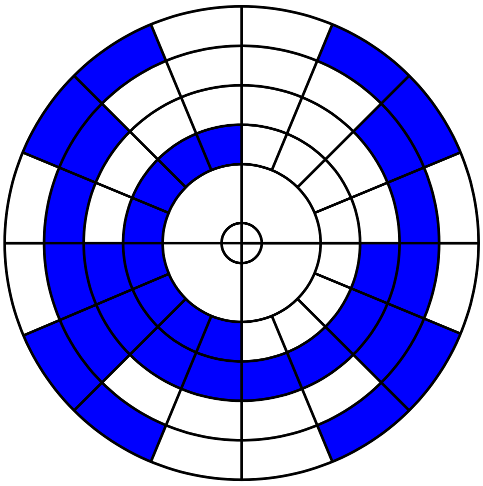
Under ideelle forhold virker systemet godt nok. Men man kan ikke med sikkerhed sige, at når skiven drejer, og der skal ske et skift fra fx 0111 til 1000 at det vil ske samtidig. Lysdioder eller transistorer kan sidde lidt skævt, så der sker et skift fra 0111 til 1111 og til 1000. Derved vil elektronikken, der omsætter koden til en vindretningsvisning på et display kortvarigt vise forkert.

Displayet kunne fx bestå af 16 lysdioder placeret som på en kompasrose.



Med GRAY-kodehjulet er der imidlertid kun 1 skift hver gang.

Gray-kodehjulet.



Mestbetydende bit inderst.

	D	C	B	A
MSB-LSB	0	0	0	0
	0	0	0	1
	0	0	1	1
	0	0	1	0
	0	1	1	0
	0	1	1	1
	0	1	0	1
	0	1	0	0
	1	1	0	0
	1	1	0	1
	1	1	1	1
	1	1	1	0
	1	0	1	0
	1	0	1	1
	1	0	0	1
	1	0	0	0

Elektronikken skal nu blot håndtere signalerne på en anden måde, så vindretningsudlæsningen bliver korrekt. Det kan fx gøres med en 4 til 16 demultiplexer, (eller demuxer).

10-gray

Gray-koden findes også i en 10-gray udgave:

10-gray

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0

0	1	1	0
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

OBS: !!! Der findes mange andre koder, hvor pladserne har andre værdier !! (Vægtede koder)

ASCII koden

ASCII koden. Står for American Standard Code for Information Interchange.

I ASCII-standarden er defineret en 7 bit kode for alle karakterer, bogstaver og tal, der blev brugt tidligere til at sende over til fx en matrix-printer. Koden bruges i dag til diverse LCD karakter-display. Sendes fx et 30h vil et LCD skrive et '0'-tal. Søg på nettet efter ASCII. Følgende skema er gaflet:

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

Extended ASCII Code

Den første ASCII-kode havde kun 7 bit. Med 8 bit kan der være dobbelt så mange karakterer. Tjek på nettet:

Bonus:

(Ikke bearbejdet endnu !!)

Addition af binære tal

Mangler

Se fx: http://www.wikihow.com/Add-Binary-Numbers
http://www.math.grin.edu/~rebelsky/Courses/152/97F/Readings/student-binary
http://www.quickanddirtytips.com/education/math/how-to-add-binary-numbers
Youtube: http://www.youtube.com/watch?v=jB_sRh5yoZk

1's komplement
Mangler

2's komplement

2's complement er den mest populære måde at angive tal med fortegn i computere.

Bruges fx hvor en microcontroller skal læse en temperatur fra en I2C-temperaturkreds. !

Anvendes det binære talsystem til at angive både positive tal, og negative tal, kan der med 8 bit angives tal fra +127 til -128. Bit 7 bruges til at angive fortegn. Er den 0, er tallet repræsenteret af de næste 7 bit på almindelig binær form.

Er MSB derimod, altså Most Significant Bit, eller "Sign bit" = 1, er hele tallet negativt. De resterende 7 bit er bare ikke angivet på den almindelige binære kode-form, men angivet med 2's complement.

Eks:

For at omregne et negativt 2's complement 8-bit tal til et positivt tal, inverteres alle 8 bit og der adderes 1.

Eller, fx -5 på sign'ed 8-bit form er 1111 1011. Det kan omregnes således!

-128 + 64 + 32 + 16 + 8 + 0 + 2 + 1 = -5.

2's complement er egentlig beregnet til processorer, der ikke kan subtrahere. Ved at angive et tal, der skal trækkes fra et andet med 2's complement, kan en addition føre til det korrekte resultat.

sign bit									
0	1	1	1	1	1	1	1	=	127
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	−1
1	1	1	1	1	1	1	0	=	−2
1	0	0	0	0	0	0	1	=	−127
1	0	0	0	0	0	0	0	=	−128

8-bit two's complement integers

Fx 3 – 1, er (her kun med 4 bit) 0011 + 1111 = 1 0010. Det 5. bit, overflow-bittet bortkastes, og det korrekte resultat haves.

For at konvertere et tal til 2's complement, inverteres alle bit, og der adderes 1.

Eks: 0000 0101 (= 5) inverteres til 1111 1010. Nu haves 1's complement, og efter addition af 1, fås 1111 1011 (= -1)

Skal -5 = 1111 1011 konverteres til det korresponderende positive tal, negeres alle bit, til 0000 0100, og efter addition af 1 fås 0000 0101 = 5.

Integer		**2's Complement**
Signed	**Unsigned**	
5	5	0000 0101
4	4	0000 0100
3	3	0000 0011
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

Eksempel:

```
7 - 12 = (-5)     0000 0111   = +7
                + 1111 0100 = -12
                ___________
                  1111 1011   = -5
```