Computersystemer 1

Lavet af: Vivek Misra

De syv idéer

- **Algoritmer:** Undersøge, forske osv.
- **Abstraktion:** Hvordan kan gøre tingene bedre fra det dårlige.
- **Kreativitet:** Fordi vi skal programmere, så kan man finde en algoritme som står kort og præcist efter formål.
- **Data:** Det går ud på, at behandle data. Det handler om, at samle alle mulige data, hvor der skal indgå mere flow. Data er det som driver det hele. Data indeholder noget information.
- **Programmering:** Det er vores håndværk, hvor vi laver beregninger.
- **Internet:** Vi skal arbejde med distribuerede systemer, hvor computerne er koblet sammen.
- **Indflydelse:** Hvordan hjælper det vores samfund, eksempelvis hvordan kan vores programmering hjælpe sygeplejerske på sygehuset med at kigge på patientens data.

Bits

- Data er meget vigtigt i vores profession.
- Indeholder information, som er kaldt for bits.
- Bits kan fortælle om lyset er slukket eller åbent
- Man skal hele tiden gemme noget bit, hvor man nogle gruppe informationer.
- Man kan repræsentere numre, karakterer, bogstaver.
- Hvis man har flere bits/information, så kan man repræsentere billeder, lyd og andre ting.
- Grupper af 8 bits, er samlet til 1 byte.

Georges Aksiom 1

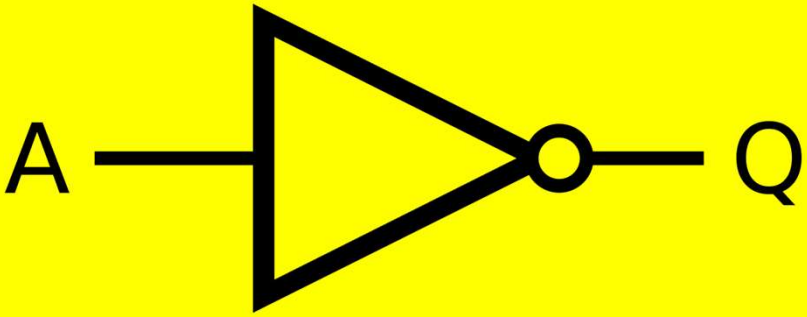
- En variabel X, som kan antage en og kun en af de to værdier.
- Hvis x er svarende til 1, så er $x \neq 0$, $x = 0$, hvis $x = 1$.

Georges Aksiom 2 (NOT)

- ”Ikke Funktionen. Negationen (NOT).
- $F = \bar{X}$

- Det fortæller, at f bliver det modsatte af X.
- Dette fortæller, at hvis man har en input på 0, får man output 1.
- Hvorimod et input på 0, giver et output på 1.
- Trekant med en prik, er symbolet på invers.
  - Dette betyder, at tallet som når dertil, vendes om.

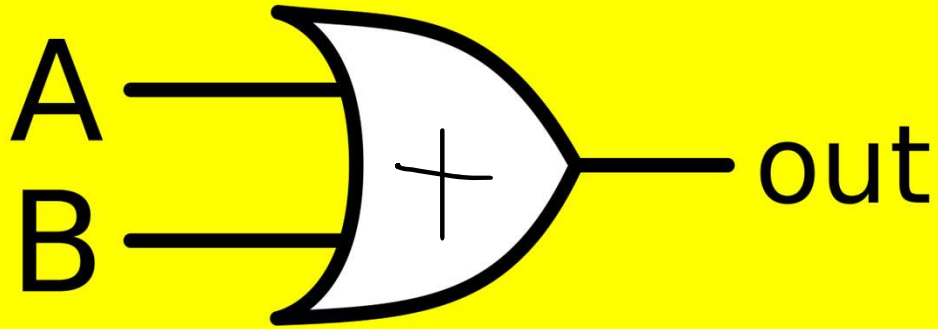
X	F
0	1
1	0



Georges Aksiom 3 (OR)

- Her kan man se, at både x skal være 1 og y skal være 1, så bliver F=1.
- Dette betyder, at bare en af inputtene bliver 1, kan de andre også være 1.
- Hvis en af tingene er sand, så kan det godt være at det sidste vil være sandt.
- Man har to bit, som giver et afbud på F. Vi har en parabolsk trekant.
- $F = X + Y$

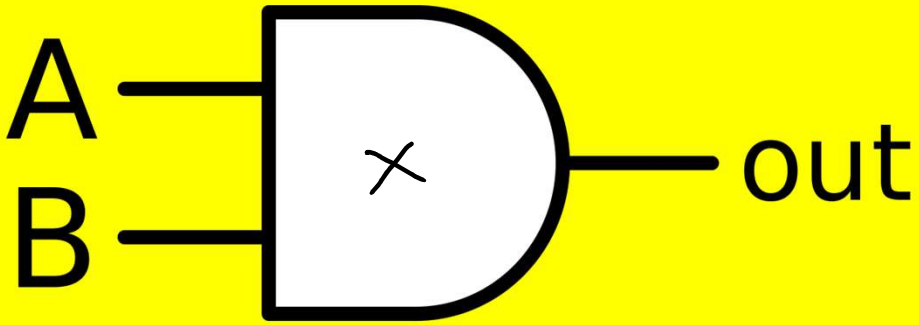
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1



Georges Aksiom 4 (AND)

- En variabel X, som kan antage en og kun en af de to værdier.
- Hvis x er svarende til1, så er $x \neq 0$, $x = 0$, hvis $x = 1$.
  - Her har man det, at både x skal være 1 og y skal være 1, så bliver F til 1.
- $F = X * Y$
- Man kalder det for en ”og” funktion, altså en ”AND-funktion”.

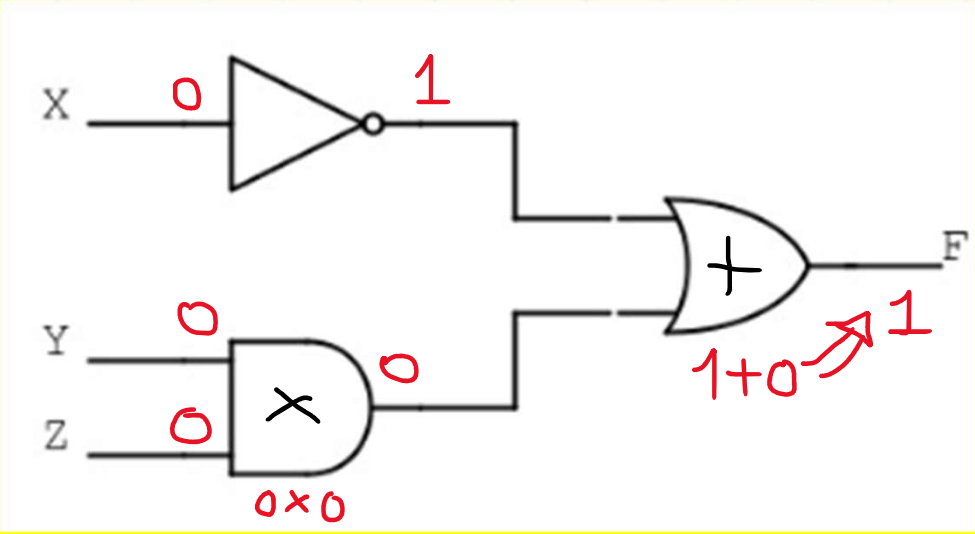
X=A	Y=B	F
0	0	0
0	1	0
1	0	0
1	1	1



Et logisk kredsløb

- Hvis alle 3 er 1, så er F svarende til 1.
- Man har en tabelmåde at skrive det på.
- Man har en matematisk måde at skrive det på.
- $F = X + Y * Z$

X	Y	Z	F
0	0	0	1
0	1	1	1
1	0	0	1
1	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Main Memory

- Man giver plads i memory (hukommelse), som har plads til 8 bit.
- Man kan gå inde i adresse 47, hvor der eksempelvis står 64.
- **Mest signifikant bit:** En bit til venstre (højere-ordre).
- **Mindst signifikant bit:** En bit som er til højre (lavere-ordre).

Adresse

- Adresse: Et navn som egentligt identificerer en celle i et computers hukommelse.
  - Navnet er egentligt numre.
  - Disse numre er ment for at starte ved 0.
  - Numringen af celler i denne manér er associeret med en ordre med hukommelsesceller.

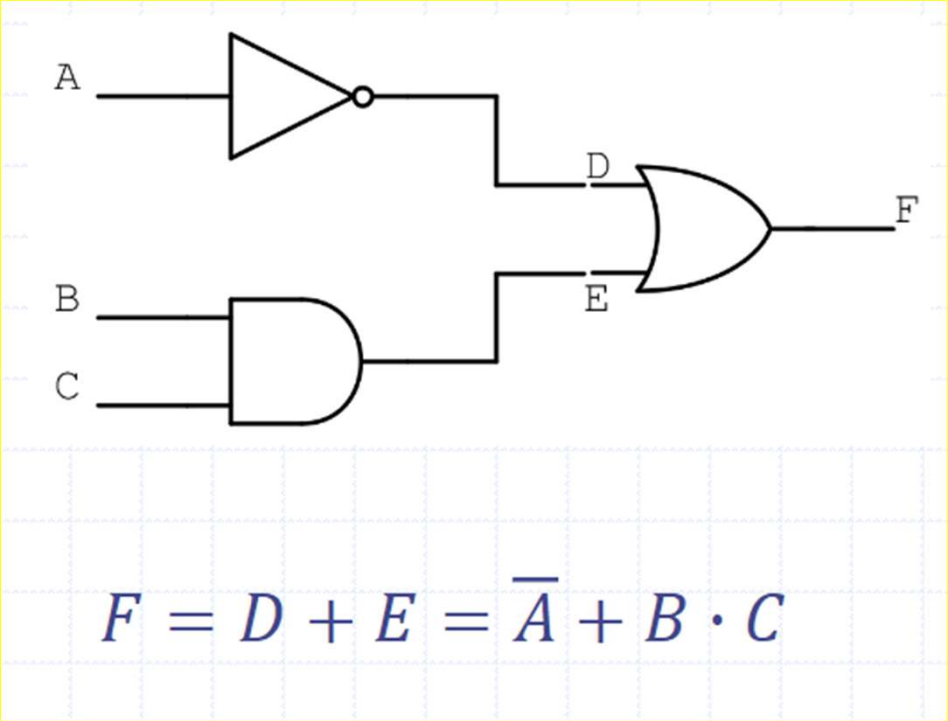
Kombinerede Funktioner

- Kig tydeligt på tabellen og sammenlign det med billedet på næste slide.

Input					F
A	B	C	D	E	F
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

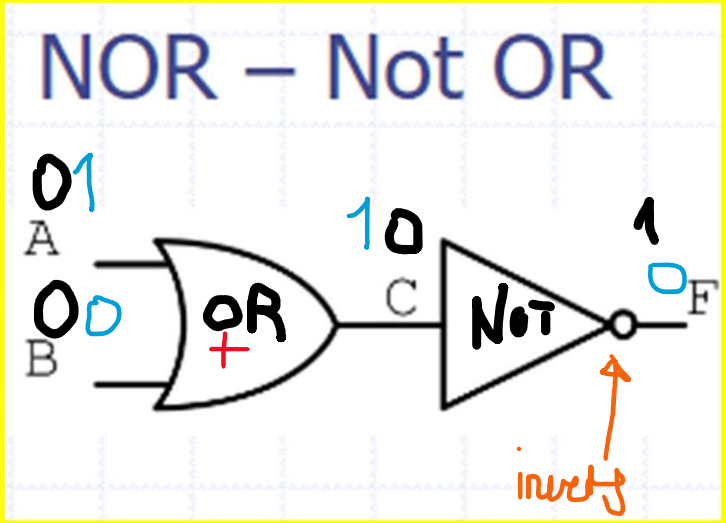
Kombinerede Funktioner

- Kig tydeligt på tabellen og sammenlignet med billedet på denne slide.



NOR og NAND

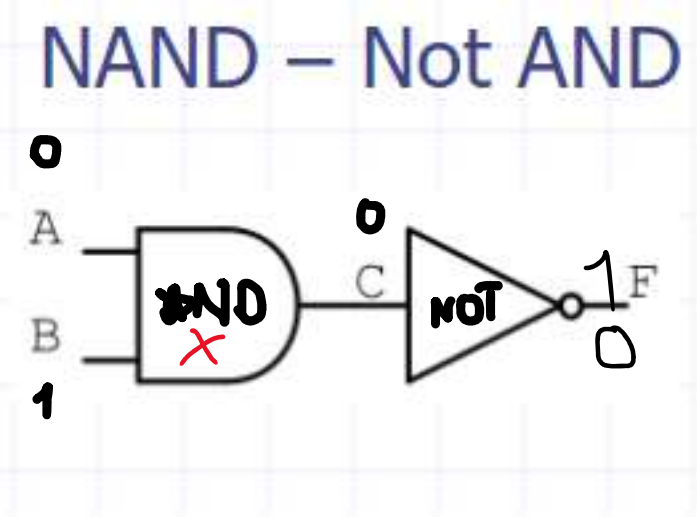
- Vi kan i denne sammenhæng se, følgende billeder som giver eksempel med kombinationer NOT og OR.



In			Out
A	B	C	F
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NOR og NAND

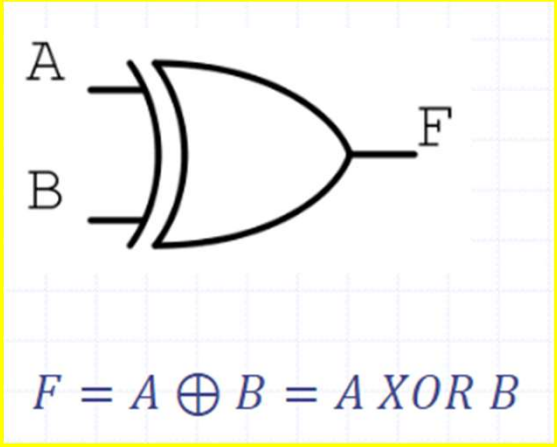
- Vi kan i denne sammenhæng se, følgende billeder som giver eksempel med kombinationer NOT og OR.



In			Out
A	B	C	F
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

XOR

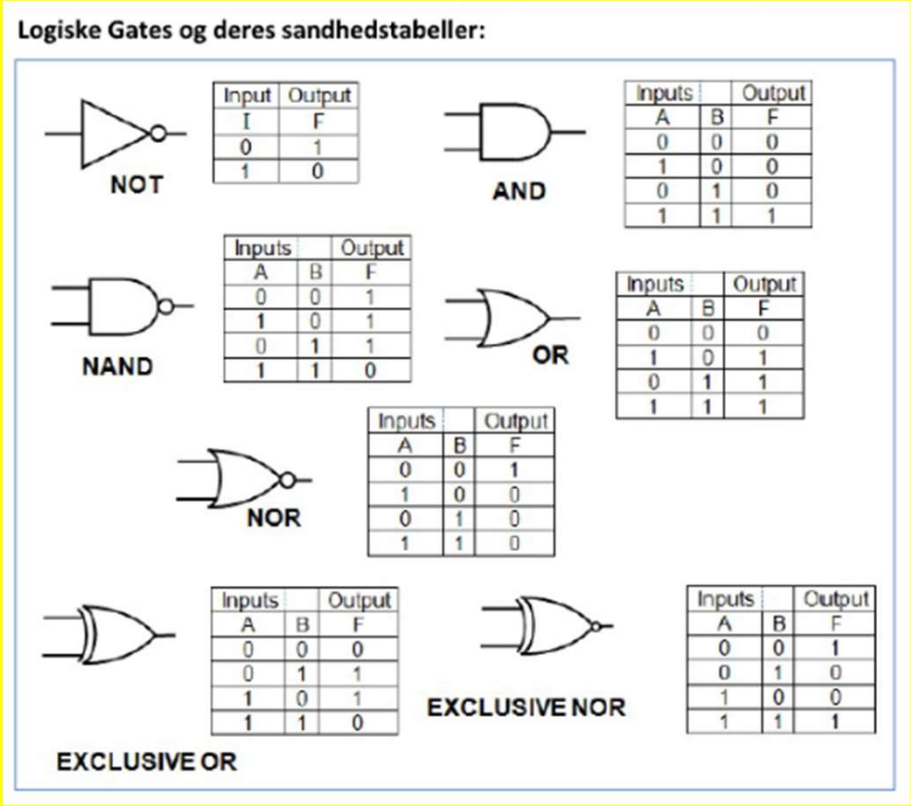
- OR: Det er den ene eller den anden, eller begge to.
- XOR: Den ene eller den anden men ikke begge to.
- XOR: De to input er forskellige.



A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

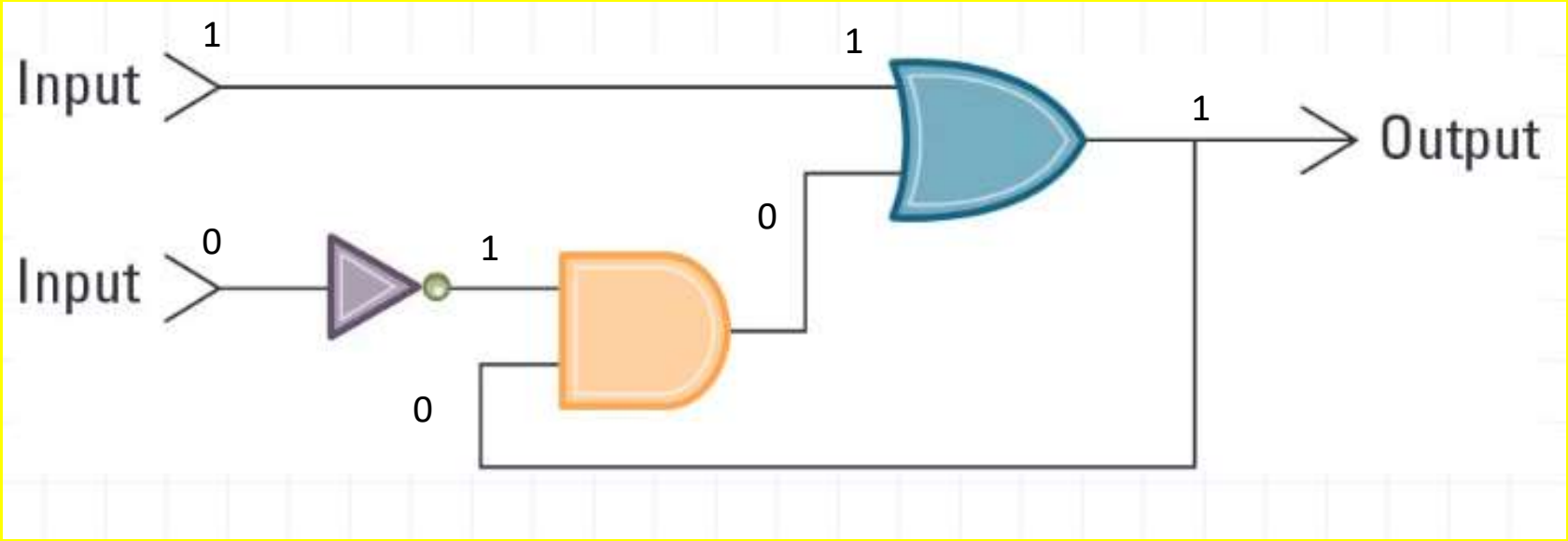
Logiske Gates

- Se på billedet, og få overblik over Gates.



Simpel Flip Flop

- Billedet viser, hvordan simple flop kører med kredsløb.



Flip Flops

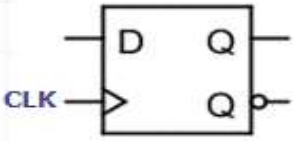
- Disse er 2 almindeligt forekommende flip-flops.
- Hvis man kigger på opbygning af computeren, så kan vi se at vi har en main memory. CPU'en i denne sammenhæng er opbygget af XOR, AND or OR. Man kan også have flip flops som kan have main resultater.

2 almindeligt forekommende flip-flop's

SR flip-flop:

Input		Output
S	R	Q
0	0	Last Q
0	1	0
1	0	1
1	1	Udefineret

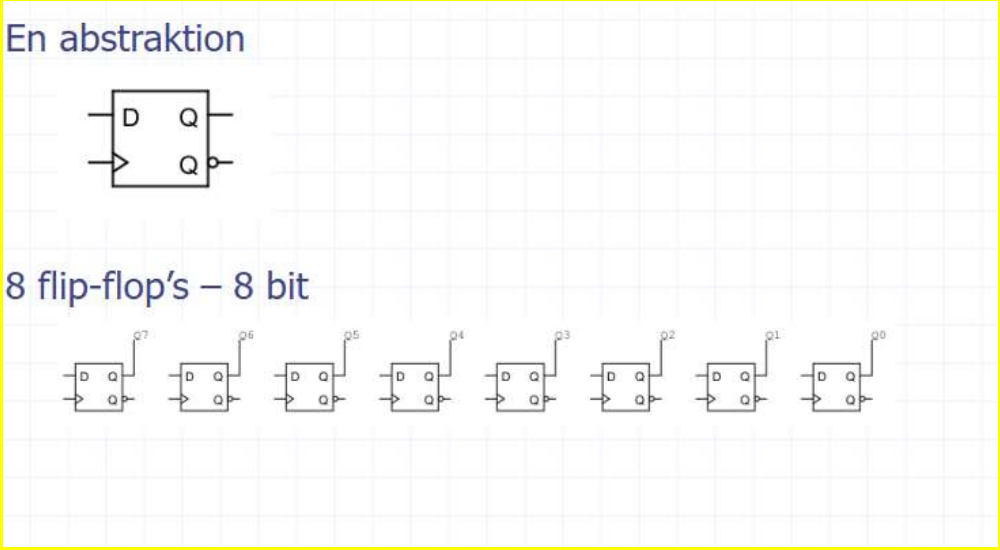
D flip-flop:



Input		Output
D	CLK	Q
X	0	Last Q
X	1	Last Q
0	↑	0
1	↑	1

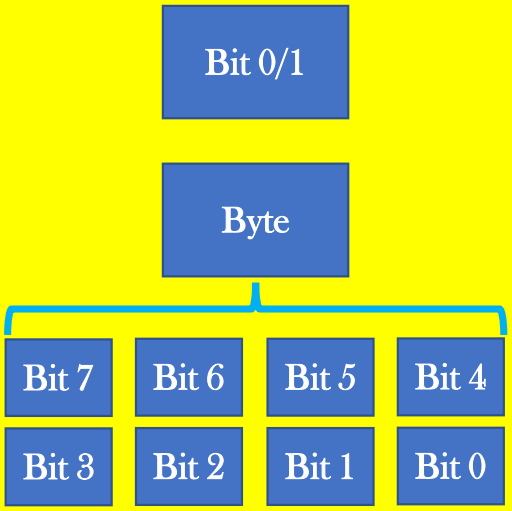
Bit and Bytes

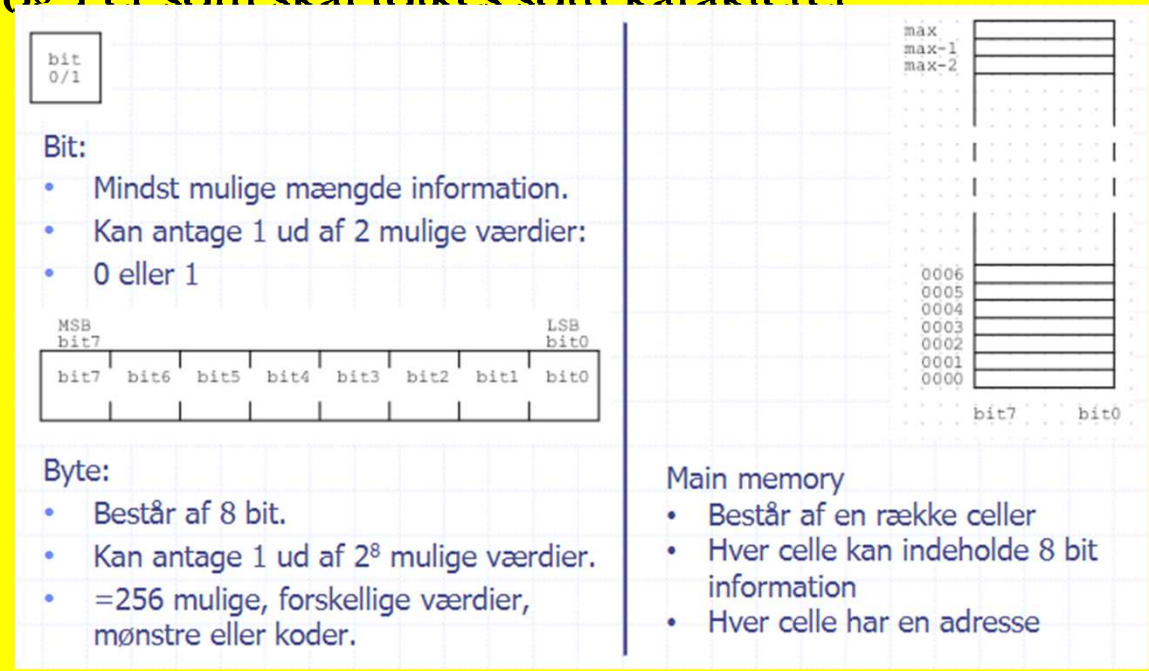
- Dette billede viser, hvordan en bit ser ud.



Bit and Bytes

- Dette billede viser, hvordan en bit ser ud.
- En bit og byte udgør lageret i en computer. På hver byte en memory kan der ligge 8 cifre, men vi kan ikke se hvad de betyder - derfor skal vi have en aftaler /adresse hvor vi siger at der skal ligges nogle 0'er og 1'er som skal tolkes som karakterer





Udregning og Mellemregning i Binære Verden!

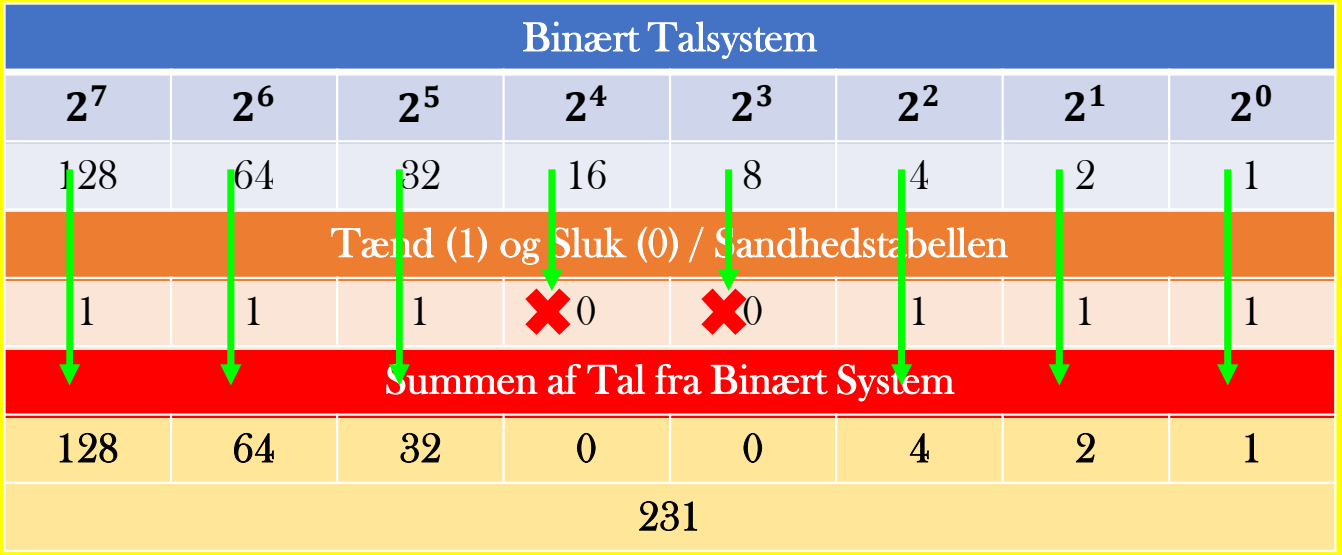
- Vi plejer, at kigge på tal med en normal måde for at undgå længere tælling.
- Base 10 systemet er kendt som decimalsystemet.
  - Det er 0,1,2,3,4,5,6,7,8,9
- Her kan vi se, at vi har eksempelvis tallet 231.

231	2	3	1
	2*100	3*10	1*1

- Her kan vi se, at ved den nederste lyseblå række at der sker en fordobling med 10.

Base 2-Systemet = Binært System

- Det består af 1 og 0.
- I stedet for at gange 10, implementerer vi konceptet med 2.



Base 16 Systemet = Hexadecimale System

- Base16-Systemet er kendt som det Hexadecimale System. I stedet for 10-tal på Rækken, skriver vi/fokuserer vi på 16-Talsystemet.

Hexadecimale Talsystem															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
										10	11	12	13	14	15

Eksempel 1 med Hexadecimale Talsystem

- Lad os tage en Eksempel med E7.
- I denne sammenhæng, kan vi se at E = 17 fra Hexadecimale Talsystem.
- Vi kan også se, at 7 er bare sig selv som et syvtal.

E7	
E	7
$14 * 16 = 224$	$7 * 1$
Summen af Tallene fra Udregningen	
$\mathbf{224 + 7 = 231}$	

Eksempel 2 med Hexadecimale Talsystem

- Vi tager en endnu en Eksempel, bare med AF3 (3 værdier).
- Vi kan se, at følgende værdier er svarende således:

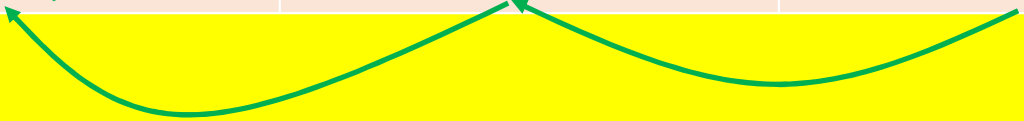
Værdi	Svarende i Hexadecimal Talsystem
A	10
F	15
3	3

E7		
A	F	3
$10 \cdot (16 \cdot 16) = 2560$	$15 * 16 = 240$	$3 * 1 = 3$
Summen af Tallene fra Udregningen		
$\mathbf{2560 + 240 + 3 = 2583}$		

Huskeregel og Oftest fejl

- Som du så i Eksemplet gange vi dobbelt med 16.

E7		
A	F	3
$10 \cdot (16 \cdot 16) = 2560$	$15 * 16 = 240$	$3 * 1 = 3$



$x16^2 = 256$ $x16^1 = 16$

Hvert gang et tal/bogstav kommer ydeligere på, fordobles værdien af 16 med opløftning i Eksponenten

Eksempel på Opgaver med Base2-Systemet

Opgave Nummer	Tal som skal konverteres	16	8	4	2	1	Hvad er Summen?	Endelige Binært Svar
A	1	X	0	0	0	1	1	0001
B	2	X	0	0	1	0	1	0010
C	3	X	0	0	1	1	2+1	0011
D	8	X	1	0	0	0	8	1000
E	9	X	1	0	0	1	8+1	1001
F	10	X	1	0	1	0	8+2	1010
G	15	X	1	1	1	1	8+4+2+1	1111
H	16	1	0	0	0	0	16	10000
I	17	1	0	0	0	1	16+1	10001
J	18	1	0	0	1	0	16+2	10010

Addition af Binære Tal

- Foregår på samme måde som vi gør siden Folkeskolen ☺
- Men teknik og tankegang er lidt anderledes med hensyn til overskud/carry forward.
- Vi starter med, at gennemgå nogle regler og derefter Opgaveeksempler.

Regler	Definitioner
1	Husk, altid at der som sagt at hvis 1 og 1, lægges sammen så får vi 0 med overskud i 1. $1 + 1 = 0$ $1\ i\ overskud$
2	Husk, altid hvis 0 lægges sammen med 0. Så får vi 0, men med ingen overskud! $0 + 0 = 0$ $X = overskud$

Eksempel med Additionsopgaver

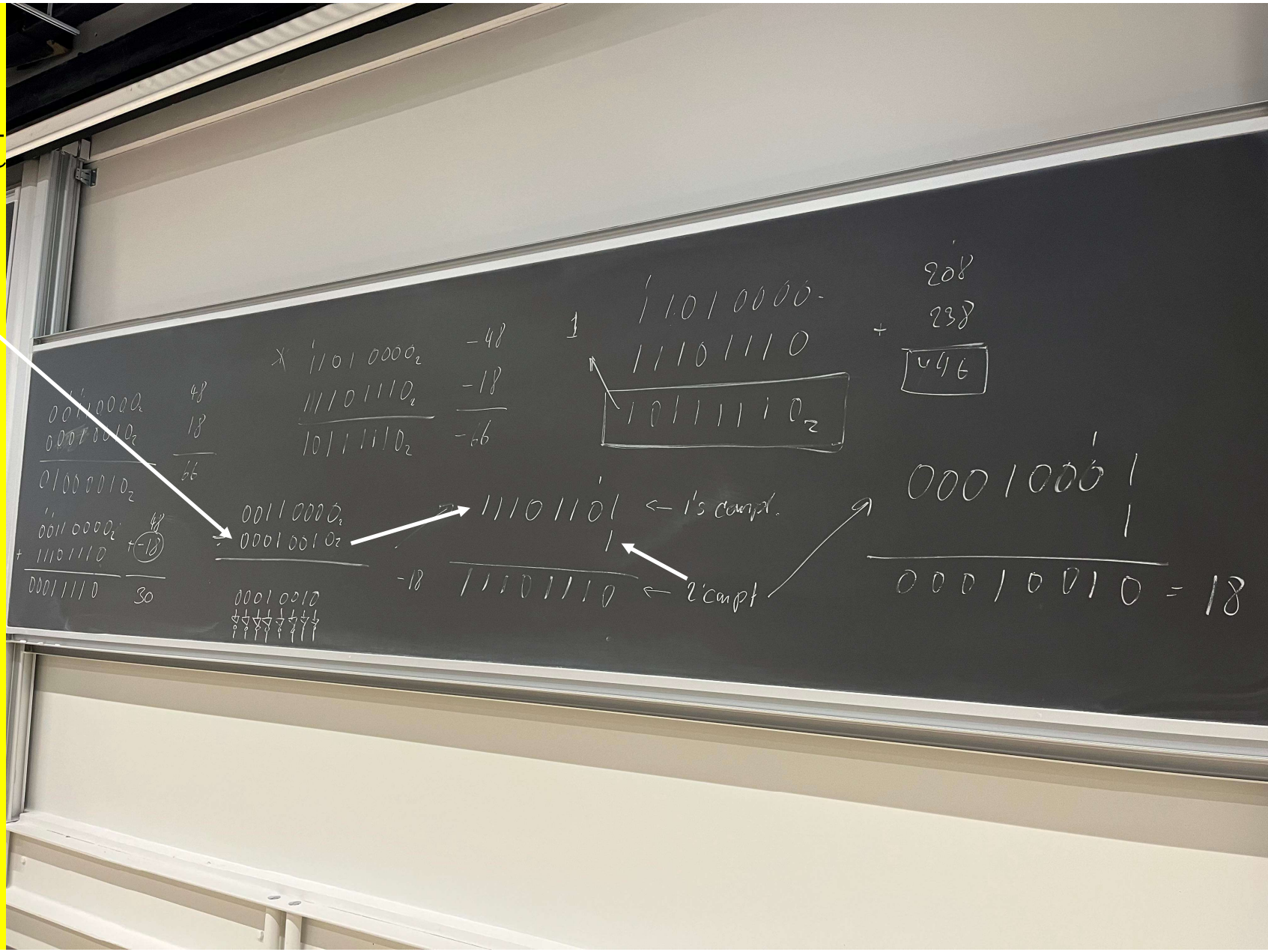
I denne sammenhæng, kan det ses at de røde 1'ere er overskuds tal som kommer efter sammenlægning af to 1'ere
På Eksempel 4, kan man se en lille 0 efter sammenlægning af 1 og 1. Dette bruges til at vise hvordan mellemregningen sker.

		1	1
01	10	11	11
10	110	01	111
----	----	----	----
11	100	110	110

1 1 1			
1001	1100	1000	10101
1 0111	0101	0011	01111
--------	--------	--------	--------
10000	10001	1011	1000100

Complement

- 1's complement, hvor tallet vendes om.
- 2's complement hvor tallet lægges sammen med 1.



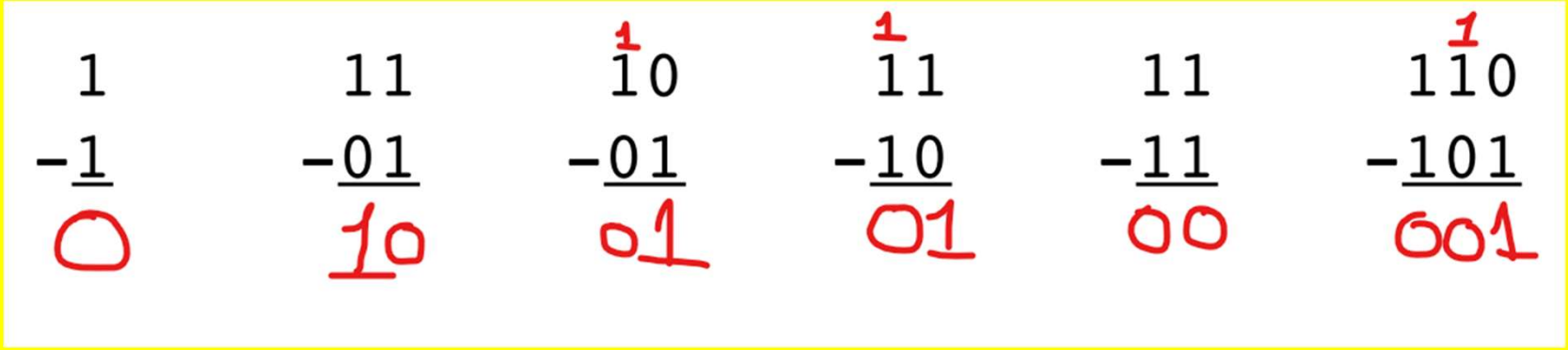
Subtraktion af Binære Tal

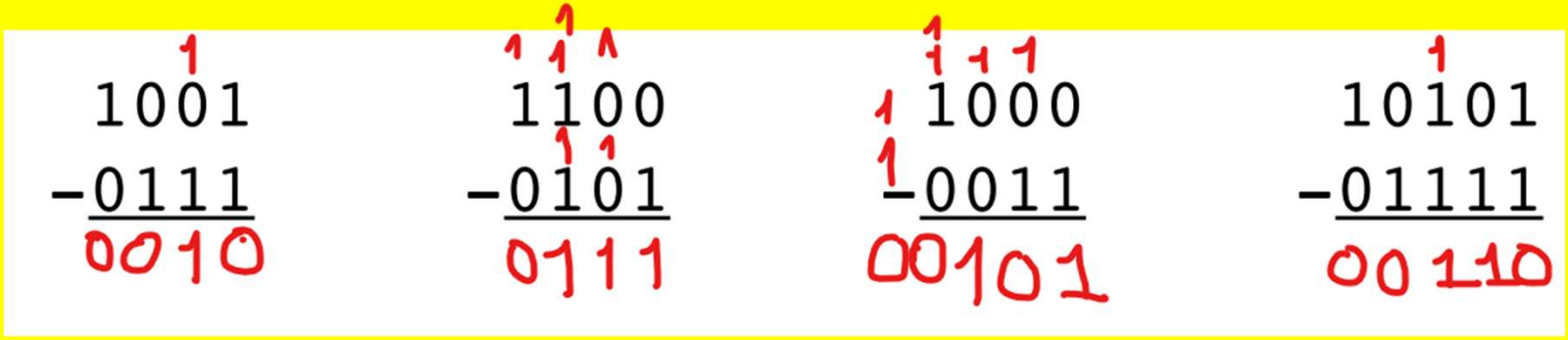
- Foregår på samme måde som vi gør siden Folkeskolen ☺
- Men teknik og tankegang er lidt anderledes med hensyn til overskud/carry forward.

Regler	Definitioner
1	Husk, altid at der som sagt at hvis 1 og 1, lægges sammen så får vi 0 med overskud i 1. $$1 - 1 = 0$$ $$X = overskud$$
2	Husk, altid hvis 1 trækkes fra 0. Så får vi 1, men med ingen overskud! $$0 - 1 = 1$$ $$1 = overskud$$
3	Husk, at hvis 0 trækkes fra 1. Så får vi 1, men med 1 i overskud. $$1 - 0 = 1$$ $$1 = overskud$$

Eksempel med Subtraktionsopgaver

Nu vises eksempler med regning med brug af reglerne fra sidste slide.





Større tal fra Decimal til Binære

- Vi skal i denne sammenhæng kende til følgende tabel.

2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

- I denne sammenhæng, kan vi tage eksempler på følgende opgaver for at skabe bedre eksempler.

Eksempler på Opgaver

- Starter med, at tage en tal som er 114.
- Nu anvendes tabellen fra sidste slide.

$$114 - 64$$
$$50 - 32$$
$$18 - 16$$
$$2 - 2$$

2^6	2^5	2^4	2^3	2^2	2^1	2^0
114-64=50	50-32=18	18-16=2	X	X	2-2=0	X
1	1	1	0	0	1	0

Eksempler på Opgaver

- Starter med, at tage en tal som er 114.
- Nu anvendes tabellen fra sidste slide.

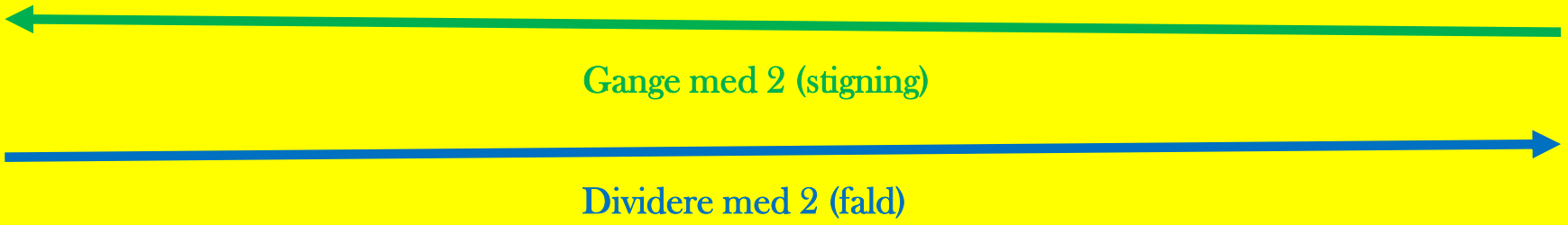
Talrække	2^6	2^5	2^4	2^3	2^2	2^1	2^0
39	X	39-32=7	X	X	7-4=3	3-2=1	1-1=0
Binært Talsvar	1	1	0	0	1	1	1

Talrække	2^6	2^5	2^4	2^3	2^2	2^1	2^0
27	X	X	27-16=11	11-8=3	X	3-2=1	1-1=0
Binært Talsvar	0	0	1	1	0	1	1

Conversion til og fra Floating Point

- Dette billede viser, hvordan en bit ser ud.

2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1



Fra Decimal til Floating Point

- Vi starter med, at tage et eksempel.

-7 ½

x yyy zzz

Det er selve de svar vi skal munde ud i.
Forvirre jer ikke selv, fordi syntaksen har andet betydning.

❖STEPS

1. Opskriv Talsystemstabellen og indsæt rigtige værdier. (Tænd og Sluk)

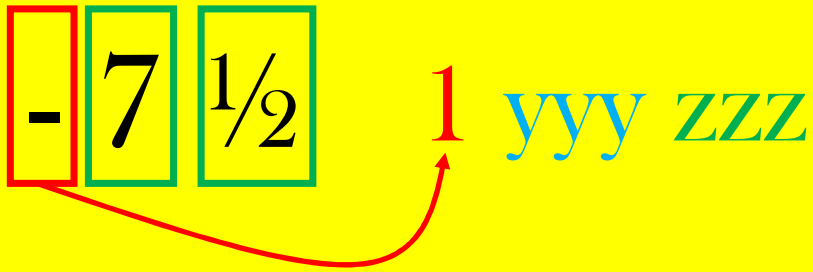
$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
0	1	1	1	1	0	0	0

4+2+1=7 og så har vi 1/2

Conversion til og fra Floating Point

2. Opskriv tal baseret på foretegn.

- Skriv 1 ved negativt fortegn.
- Skriv 0 ved positivt fortegn.



Conversion til og fra Floating Point

3. Find skridt.

- OBS: Regler.
  - Hvis 1-tallet står før Kommaet mellem 1 og 1/2 , skal kommaet (den usynlige) rykkes foran fra 1-Tallet og frem til Kommaet mellem 1 og ½.
  - Hvis 1-tallet står efter Kommaet mellem 1 og ½ rykkes kommaet mod højre indtil det (usynlige) ikke lander efter (4-tal med 1).

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
0	1	1	1	1	0	0	0

VI BRUGER REGEL NUMMER 1., I DENNE SAMMENHÆNG ☺

Conversion til og fra Floating Point

- Fordi vi er rykket mod Divisionsretningen, kan vi i denne sammenhæng finde ud af følgende.
- Vi ved, at det er 3 skridt mod højre.
  - Derfor skriver vi -3 iforhold til 3.bit.
  - Dette skrive følgende fra bogen på side 71:
  - -3 = (101)

-	7	½

1 101 zzz

Conversion til og fra Floating Point

- De sidste grønne z'ere er svarende til de fire 1'er fra tabellen.
  - Disse er også kaldt Mantissaen.

-	7	½	1 101 1111

5. Konkluder

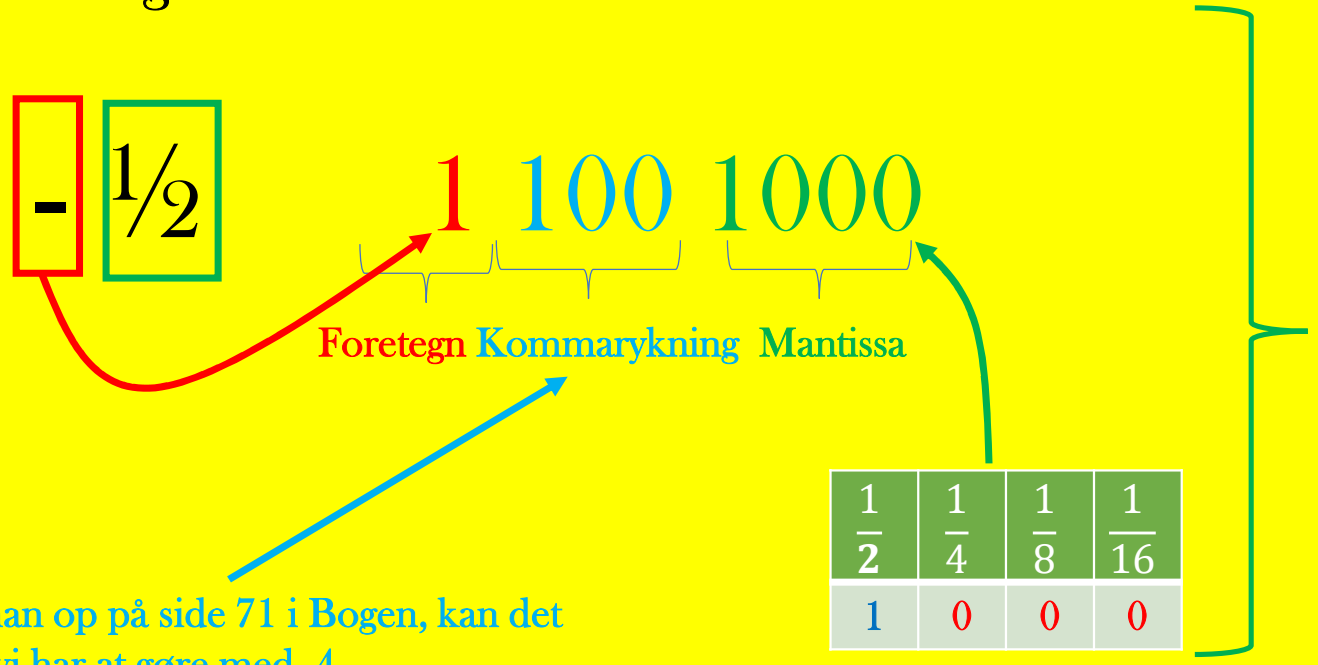
- Vi kan konkludere at vores -7 ½ bliver til 1 101 1111

Fra Floating Point til Decimal

- Nu skal vi til at tænke omvendt!
- Vi skal aflæse det fra en udefra-synspunkt som leder til det samlede svar i Decimal!
- Vi viser et eksempel på Næste Slide.

Fra Floating Point til Decimal (Eksempel)

- Vi tager eksempel med halv – ½ , hvor vi skal være obs på negativt foretegn.



Python ☺

- Ønsker du at kende mere til Python.
- Kan det anbefales, at tjekke til W3-Schools ☺
- Websiden giver et bedre overblik over, hvordan kodet er opbygget.

Computersystemer 1

Lavet af: Vivek Misra

Håber I kan lide det ☺