UNIVERSITEIT GENT

Informatica

Prof. dr. Seppe vanden Broucke

Hoofdstuk 1: Gegevensopslag

Gegevensopslag

Hoofdstuk	12de editie	13de editie
1. Gegevensopslag	Subhoofdstukken: alles, 1.8 kun je kort doornemen om de Python voorbeeldjes beter te snappen Kadertjes: “Analog versus Digital” en “Single Precision Floating Point” Appendix A: ASCII-tabel (niet vanbuiten leren maar kunnen gebruiken)	Idem

Inhoud

- Oplossing verschilmachine
- Bits en hoe ze worden opgeslagen
- Het werkgeheugen van de computer
- Massageheugen
- Gegevens voorstellen met bitpatronen
- Het binair talstelsel
- Het opslaan van gehele getallen
- Het opslaan van reële getallen
- Compressie
- Communicatiefouten

Oplossing verschilmachine

- Schrijf een algoritme voor het berekenen van het kwadraat van een willekeurig natuurlijk getal (> 1) volgens de methode van de “successieve verschillen”
- Gegeven: $0^2 = 0$ en $1^2 = 1$ en $2^2 = 4$ (deze laatste eigenlijk niet nodig)
- Gebruik enkel de operatoren + en -
- Het “tweede verschil” is steeds 2 (wiskundig bewezen)
  - $3 + 2 = 5$ en $5 + 4 = 9$
  - $5 + 2 = 7$ en $7 + 9 = 16$
  - Enzoverder tot het gewenste getal bereikt is

Oplossing verschilmachine

```
Invoer: K = natuurlijk getal > 1 waarvoor we het kwadraat willen berekenen

Uitvoer: K^2

Procedure:

Stap 1. (Begin met laatste en voorlaatste gegeven kwadraten:)
        Zet X_VL = 0, X_L = 1
        Zet X_KW_VL = 0, X_K_L = 1

Stap 2. Als X_L niet gelijk is aan K:
            Zet V_VORIGE = X_K_L - X_KW_VL
            Zet V_VOLGENDE = V_VORIGE + 2
            Zet X_VL = X_VL + 1
            Zet X_L = X_L + 1
            Zet X_KW_VL = X_K_L
            Zet X_K_L = X_K_L + V_VOLGENDE
            Ga terug naar stap 2
        Zoniet: ga naar stap 3

Stap 3. Het eindresultaat is gelijk aan X_K_L
```

Oplossing verschilmachine

```
X_VL    X_L     X_KW_VL     X_K_L
----    ---     -------     -----
0       1       0           1
1       2       1           4           + (4-1 + 2 = 5) = 9
2       3       4           9           + (9-4 + 2 = 7) = 16
3       4       9           16          enzoverder...
4       5       16          25
5       6       25          36
```

Oplossing verschilmachine

```
x_voorlaatste = 0
x_laatste = 1
x_kwadraat_voorlaatste = 0
x_kwadraat_laatste = 1

k = int(input("Van welk getal wil je het kwadraat (>1): "))

while k != x_laatste:
  verschil_vorige_kwadraten = x_kwadraat_laatste - x_kwadraat_voorlaatste
  verschil_volgende_kw = verschil_vorige_kwadraten + 2
  x_voorlaatste += 1
  x_laatste += 1
  x_kwadraat_voorlaatste = x_kwadraat_laatste
  x_kwadraat_laatste += verschil_volgende_kw

print(f"Het kwadraat van {x_laatste} = {x_kwadraat_laatste}")
```

Bits en hoe ze worden opgeslagen

- Bit: binary digit
- In de wiskunde:
  - Cijfer van het binair talstelsel (basis = 2)
    - Vergelijk: decimaal talstelsel (basis = 10)
    - Mogelijke waarden: 0 en 1
- In de informatica:
  - Symbool gebruikt bij het voorstellen van gegevens
    - Numerieke waarden, Booleaanse waarden, letters en leestekens, beelden, geluiden, …
    - Gegevens vertaald naar bitpatronen
  - Vb. ASCII code voor “!”-teken = 00100001
  - Deze voorstelling is gebaseerd op afspraken en standaarden
    - ASCII: American Standard Code for Information Interchange

Bits en hoe ze worden opgeslagen

ASCII

Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex
line feed	00001010	0A	>	00111110	3E	^	01011110	5E
carriage return	00001011	0B	?	00111111	3F	_	01011111	5F
space	00100000	20	@	01000000	40	`	01100000	60
!	00100001	21	A	01000001	41	a	01100001	61
"	00100010	22	B	01000010	42	b	01100010	62
#	00100011	23	C	01000011	43	c	01100011	63
$	00100100	24	D	01000100	44	d	01100100	64
%	00100101	25	E	01000101	45	e	01100101	65
&	00100110	26	F	01000110	46	f	01100110	66
'	00100111	27	G	01000111	47	g	01100111	67
(	00101000	28	H	01001000	48	h	01101000	68
)	00101001	29	I	01001001	49	i	01101001	69
*	00101010	2A	J	01001010	4A	j	01101010	6A
+	00101011	2B	K	01001011	4B	k	01101011	6B
'	00101100	2C	L	01001100	4C	l	01101100	6C
.	00101101	2D	M	01001101	4D	m	01101101	6D
/	00101110	2E	N	01001110	4E	n	01101110	6E
0	00101111	2F	O	01001111	4F	o	01101111	6F
1	00110000	30	P	01010000	50	p	01110000	70
2	00110001	31	Q	01010001	51	q	01110001	71
3	00110010	32	R	01010010	52	r	01110010	72
4	00110011	33	S	01010011	53	s	01110011	73
5	00110100	34	T	01010100	54	t	01110100	74
6	00110101	35	U	01010101	55	u	01110101	75
7	00110110	36	V	01010110	56	v	01110110	76
8	00110111	37	W	01010111	57	w	01110111	77
9	00111000	38	X	01011000	58	x	01111000	78
:	00111001	39	Y	01011001	59	y	01111001	79
;	00111010	3A	Z	01011010	5A	z	01111010	7A
<	00111011	3B	[	01011011	5B	{	01111011	7B
=	00111100	3C	\	01011100	5C	|	01111100	7C
	00111101	3D	]	01011101	5D	}	01111101	7D

Bits en hoe ze worden opgeslagen

Waarom binair

- Waarom maken we het ons zo moeilijk? Een talstelsel met basis 2?
- Met informatietechnologie:
  - Elk apparaat/materiaal dat twee standen heeft, kan gebruikt worden voor gegevensopslag
    - Elektronische schakeling (geleidend / niet geleidend)
    - Magnetiseerbaar oppervlak (positief geladen / negatief geladen)
    - Reflecterend oppervlak (reflecterend / niet reflecterend)
    - Vakje siliciumdioxide (vol / leeg)
    - Condensator (geladen / niet geladen)
    - …

Bits en hoe ze worden opgeslagen

Waarom binair

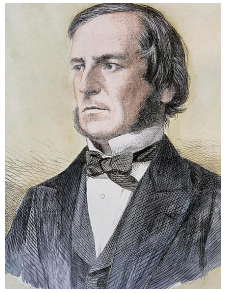
- Hoe weten we hier welk teken welk bitpatroon krijgt?
- Gebaseerd op standaarden (afspraken)
- Maar wel op zo'n manier dat het logisch is (voor rekenen, voor sorteren, voor berekeningen)
- Maar we zullen zien dat zelfs voor b.v. negatieve getallen er verschillende voorstellingswijzen zijn

Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex
line feed	00001010	0A	>	00111110	3E	^	01011110	5E
carriage return	00001011	0B	?	00111111	3F	_	01011111	5F
space	00100000	20	@	01000000	40	`	01100000	60
!	00100001	21	A	01000001	41	a	01100001	61
"	00100010	22	B	01000010	42	b	01100010	62
#	00100011	23	C	01000011	43	c	01100011	63
$	00100100	24	D	01000100	44	d	01100100	64
%	00100101	25	E	01000101	45	e	01100101	65
&	00100110	26	F	01000110	46	f	01100110	66
'	00100111	27	G	01000111	47	g	01100111	67
(	00101000	28	H	01001000	48	h	01101000	68
)	00101001	29	I	01001001	49	i	01101001	69
*	00101010	2A	J	01001010	4A	j	01101010	6A
+	00101011	2B	K	01001011	4B	k	01101011	6B
'	00101100	2C	L	01001100	4C	l	01101100	6C
.	00101101	2D	M	01001101	4D	m	01101101	6D
/	00101110	2E	N	01001110	4E	n	01101110	6E
0	00101111	2F	O	01001111	4F	o	01101111	6F
1	00110000	30	P	01010000	50	p	01110000	70
2	00110001	31	Q	01010001	51	q	01110001	71
3	00110010	32	R	01010010	52	r	01110010	72
4	00110011	33	S	01010011	53	s	01110011	73
5	00110100	34	T	01010100	54	t	01110100	74
6	00110101	35	U	01010101	55	u	01110101	75
7	00110110	36	V	01010110	56	v	01110110	76
8	00110111	37	W	01010111	57	w	01110111	77
9	00111000	38	X	01011000	58	x	01111000	78
:	00111001	39	Y	01011001	59	y	01111001	79
;	00111010	3A	Z	01011010	5A	z	01111010	7A
<	00111011	3B	[	01011011	5B	{	01111011	7B
=	00111100	3C	\	01011100	5C	|	01111100	7C
	00111101	3D	]	01011101	5D	}	01111101	7D

Bits en hoe ze worden opgeslagen

Booleaanse algebra

- Wiskundige theorie van bewerkingen op Booleaanse waarden
  - True / False, Waar / Onwaar, 1 / 0, Ja / Nee
  - Kan gebruikt worden om bits te bewerken



George Boole

Operatoren:

Afkorting	Operator	Vergelijkbaar met	Symbool
AND	de conjunctie	vergelijkbaar met vermenigvuldigen	x·y
XOR	de disjunctie, exclusieve keuze	vergelijkbaar met optellen	x+y
NOT	het complement	vergelijkbaar met negatie	¬x (of ~x of !x)
(I)OR	inclusieve keuze		

Bits en hoe ze worden opgeslagen

Booleaanse algebra

The AND operation

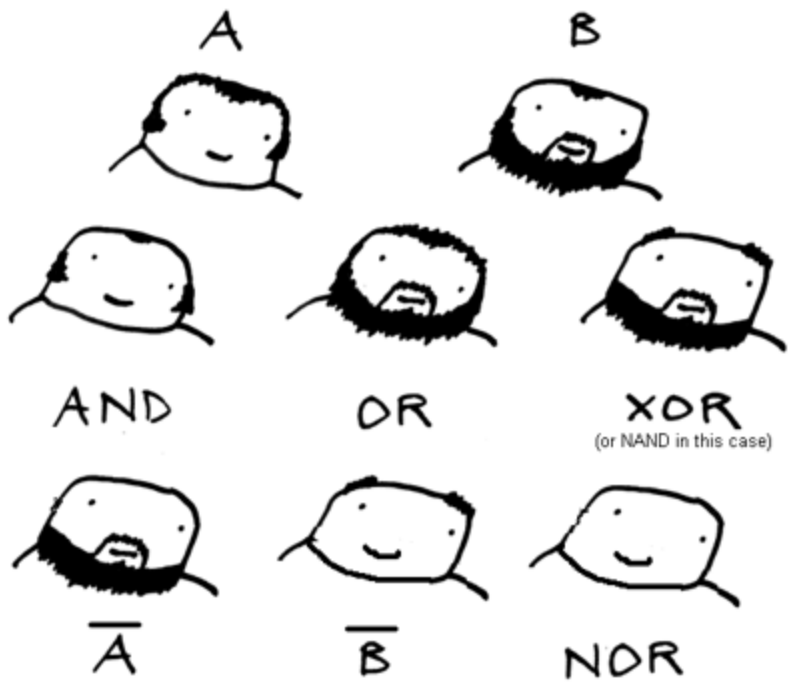
0	0	1	1
AND 0	AND 1	AND 0	AND 1
0	0	0	1

The OR operation

0	0	1	1
OR 0	OR 1	OR 0	OR 1
0	1	1	1

The XOR operation

0	0	1	1
XOR 0	XOR 1	XOR 0	XOR 1
0	1	1	0



Bits en hoe ze worden opgeslagen

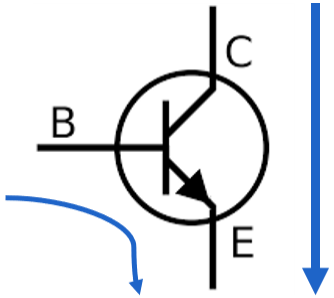
Gates en flip-flops

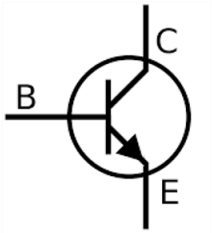
- Gate (“poort”)
  - Apparaat dat een Booleaanse bewerking uitvoert
  - Verschillende implementaties mogelijk, b.v. met elektronische schakelingen
  - Eén (NOT) of twee invoeren (AND, OR, XOR)
  - Eén uitvoer

Bits en hoe ze worden opgeslagen

Gates en flip-flops

- Transistor
  - Drie laagjes siliconemateriaal
  - Emitter, Base, Collector

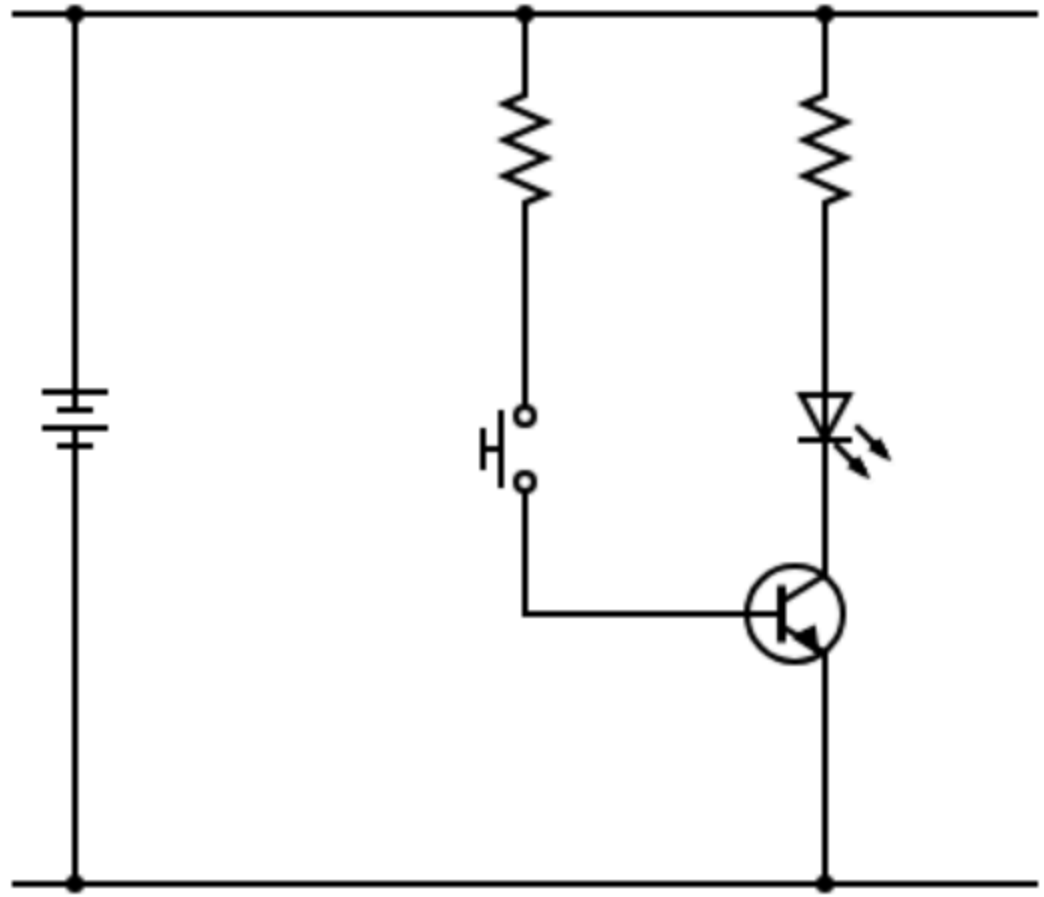


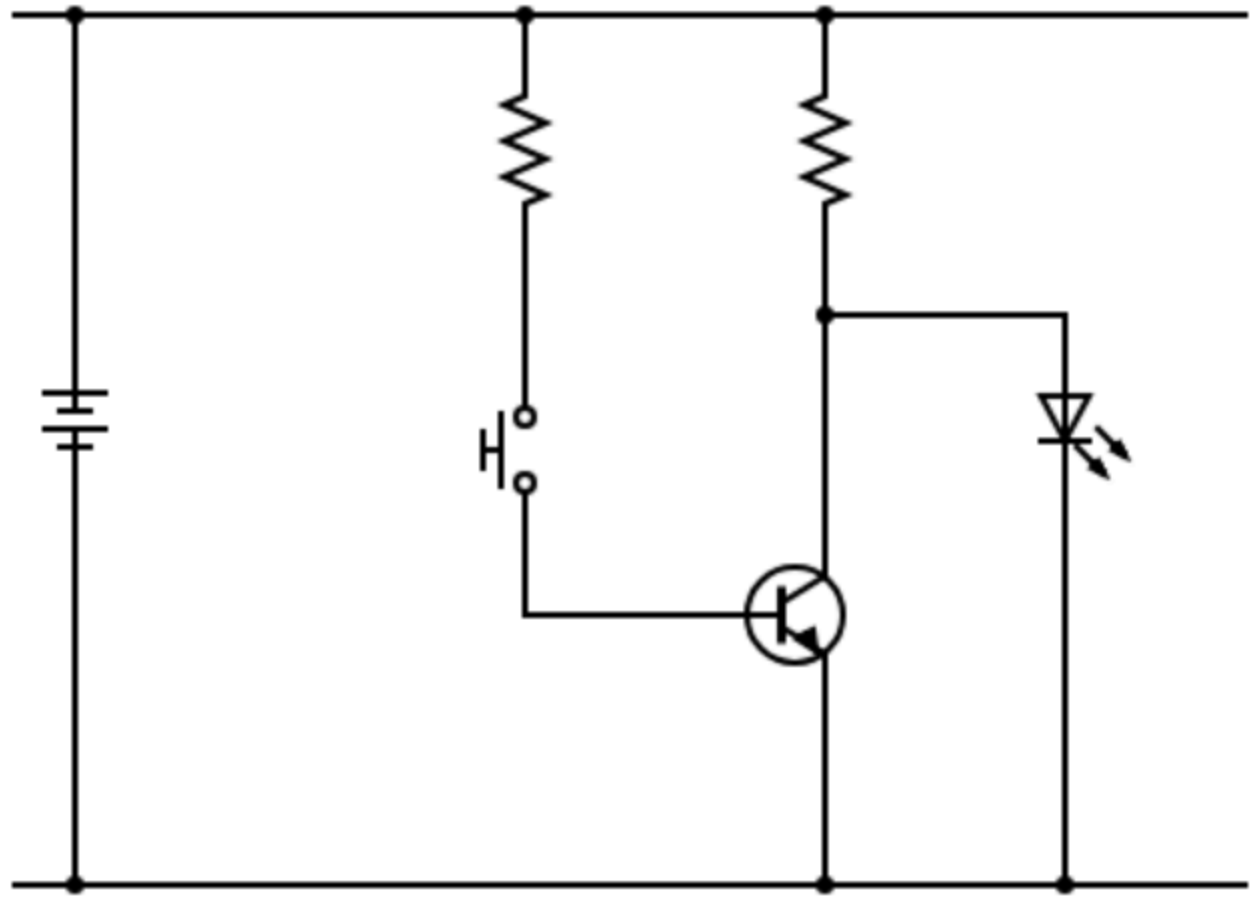


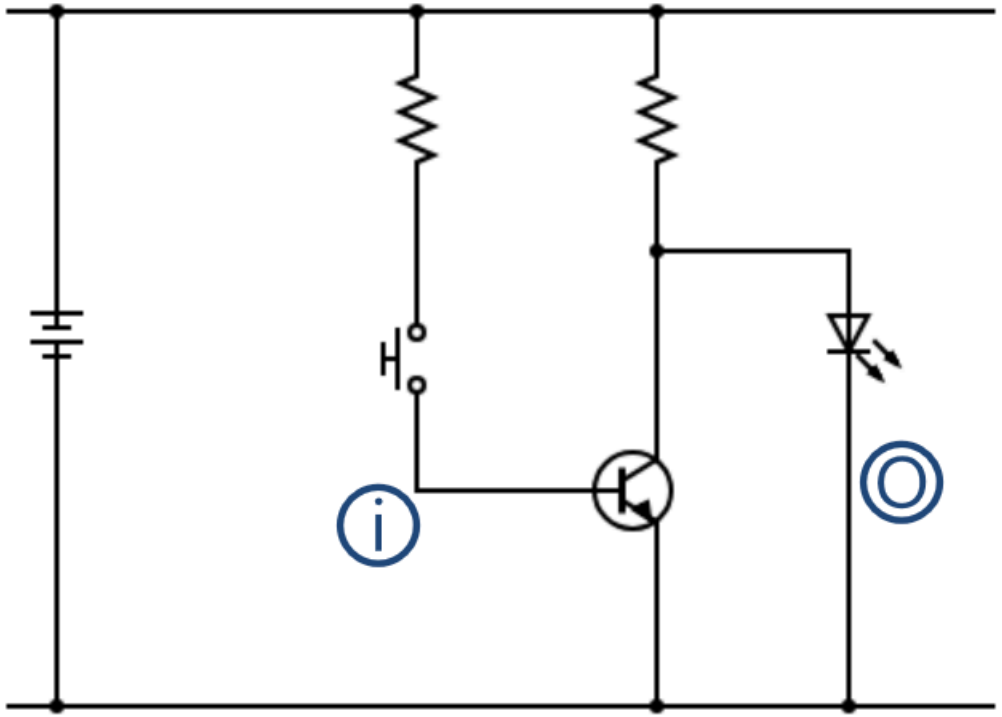
Transistor “aan”: klein spanningsverschil tussen B-E → grote stroom kan vloeien tussen C-E
Transistor “uit”: geen spanningsverschil tussen B-E → geen stroom kan vloeien tussen C-E


2N
3904
331

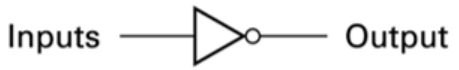

CA3046
H 9327



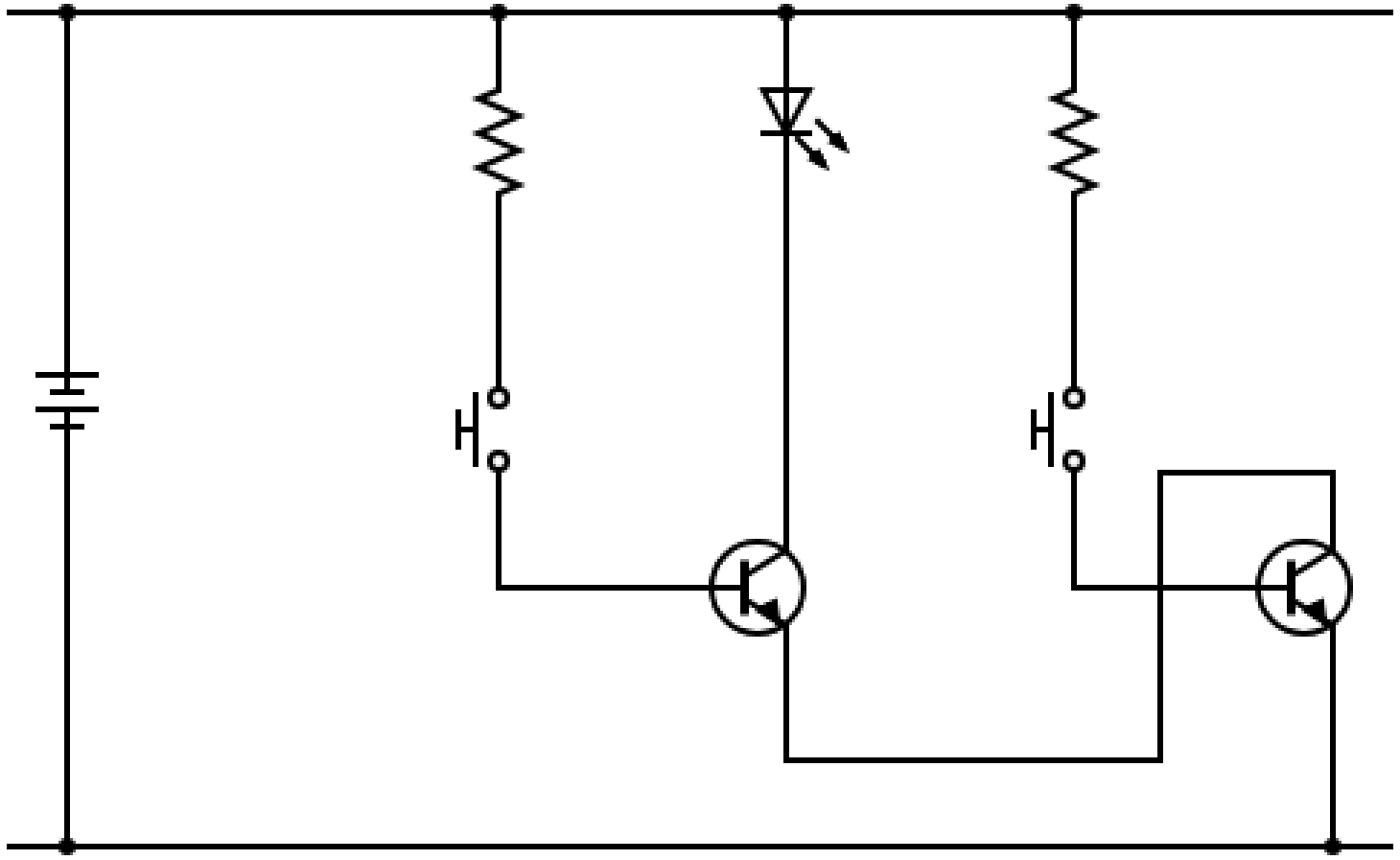


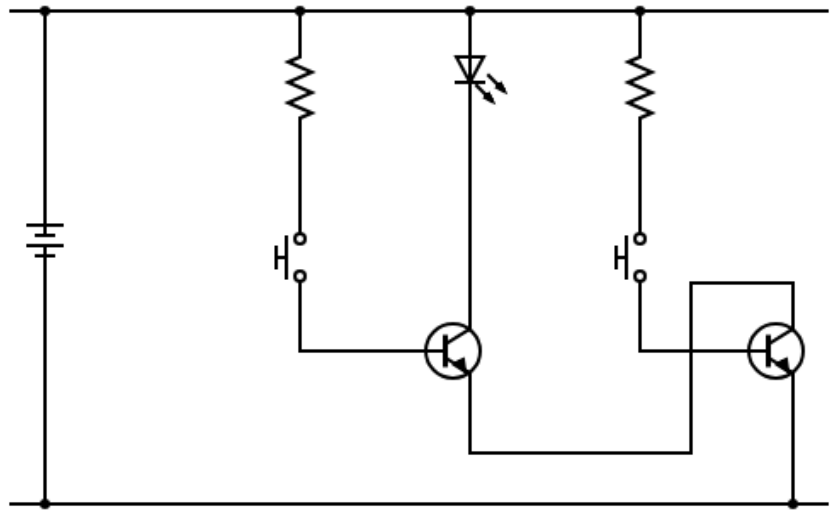


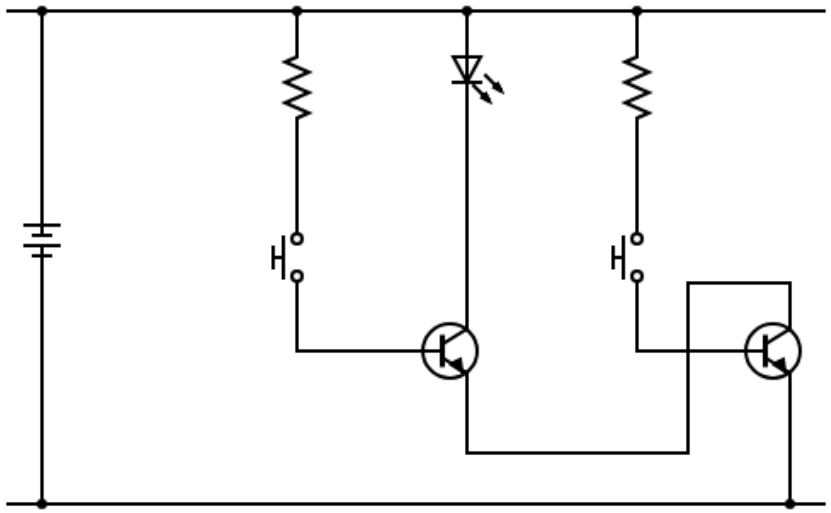
NOT

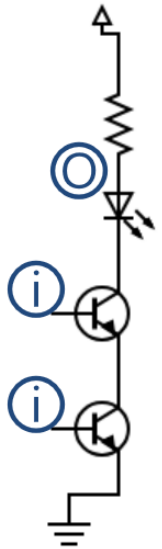


Inputs	Output
0	1
1	0

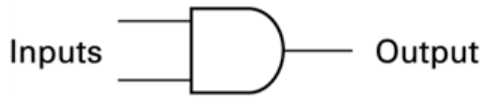




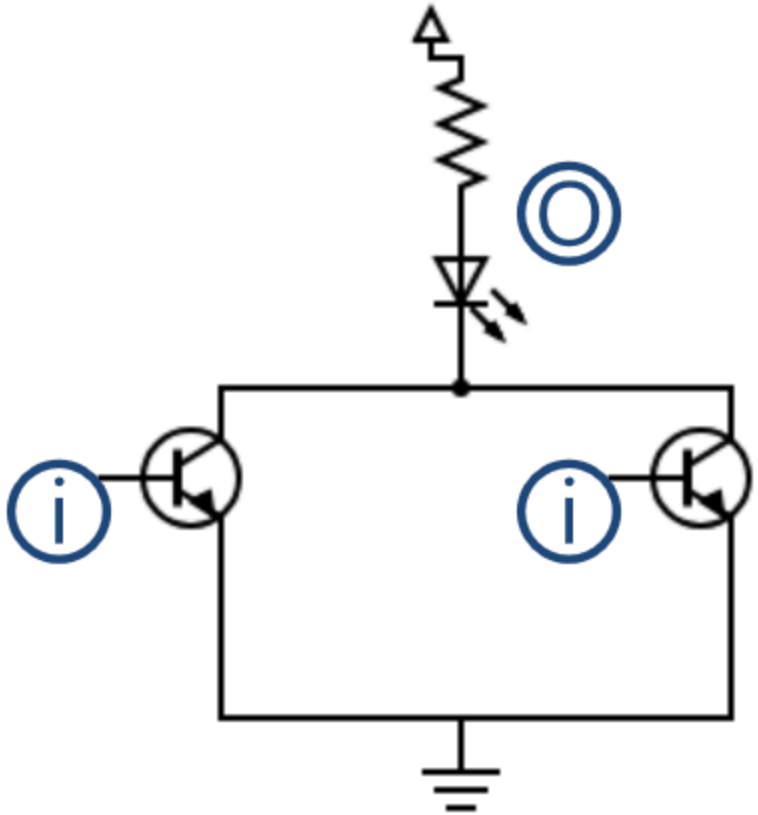




AND

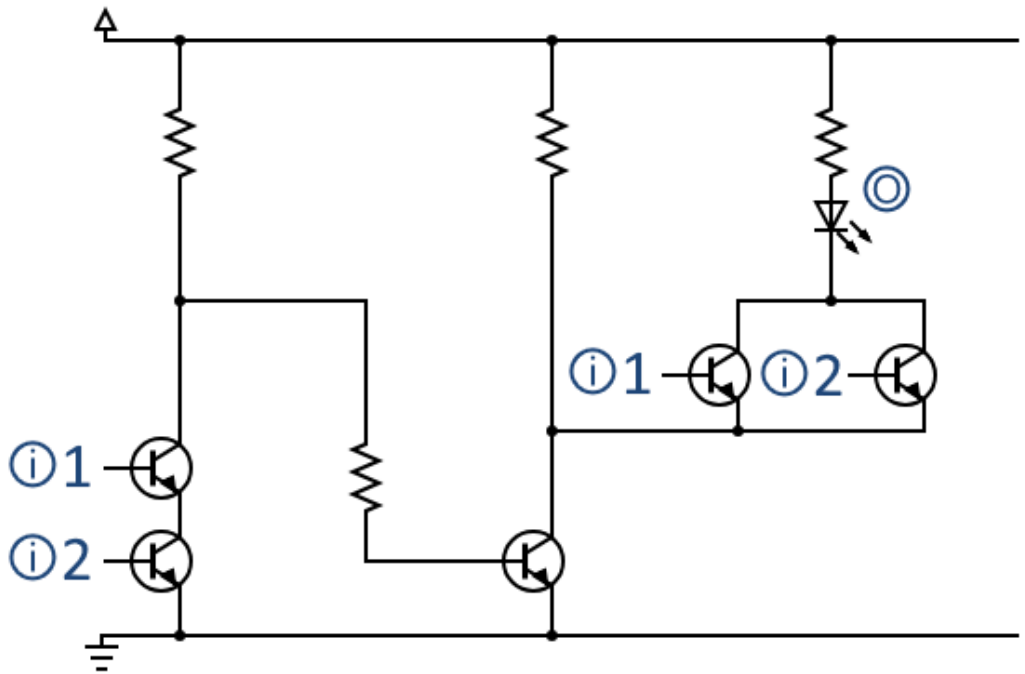


Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1



OR

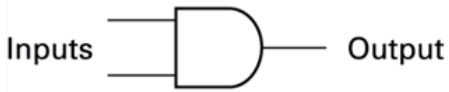
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1



XOR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

AND

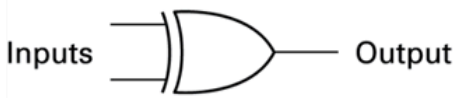


Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR

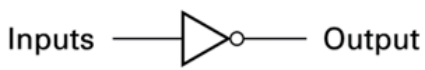
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR

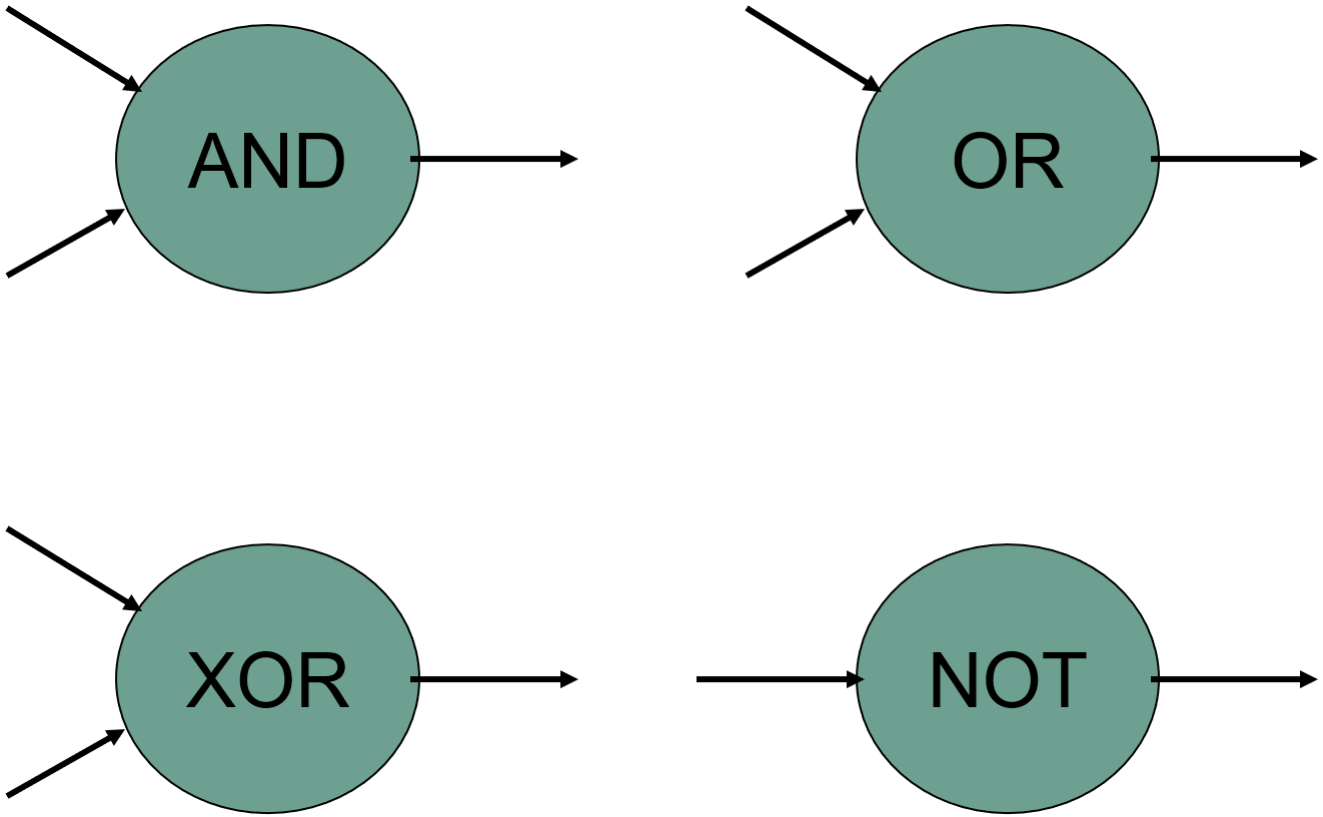


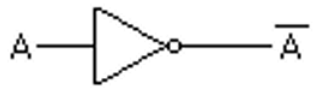
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

NOT

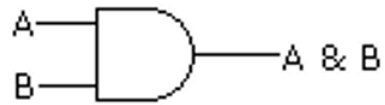


Inputs	Output
0	1
1	0

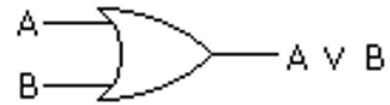

AND
OR
XOR
NOT



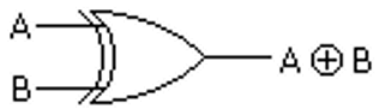
A	NOT A
0	1
1	0



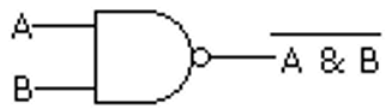
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



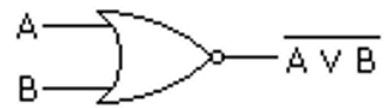
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

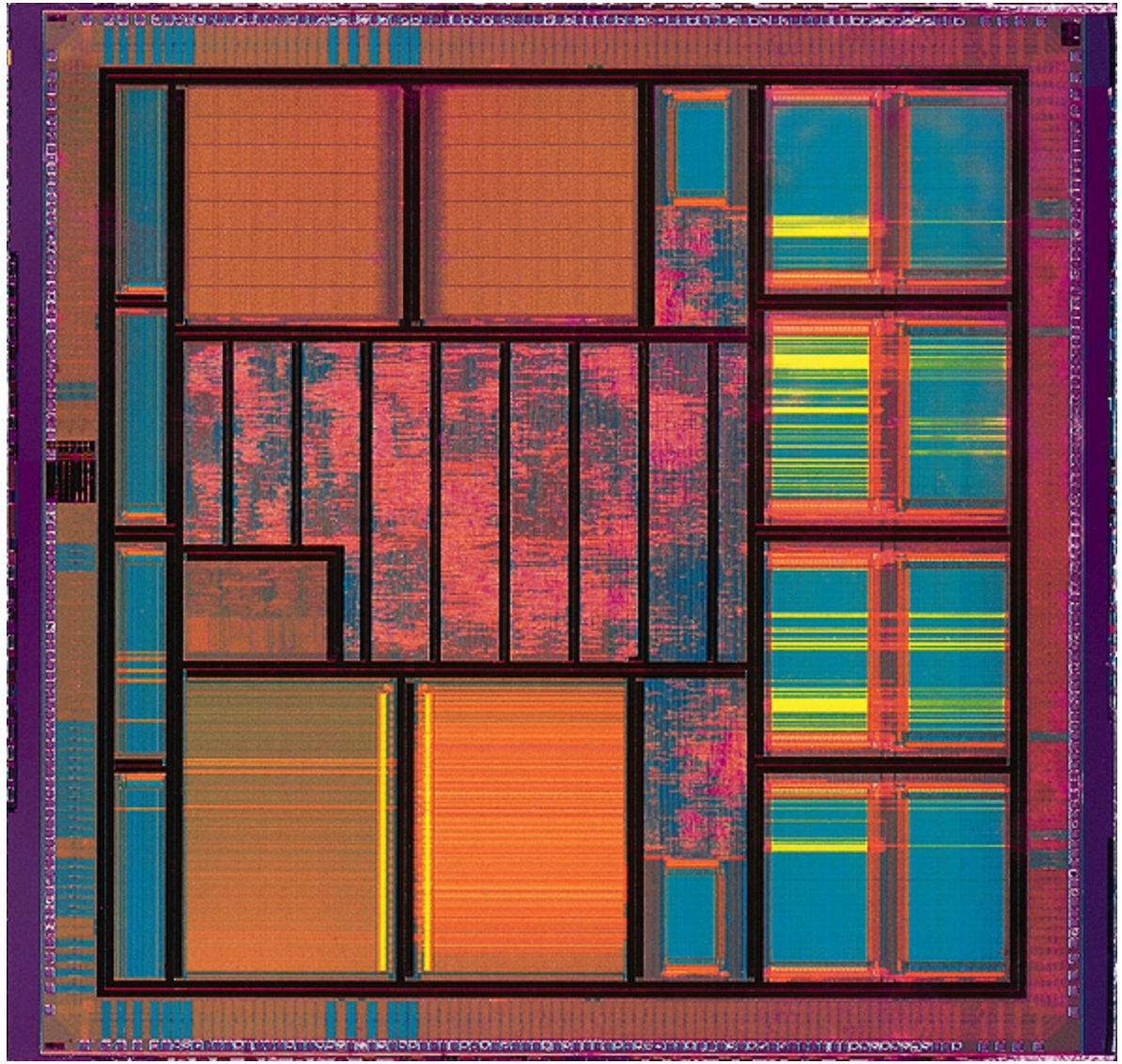


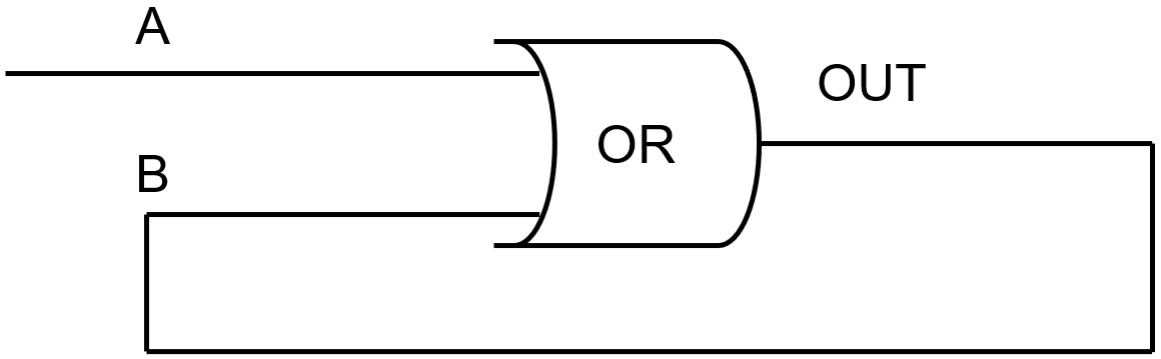
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

Bits en hoe ze worden opgeslagen

Gates en flip-flops

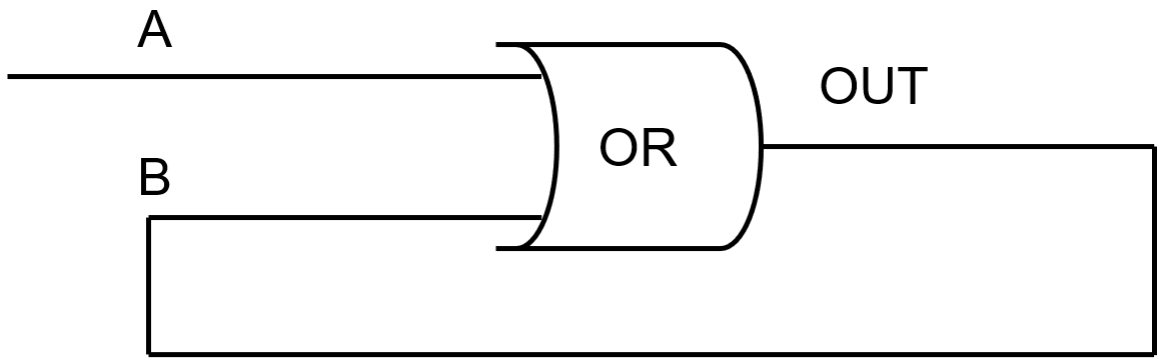
- We kunnen berekeningen doen, maar hoe slaan we iets op?
  - Een computer kan rekenen, maar heeft ook geheugen nodig
- Flip-flop
  - Schakeling voor het opslaan van één bit
  - Twee invoerlijnen, één uitvoerlijn
  - Gebouwd met gates (dus: transistoren)
    - Beide invoerlijnen 0 → uitvoerlijn verandert niet
    - Bovenste invoerlijn op 1 → uitvoerlijn wordt 1 (verandert niet als bovenste invoerlijn terug op 0)
    - Onderste invoerlijn op 1 → uitvoerlijn wordt 0 (verandert niet als onderste invoerlijn terug op 0)
  - Very Large-Scale Integration (VLSI)
    - Integratie van miljoenen flip-flops samen met besturingsschakelingen op een chip



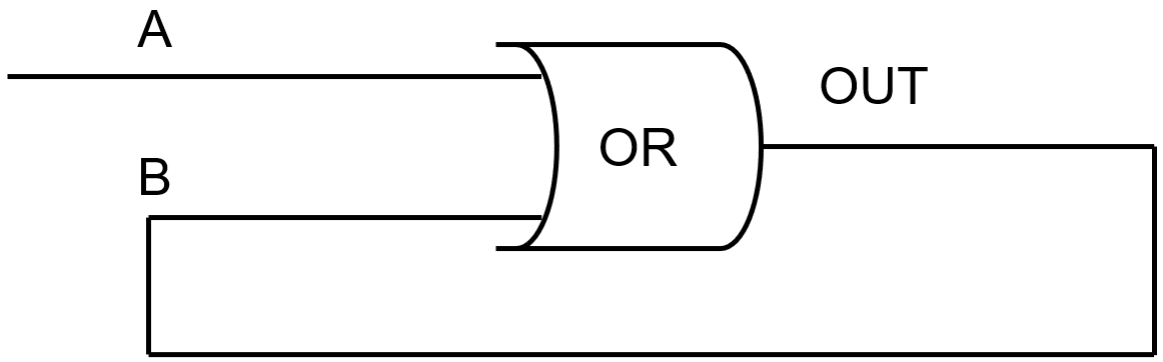


OR

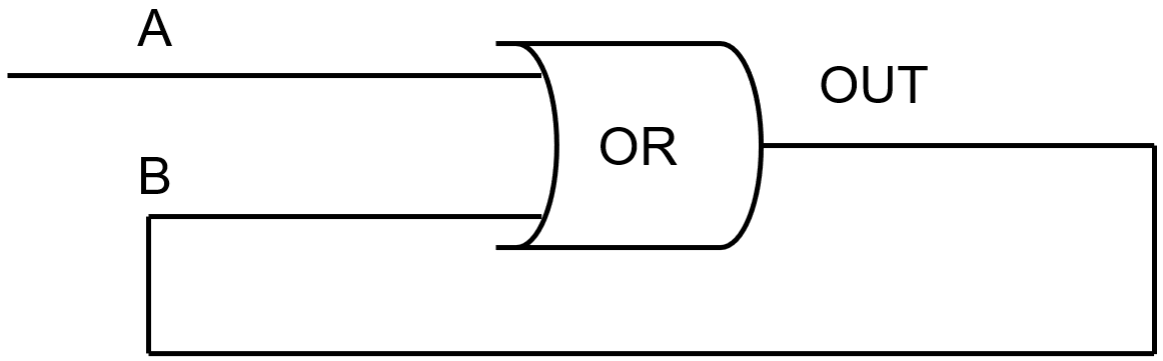
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1



A (IN)	B	OUT
0	0	0

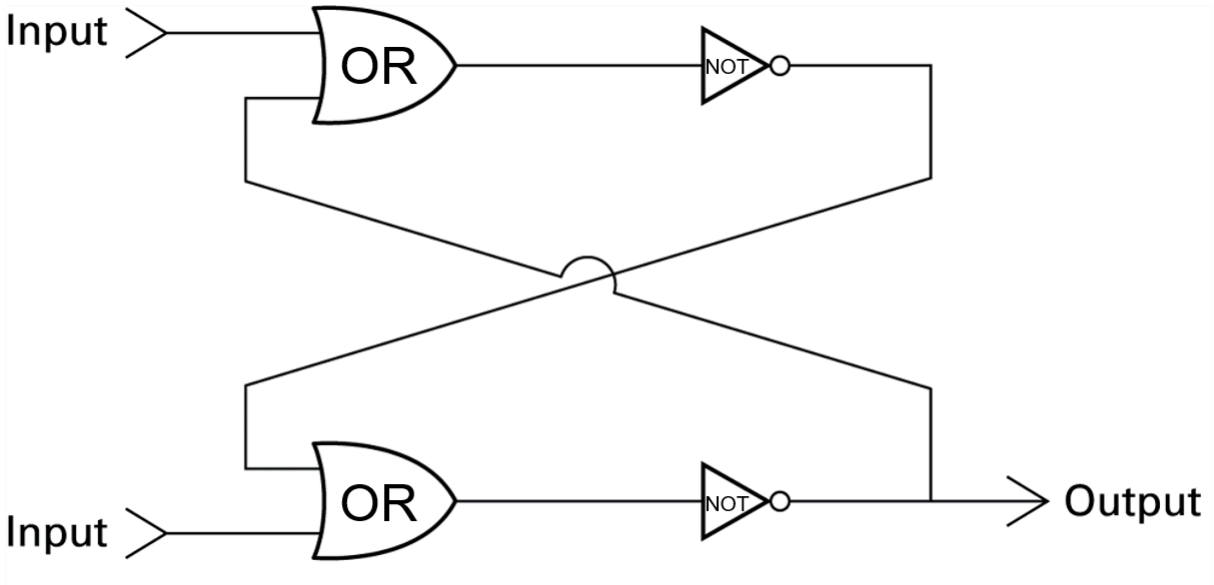


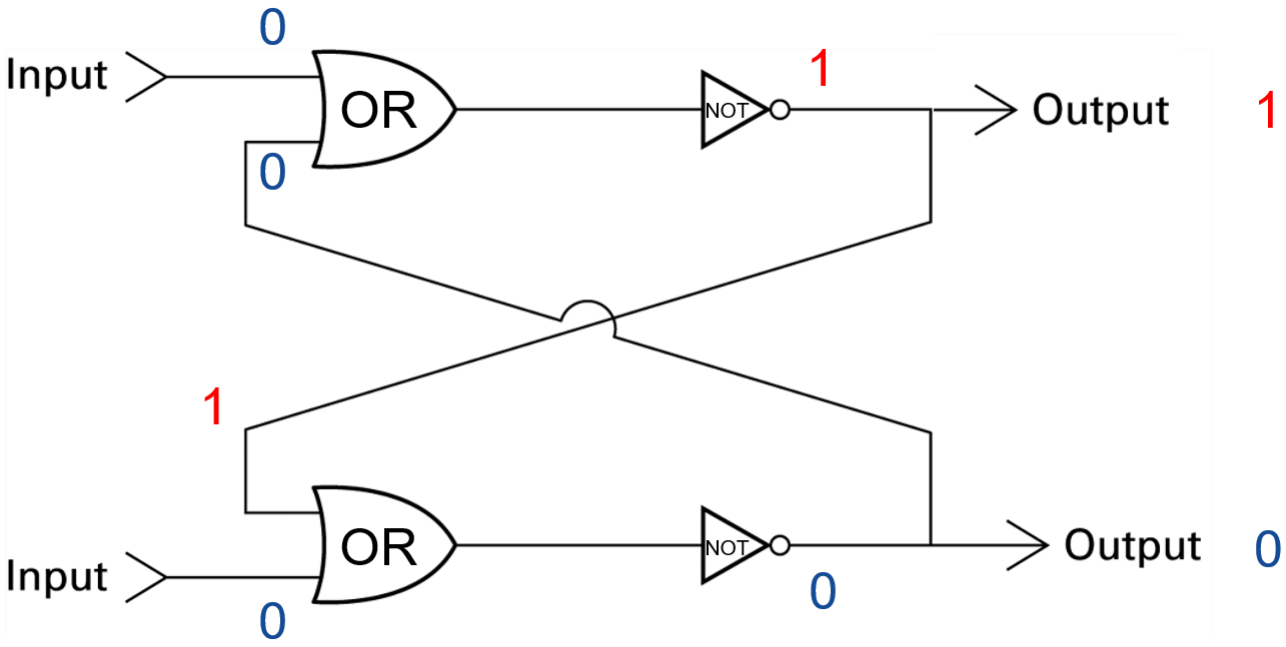
A (IN)	B	OUT
0	0	0
1	0	1
1	1	1



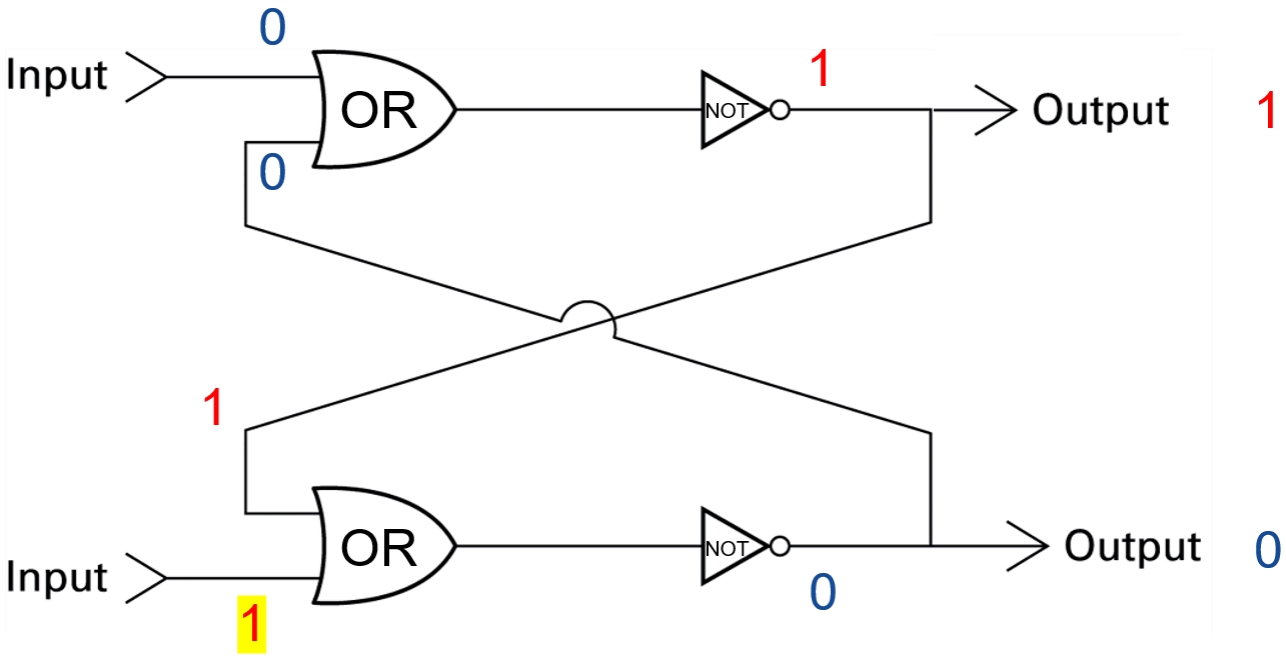
A (IN)	B	OUT
0	0	0
1	0	1
1	1	1
0	1	1

- Dit is een “latch”: eenmaal IN = 1 blijft OUT = 1, ook als IN weer 0 wordt
- Maar nog geen echt geheugen: hoe krijgen we de waarde weer op 0?

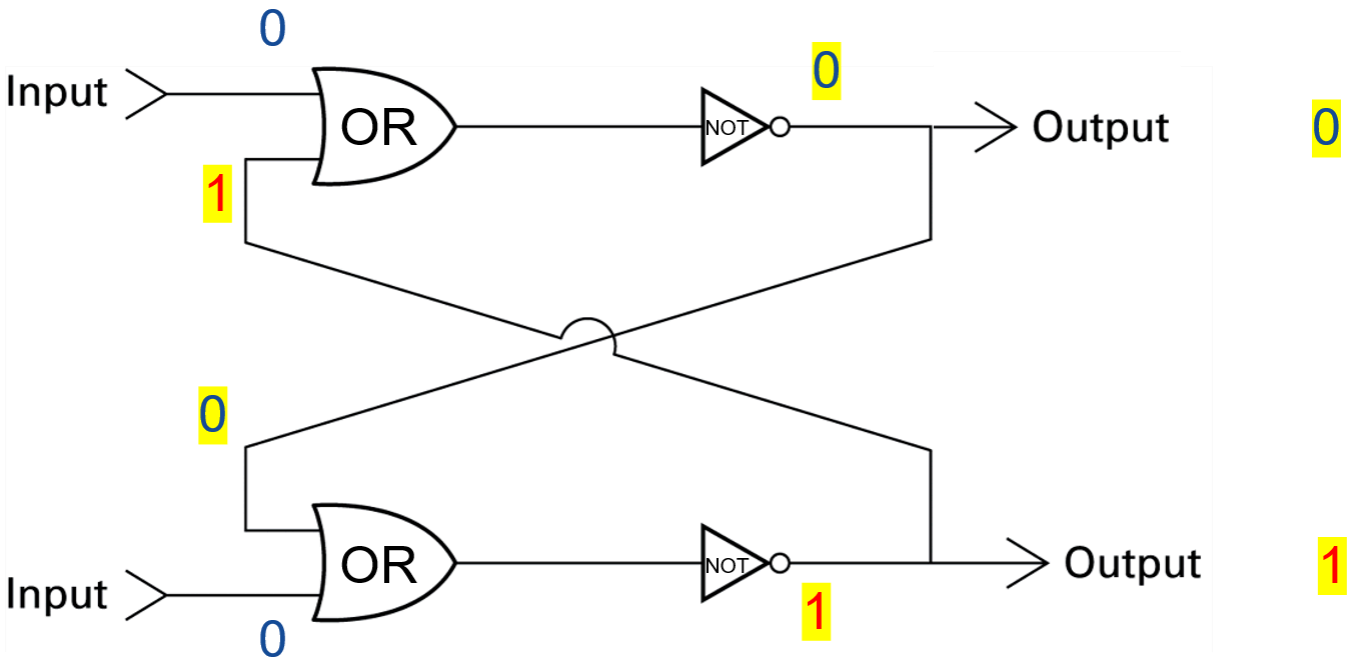

Input
OR
NOT
Input
OR
NOT
Output



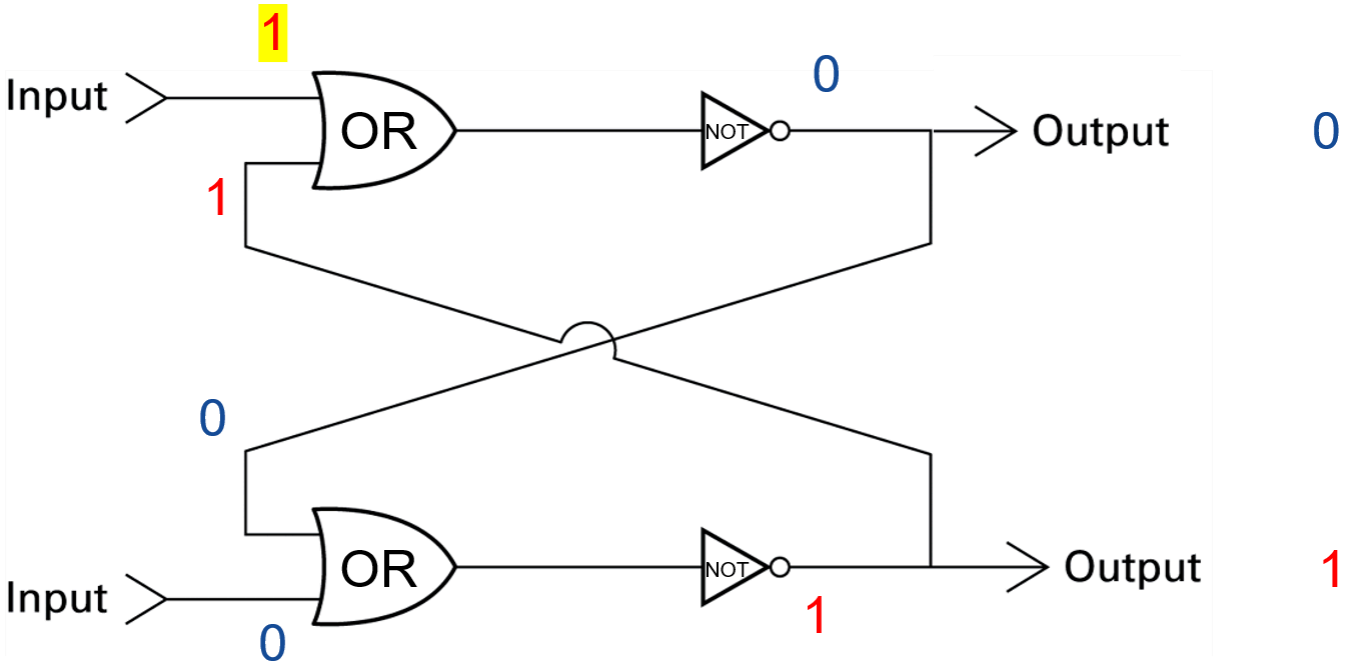
- Stabiele situatie…



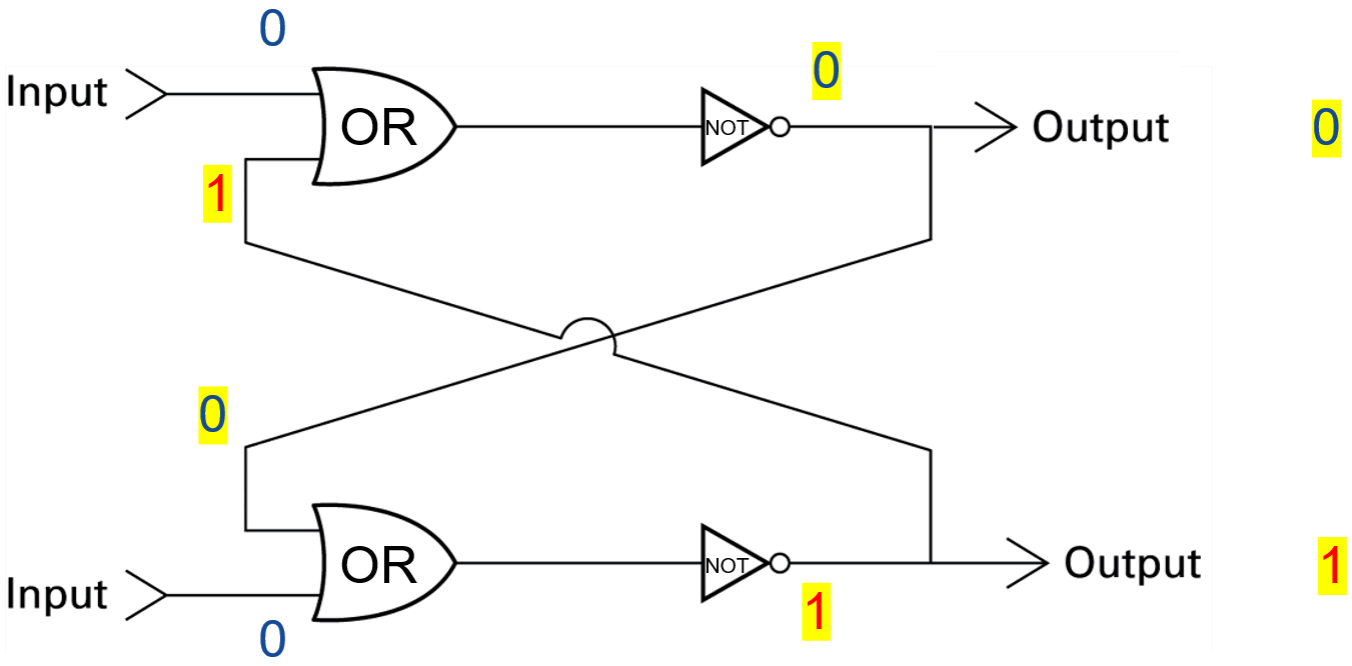
- Input wijzigt… gebeurt er iets?



- Bovenste lijn indrukken en loslaten… Nieuwe stabiele situatie…



- Nu bovenste input wijzigen doet niets

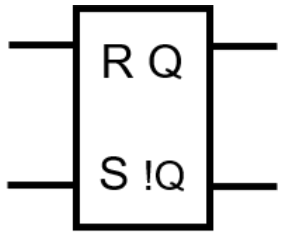


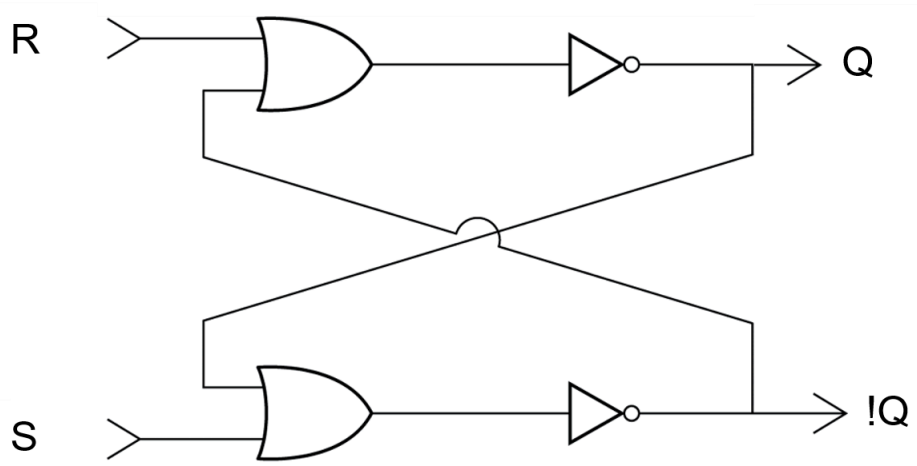
- Twee latches: mogelijkheid om 1 bit op te slaan (output 1 of 2)
- Deze schakeling kan “flip-floppen” tussen 0 en 1

Bits en hoe ze worden opgeslagen

Gates en flip-flops

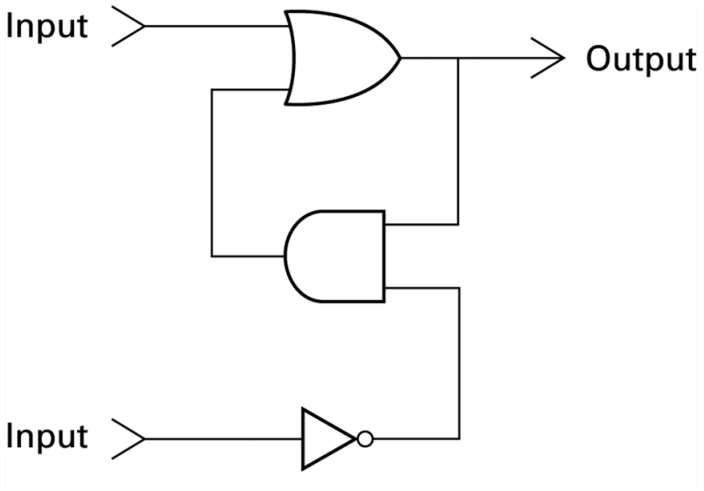
- Dit is de “SR latch” (set, reset): een flip-flop met twee latches
  - Inputs R, S
  - Outputs Q, !Q
  - Om Q = 1 te zetten moet S = 1 worden



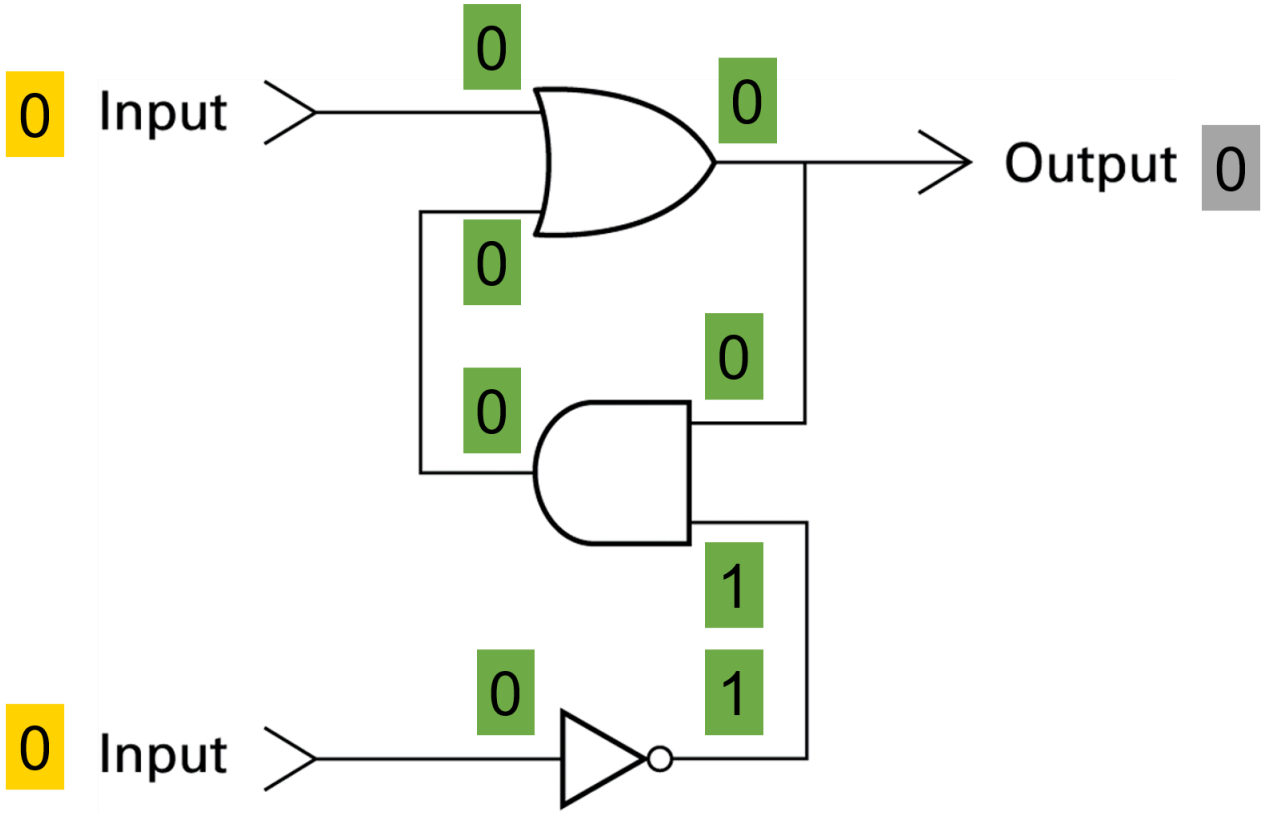


Bits en hoe ze worden opgeslagen

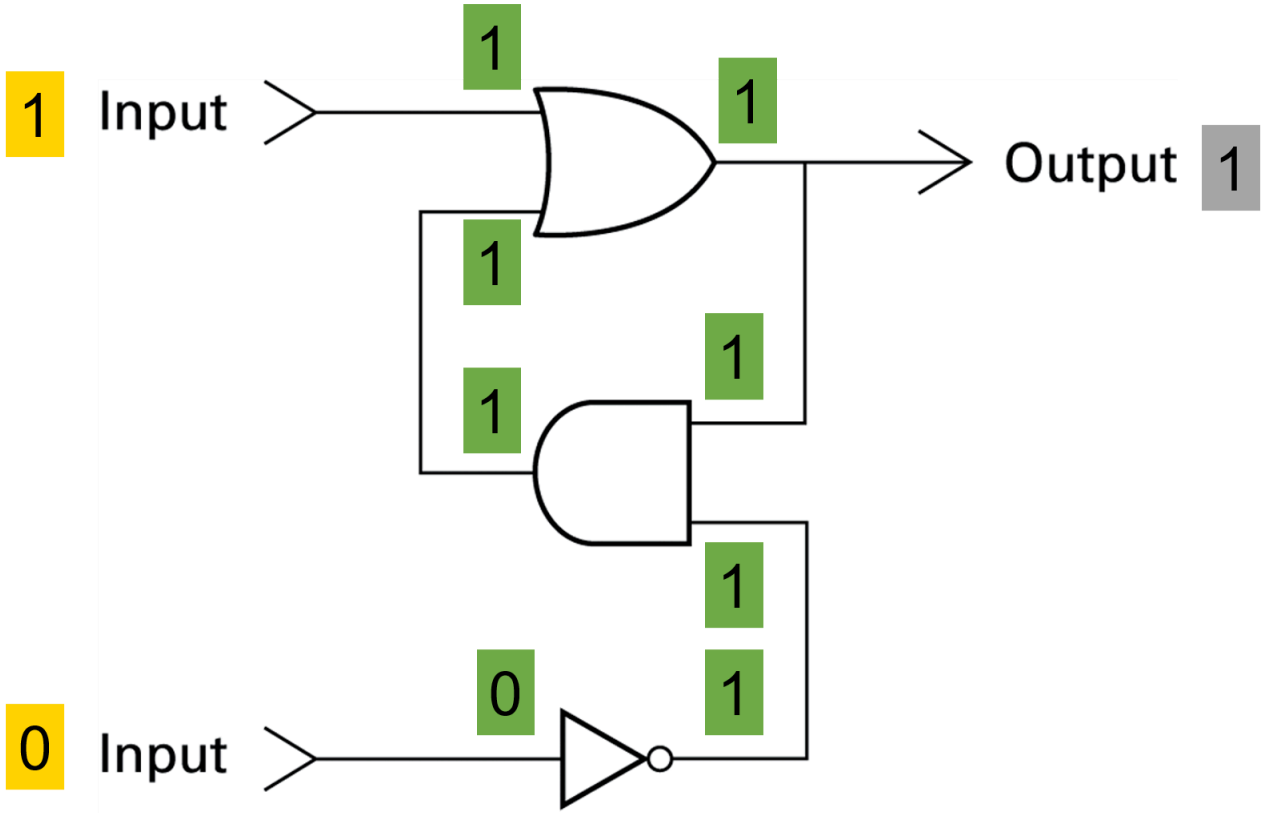
Gates en flip-flops



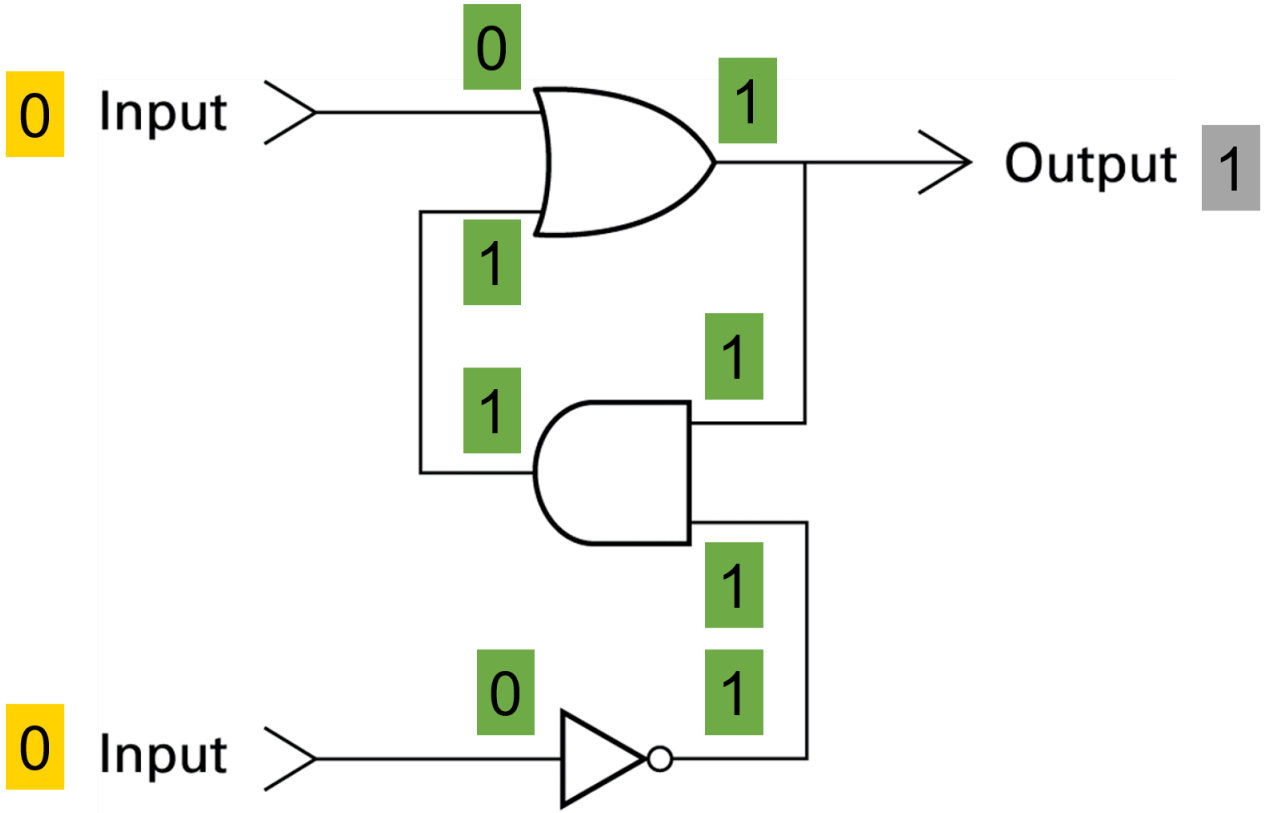
- Een alternatieve implementatie
  - Met een OR, **AND** en NOT gate



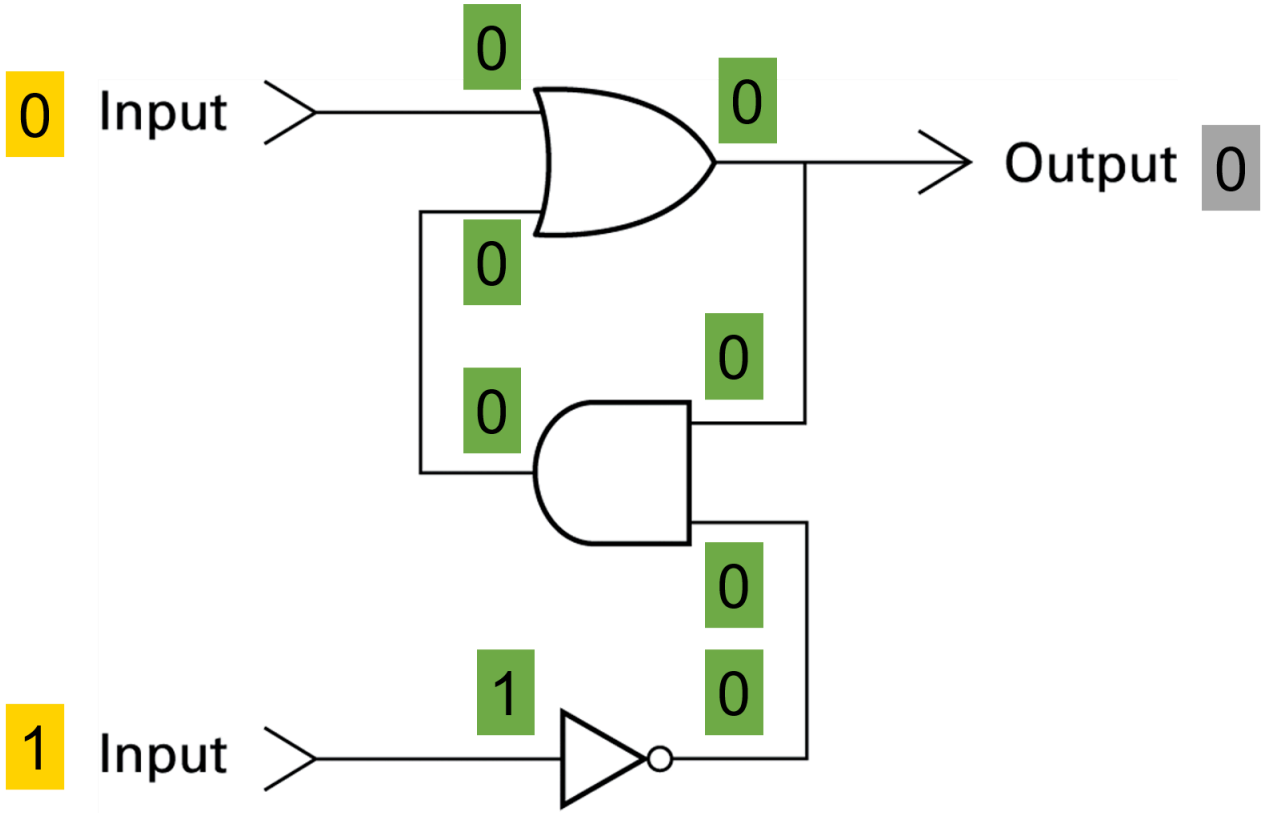
- Beginsituatie: beide invoeren en uitvoer 0



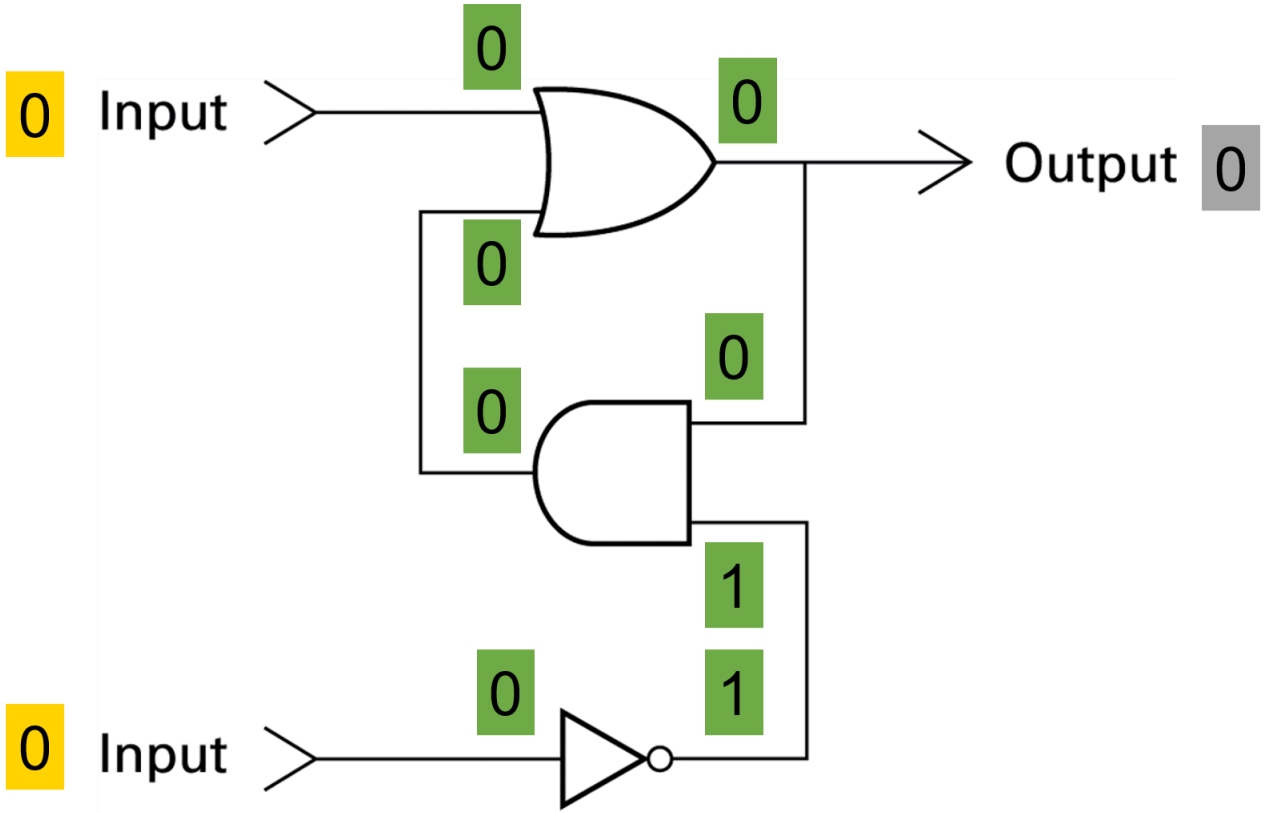
- De bovenste invoerlijn op 1 zetten verandert de output van 0 naar 1



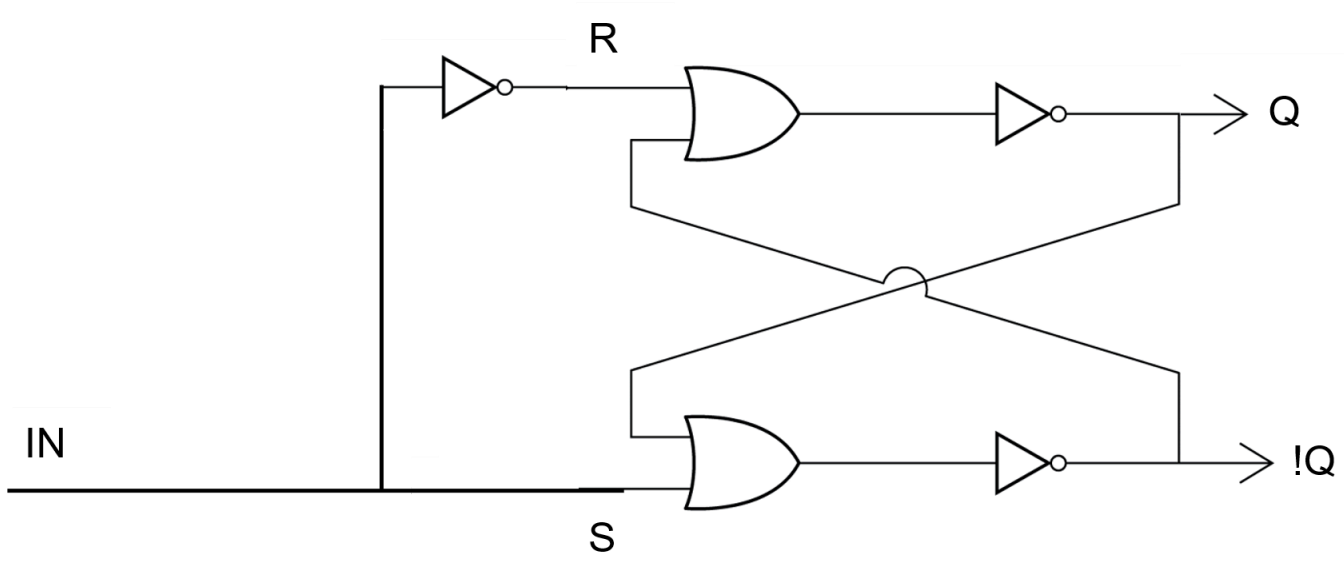
- De uitvoer blijft 1 ook al verandert de bovenste invoerlijn van 1 naar 0 (latch)



- De onderste invoerlijn op 1 zetten verandert de uitvoer van 1 naar 0



- De uitvoer blijft weer 0 ook al verandert de onderste invoerlijn terug naar 0

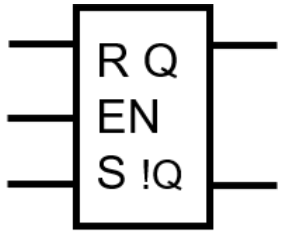


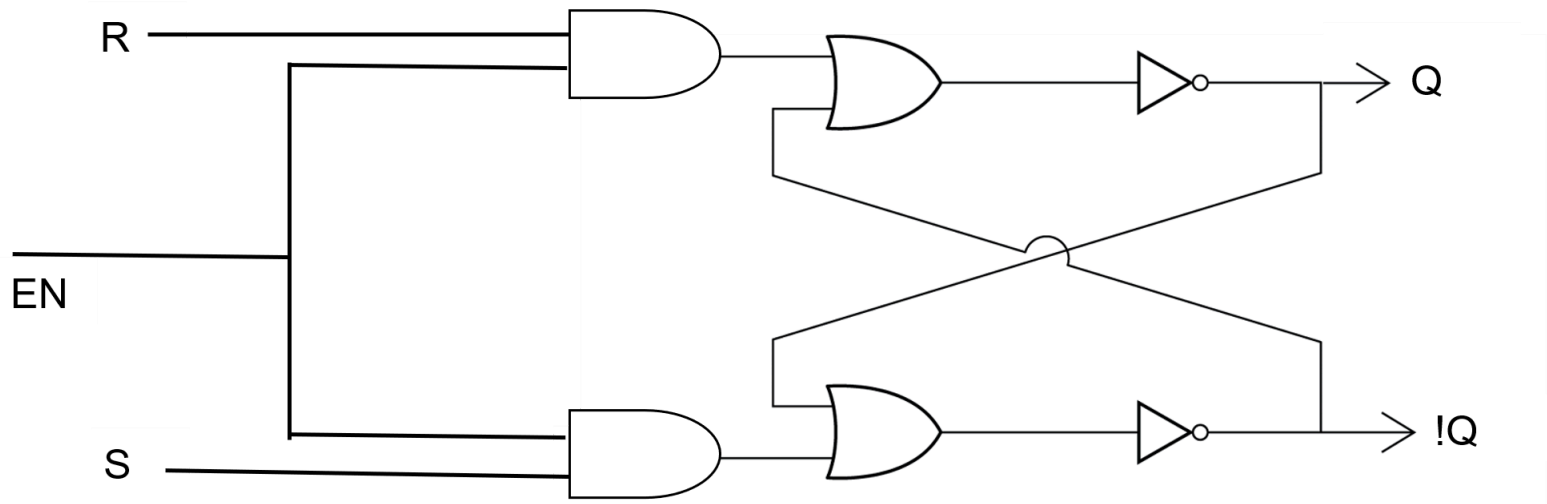
- Alternatief concept met één invoerlijn
- Dit is geen echte “latch” meer, maar…

Bits en hoe ze worden opgeslagen

Gates en flip-flops

- Dit is de “SR latch met enable”
  - Een computer drukt niet op knopjes
  - In plaats daarvan is er een “klok” (kwartskristal in een chip of iets dergelijks) dat om de zoveel tijd een puls uitstuurt: het enable signaal

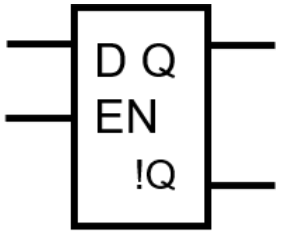


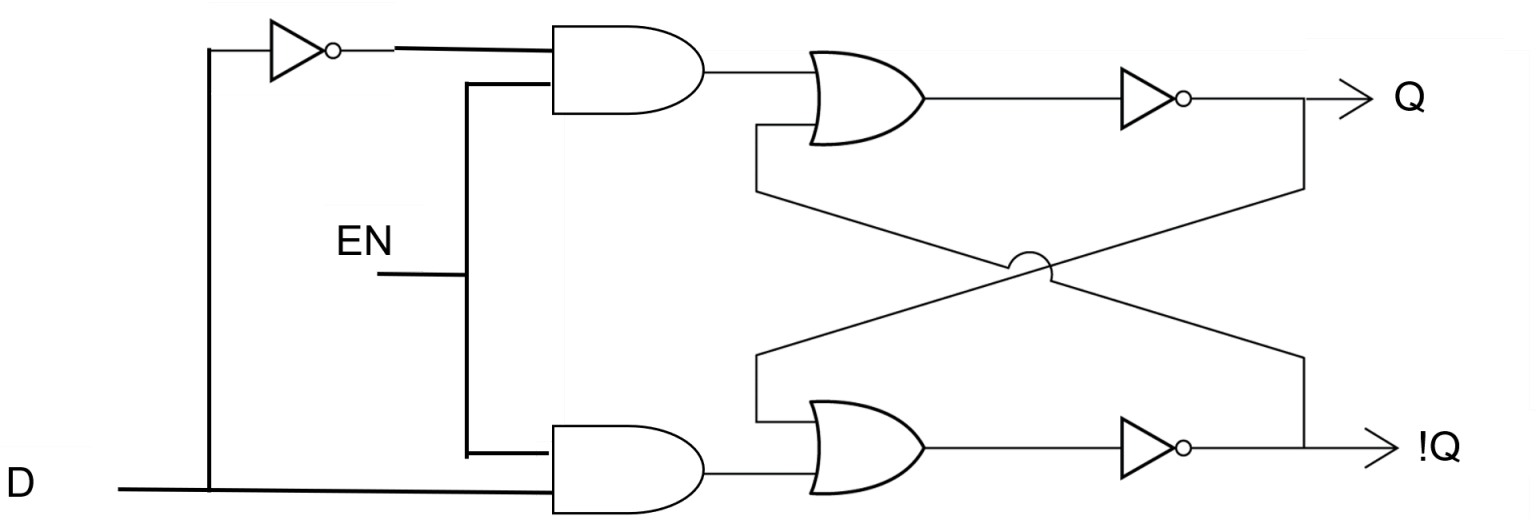


Bits en hoe ze worden opgeslagen

Gates en flip-flops

- Dit is de “D latch”
  - Enable (EN): geeft de klokpuls aan dat we mogen lezen uit D?
  - Data (D): wat is het “data” signaal dat we moeten inlezen (0 of 1)?





Bits en hoe ze worden opgeslagen

Gates en flip-flops

- En zo komen we eindelijk tot het concept van een computer
  - Binaire operaties met gates voor “berekeningen”
  - Geheugen met latches en flip-flops
  - Plus: een klok
    - Elektrische stroom voor het circuit aan te drijven (EN)
    - Maar ook voor data (gegevens) weer te geven (D)

Bits en hoe ze worden opgeslagen

Gates en flip-flops

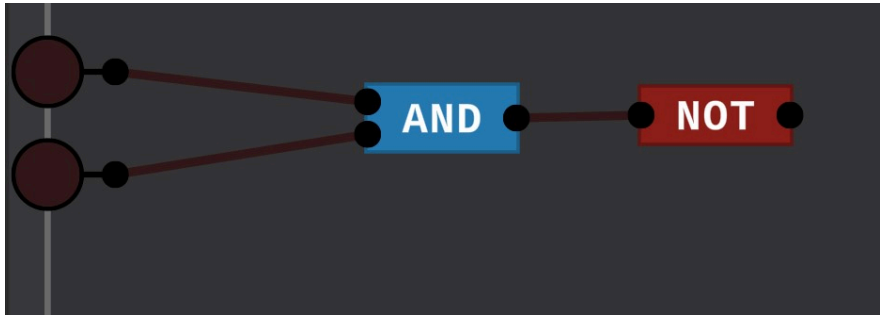
- Overdreven? Niet echt…
- Voor zij die wat meer uitleg/informatie willen over de basis elektronische bouwstenen van een (“ouderwetse”) computer (bron: Ben Eater):
  - Making logic gates from transistors: https://www.youtube.com/watch?v=sTu3LwpF6XI
  - SR latch: https://www.youtube.com/watch?v=KM0DdEaY5sY
  - D latch: https://www.youtube.com/watch?v=peCh_859q7Q
  - D flip-flop: https://www.youtube.com/watch?v=YW-_GkUguMM
  - JK flip-flop: https://www.youtube.com/watch?v=F1OC5e7Tn_o
  - Master-slave JK flip-flop: https://www.youtube.com/watch?v=rXHSB5w7CyE

Bits en hoe ze worden opgeslagen

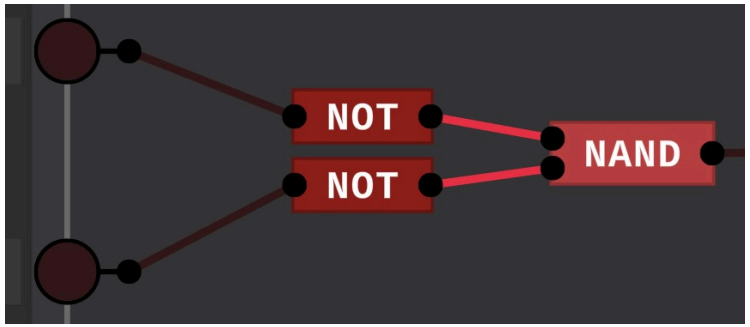
Gates en flip-flops

- Je kan dit ook eens nabouwen met Digital Logic Sim
- Bijhorende video's: https://www.youtube.com/playlist?list=PLFt_AvWsXl0dPhqVsKt1Ni_46ARyiCGSq

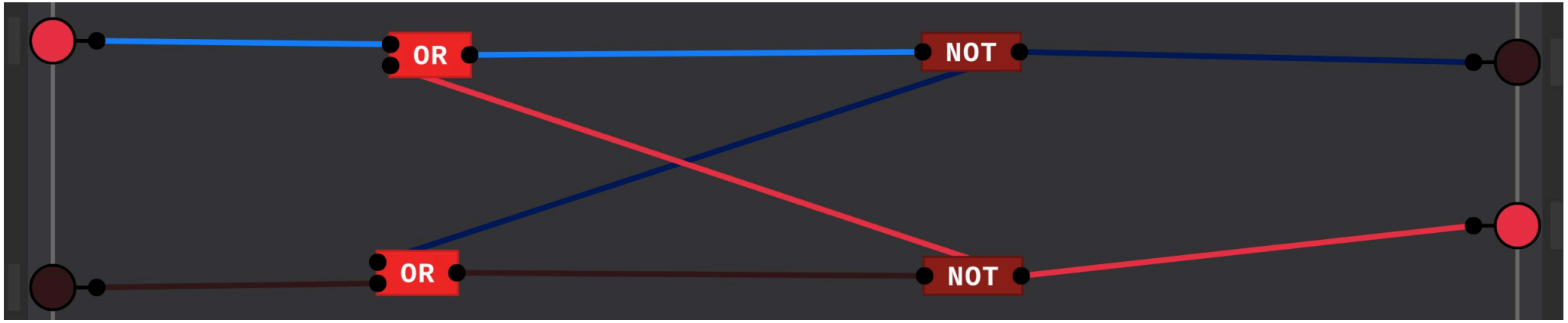
Met AND en NOT maken we een NAND:



En vervolgens een OR:



En dat is voldoende voor een SR latch:



LogicSimSRLatch

Macuyiko

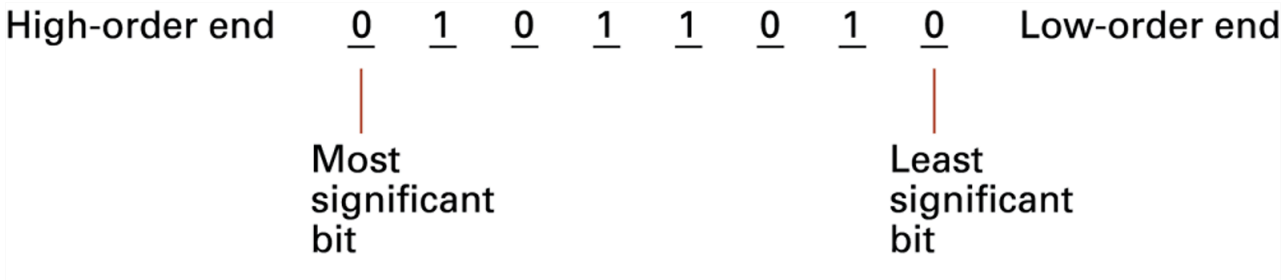
Watch on

https://www.youtube.com/v/MLDjVPUSny8

Het werkgeheugen van de computer

- Cel
  - Groep van acht schakelingen die elk een bit kunnen bevatten
  - Reeks van acht bits = een byte
  - Heeft een unieke naam, een “adres”
    - Opéénvolgende getallen, vanaf nul
    - Geordend → “vorige cel”, “volgende cel” zinvol

Het werkgeheugen van de computer

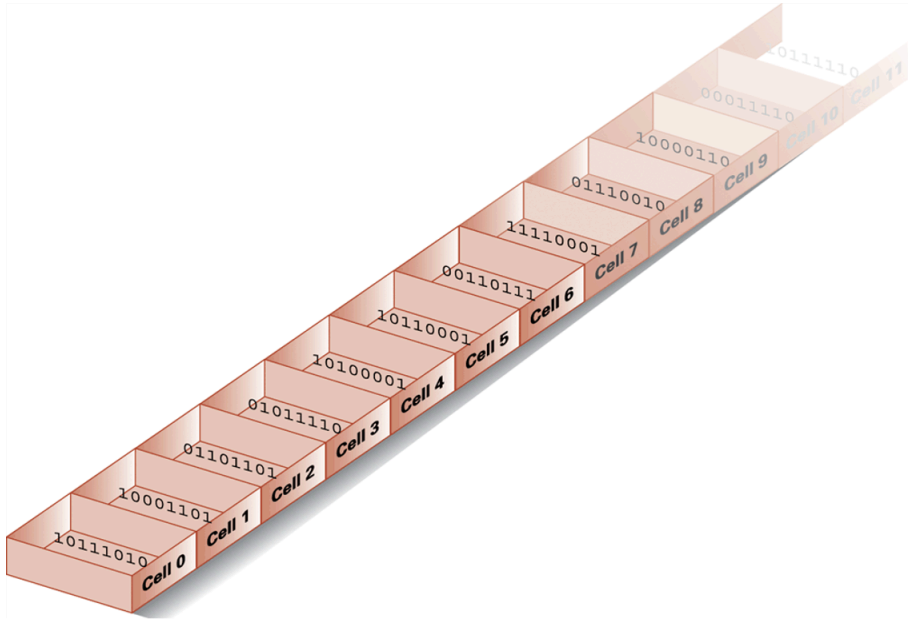


- Benoemen van de bits in de byte
  - Voor ons is het vermoedelijk het makkelijkste om het meest significante cijfer vooraan te plaatsen (vergelijkbaar met ons decimaal talstelsel)
  - Maar omgekeerd kan ook, dit heet Endianness

> The adjective endian has its origin in the writings of 18th century Anglo-Irish writer Jonathan Swift. In the 1726 novel Gulliver's Travels, he portrays the conflict between sects of Lilliputians divided into those breaking the shell of a boiled egg from the big end or from the little end. He called them the "Big-Endians" and the "Little-Endians"
>
> … en heel wat andere "architecturen"

Het werkgeheugen van de computer

- Vermits de cellen in het werkgeheugen geordend zijn, vormt hun inhoud één lange reeks van bits
  - Dit maakt het mogelijk om bitpatronen op te slaan met een lengte groter dan b.v. acht
  - Een bitpatroon van 16 bits kan opgeslagen worden in twee opeenvolgende geheugencellen



Het werkgeheugen van de computer

- De waarde van cellen 0 tot en met 2 is:
  - 10010101 00110001 00001101
  - Moeilijk te lezen, te onthouden, over te schrijven…
- Hexadecimale notatie = verkorte notatie voor het noteren van bitstromen
  - Een ander talstelsel…

Het werkgeheugen van de computer

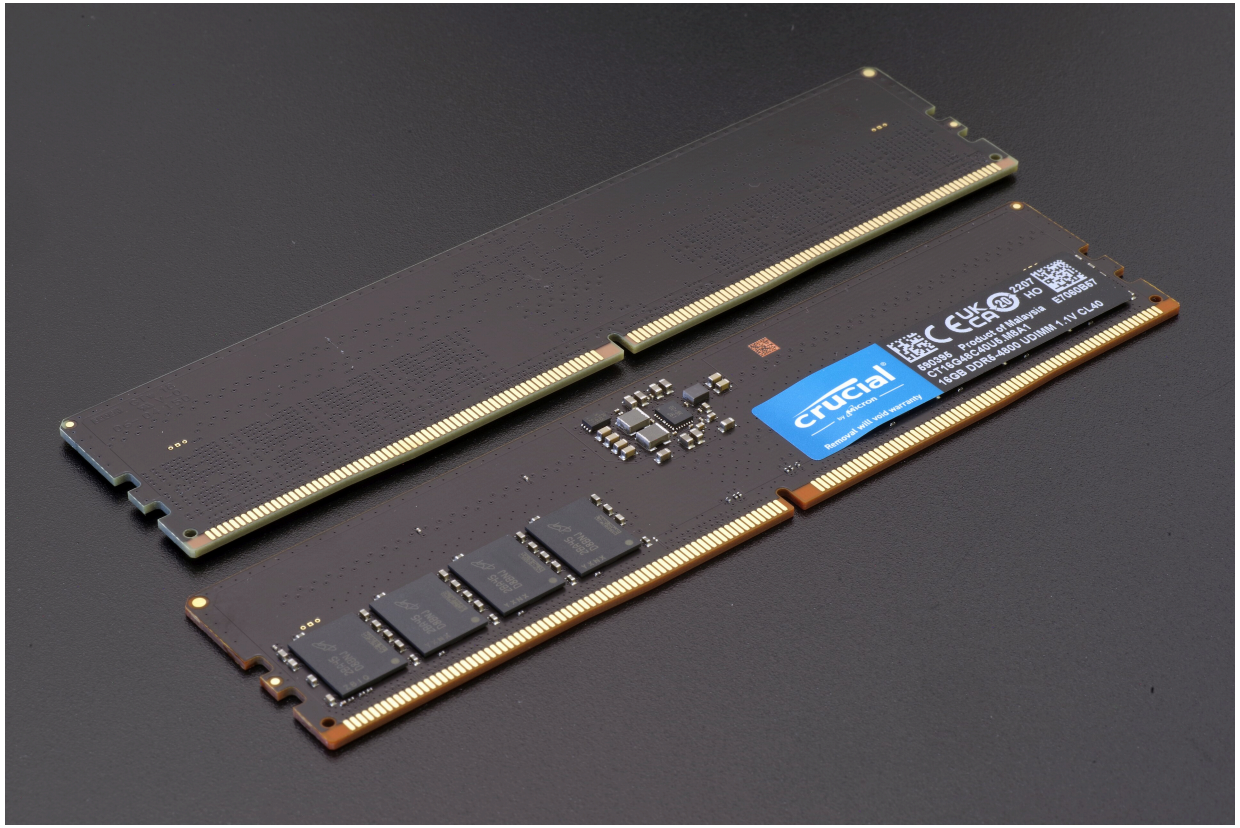
- In de wiskunde:
  - Cijfer van het 16-delig talstelsel (basis = 16)
    - Mogelijke waarden: 0-9, A-F: 0-F → 16 waardes
    - $16 = 2^4$ → om eenzelfde aantal waarden voor te stellen zijn 4 bits nodig
- In de informatica:
  - Lengte bitpatronen meestal een veelvoud van 4
  - 9 = 1001
  - 1001 0101 0011 0001 0000 1101 = 95310D
  - (Ook hier: endianness: “Big endian”)

Bit pattern	Hexadecimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

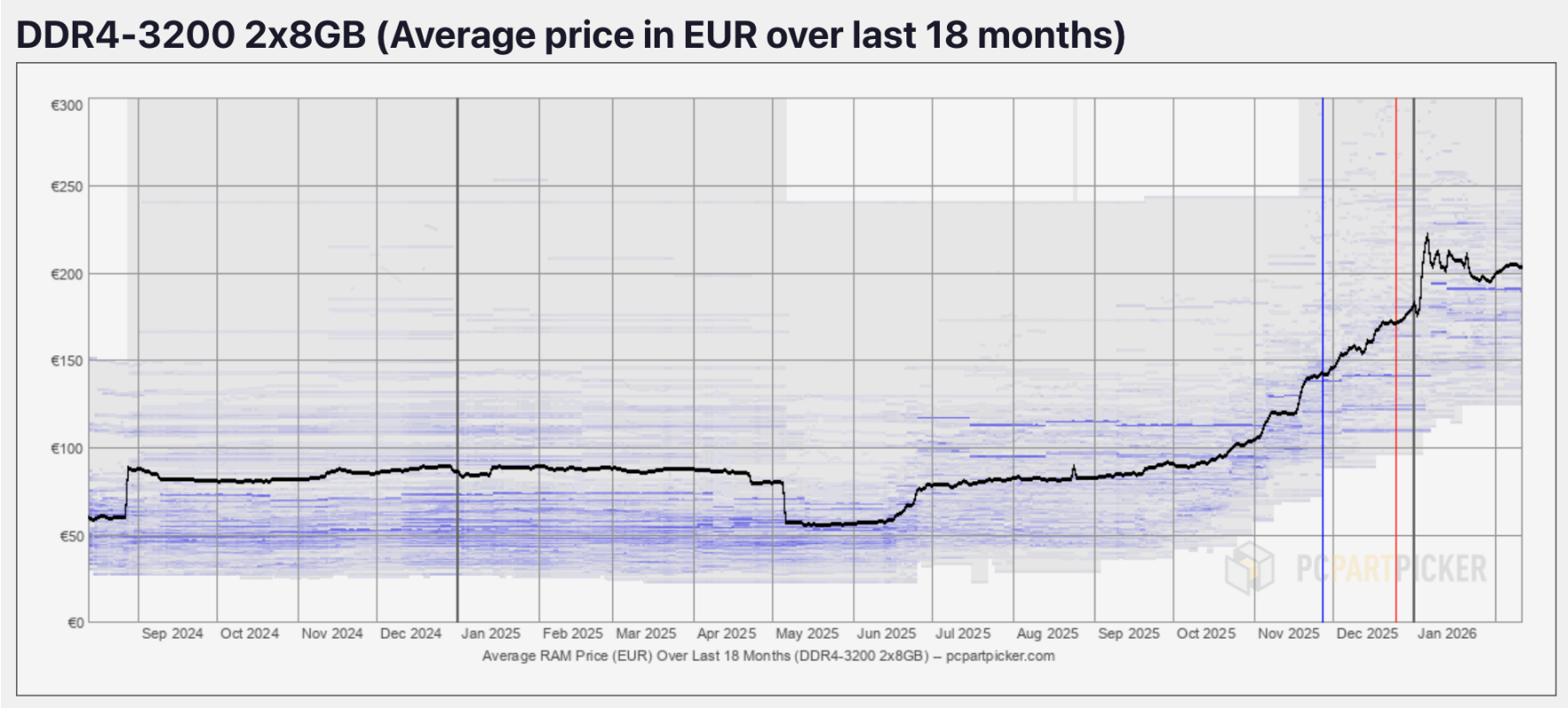
Het werkgeheugen van de computer

- Meerdere bits opslaan na elkaar
  - In reeksen van cellen: meestal 8 bits
  - Elke cel heeft haar eigen adres
  - Hoe zal dit worden voorgesteld? Ook een nummer, dus ook een bitpatroon
    - De cel op adres 0101 bevat waarde 00001000
    - Er is dus een verschil tussen celruimte en adresruimte
- Waar?
  - In de computer zelf (registers en “cache” vlak bij de CPU, zie infra.)
  - Random Access Memory (RAM)
  - Dynamisch RAM (DRAM of SDRAM)
    - Gebruik van flip-flops is verouderd
    - Verdere miniaturisatie en snellere responstijd door gebruik van nieuwere technologieën die kleine elektrische ladingen gebruiken
    - Refresh-schakelingen nodig die meerdere keren per seconde herladen
  - Double Data Rate Synchronous Dynamic Random-Access Memory (DDR SDRAM), van 1998 tot nu

“ DDR5 has about the same 14 ns latency as DDR4 and DDR3. DDR5 octuples the maximum dual in-line memory module (DIMM) capacity from 64 GB to 512 GB.



DDR5 also has higher frequencies than DDR4, up to around 64GB/s of bandwidth. Higher speeds have been achieved using liquid nitrogen. ”



- “DDR5 memory prices have exploded over the last trimester, with triple and, in some cases, quadruple increases. For instance, a conventional 32GB (2x16GB) DDR5 memory kit sold for around $100 to $200 in October 2025, but the same kit now starts at $350, if it’s even in stock.”“

Het werkgeheugen van de computer

Opslagcapaciteit

- In een moderne computer heeft zowel de CPU geheugen en is er RAM
  - Niet hetzelfde als je harde schijf of SSD schijf (“C:”)
  - Een addreseerbare groep van cellen
- Maar zelfs dan…
  - 1 bit: 0 of 1
  - 1 nibble: 4 bits (of 1 hexademicaal cijfer 0-F)
  - 1 byte: 8 bits (of 2 hexademicale cijfers)
  - Kilobyte (KB) = 2^{10} bytes = 1024 bytes = ± 1000 bytes
  - Megabyte (MB) = 2^{20} bytes = $2^{10}KB$ = ± 1 000 000 bytes
  - Gigabyte (GB) = 2^{30} bytes = $2^{20}KB$ = $2^{10}MB$ = ± 1 000 000 000 bytes
  - Terabyte (TB) = 2^{40} bytes = $2^{30}KB$ = $2^{20}MB$ = $2^{10}GB$ = ± 1 000 000 000 000 bytes
  - Nu ja, Kibi en Kilo…

Model	SanDisk Ultra (1.00)
Serial Number	4C531123501008106055
Size	16 GB (16 008 609 792 bytes)
Partitioning	GUID Partition Table

Model	USB SanDisk 3.2Gen1 (1.00)
Serial Number	0401786565474f13dd1c525451a0
Size	15 GB (15 401 484 288 bytes)
Partitioning	GUID Partition Table

> SanDisk defines 1 GB as 1,000,000,000 bytes. Operating Systems define 1 GB as 1,073,741,824 BYTES. A portion of the total capacity is used to store certain functions including optimizations of the memory that support performance and endurance and therefore is not available for user storage. This is disclosed on our packaging and marketing materials when you see the statement “Actual user storage less.”

Het werkgeheugen van de computer

Opslagcapaciteit

- Google: 30 PB processed daily
- Twitter: 340 million daily tweets, 7 TB added daily
- Facebook: 12 TB daily content added

- App op GSM: 1 – 100 MB
- MP3: 3 MB
- Gedownloade film: 700 MB – 2 GB

- Harde schijf: 0.5 – 2 TB (en meer)
- DVD: 4GB
- CD: 700MB
- Floppy disk: 1.44 MB

Massa-geheugen

- Werkgeheugen is typisch “vluchtig”
- Massa geheugen niet
  - Meestal grotere opslagcapaciteit (in totaal)
  - Meestal lagere kosten (per opgeslagen bit)
  - Vaak “offline”, dus vervoerbaar/uitwisselbaar
- Nadelen t.o.v. werkgeheugen
  - Als roterende schijven gebruikt worden, dan trager
  - Zoektijd (beweging arm met lees/schrijfkop)
  - Rotatievertraging (in functie van rotatiesnelheid)
  - Overdrachtssnelheid tijdens lezen/schrijven (in functie van rotatiesnelheid en densiteit gegevens op oppervlak)
  - Maar zelfs met niet-mechanische media (SSD’s), toch nog een snelheidsverlies

Massa-geheugen

Magnetisch geheugen

Massa-geheugen

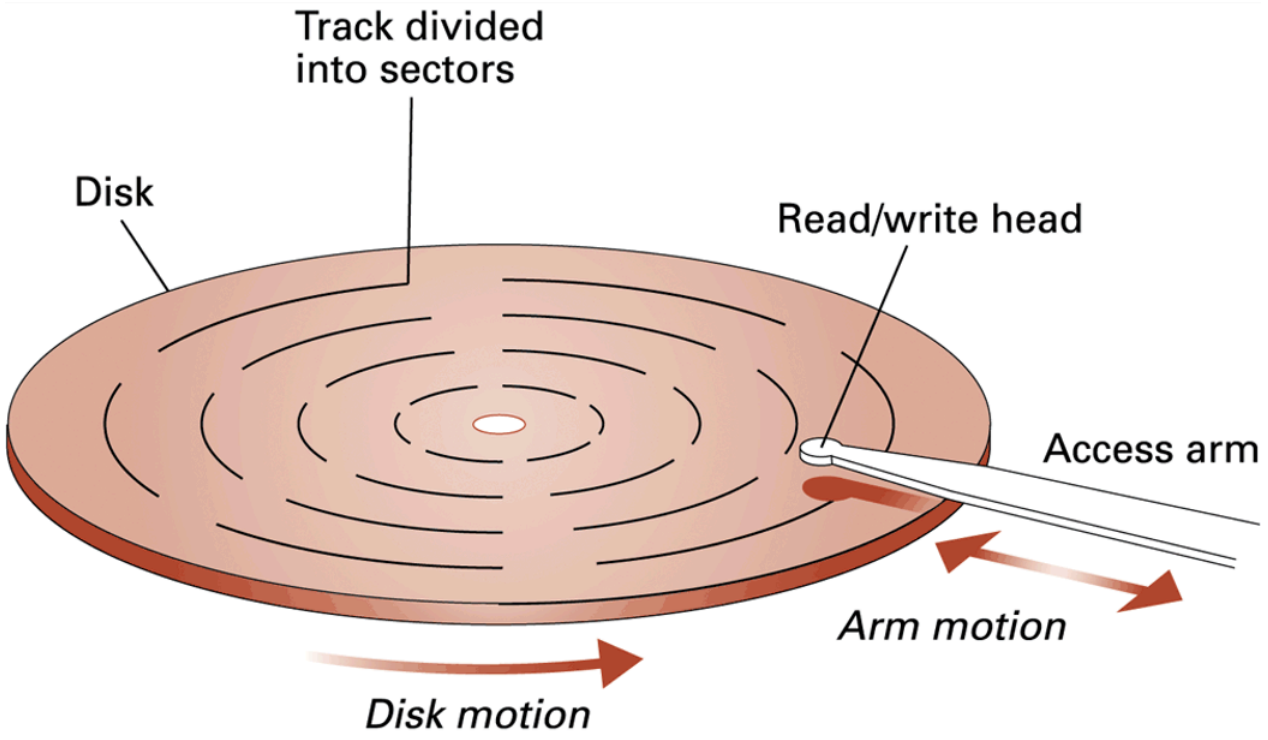
Magnetisch geheugen

Massa-geheugen

Magnetisch geheugen

Massa-geheugen

Magnetisch geheugen



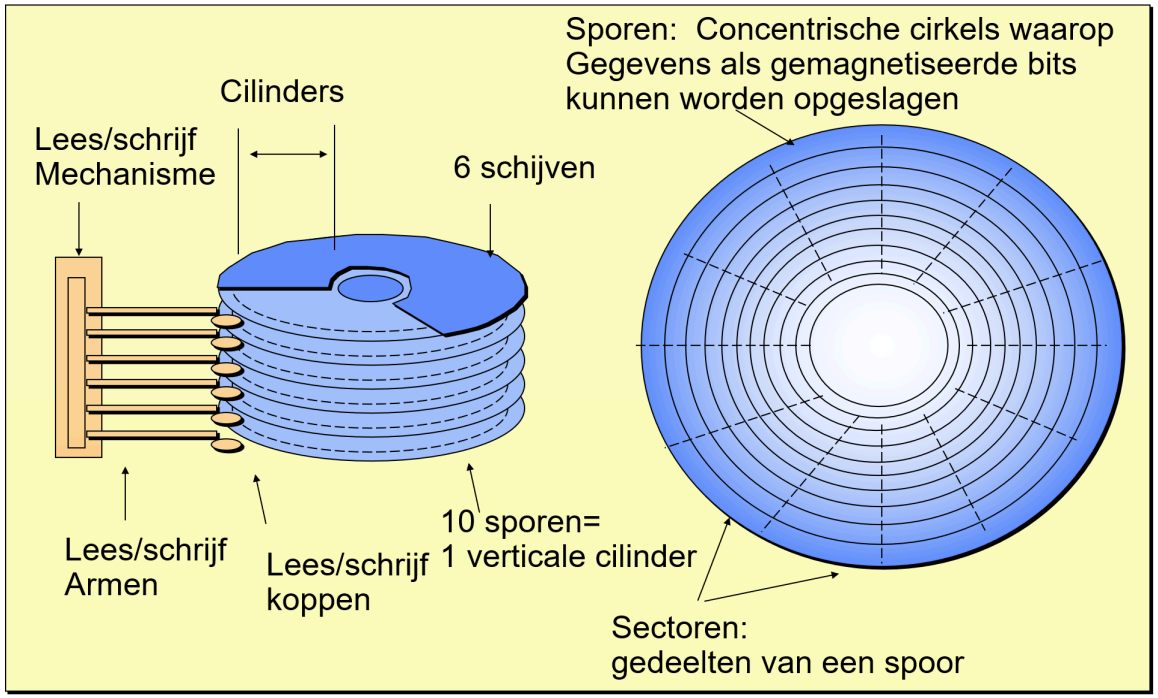
Massa-geheugen

Magnetisch geheugen

- De sporen op een magnetisch schijfgeheugen zijn concentrisch
- Het aanbrengen van de sporen en sectoren noemen we formatteren
- In eenvoudige magneetschijfgeheugensystemen bevat elk spoor eenzelfde aantal sectoren en elke sector hiervan bevat een lange reeks bits (een bitstroom) van eenzelfde grootte (meestal 0.5 KB tot enkele KB)
  - Dit betekent dat de densiteit van de gegevens toeneemt naarmate we het middelpunt van de schijf naderen
  - Hoge-capaciteits magneetschijfgeheugensystemen maken gebruik van zoned-bit recording, waarbij de schijf is ingedeeld in zones (typisch 10) van elk een aantal aangrenzende sporen. In een buitenliggende zone staan er meer sectoren op elk spoor dan in een binnenliggende zone, waardoor het totale schijfoppervlak beter benut wordt

Massa-geheugen

Magnetisch geheugen



Massa-geheugen

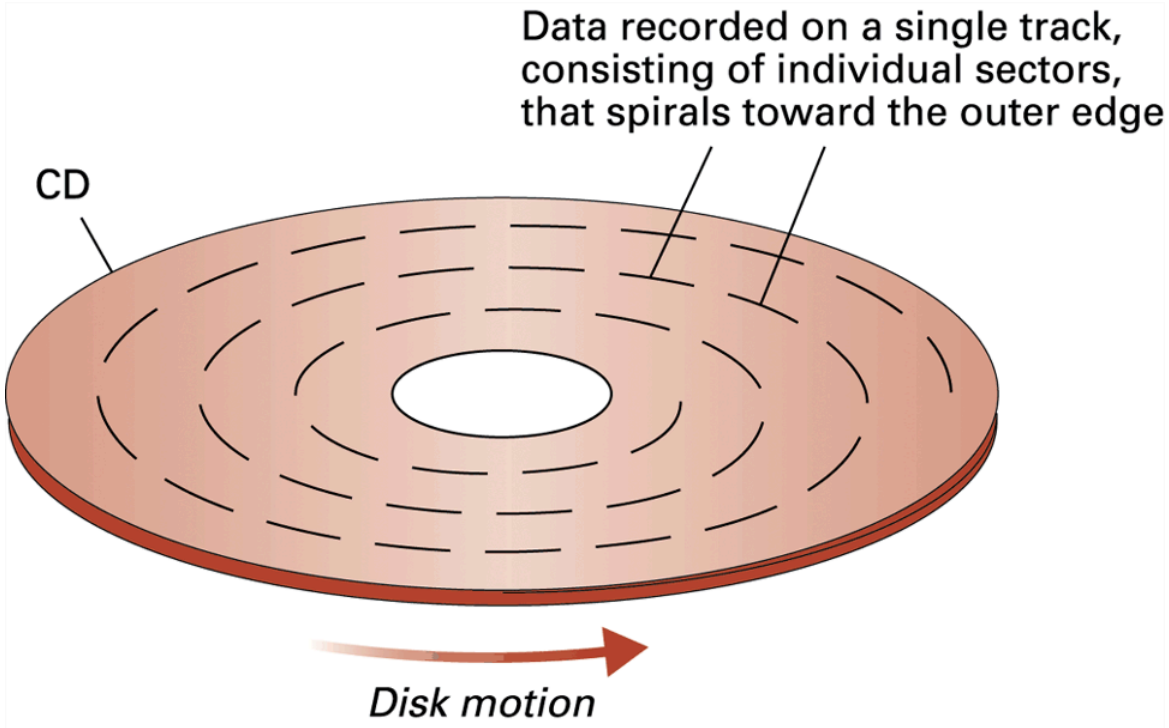
Magnetisch geheugen

- Magneetschijfgeheugensystemen bevatten meestal meerdere schijven op een centrale as gemonteerd
- Voor elk bruikbaar schijfoppervlak is er een lees/schrijfkop die gemonteerd is op een lees/schrijfarm
  - Deze bewegen samen over en tussen de schijfoppervlakken
  - De sporen die op die manier door de lees/schrijfkoppen tesamen bereikt worden noemt men een cilinder
- Opslagcapaciteiten van zijn te situeren in termen van tientallen/honderden GB tot enkele TB
- De rotatiesnelheid bedraagt meerdere duizenden omwentelingen per minuut (RPM: rotations per minute), waardoor overdrachtssnelheden gemeten worden in termen van MB per seconde


HARDE SCHIJF
BATTERIJ
RAM
TOSHIBA
MK5065GSXF
HDD2J62 P TV02 T
Warning:
Do not remove
the battery.
Avertissement :
ne pas retirer
la batterie.
Achtung:
Entfernen Sie die
Batterie nicht.
警告：
バッテリーを
取り外さないで
ください。

Massa-geheugen

Optisch geheugen



Massa-geheugen

Optisch geheugen

- De sectoren op b.v. een Compact Disk (CD) hebben een capaciteit van 2 KB
  - De opslagdensiteit op het spoor dat zich van binnen naar buiten afrolt is overal hetzelfde, dus de rotatiesnelheid van de schijf moet aangepast worden naarmate dichter of verder van het middelpunt gegevens geschreven/gelezen worden
  - De toegangstijden t.o.v. van deze van magnetische schijfgeheugens kunnen korter of langer zijn
    - Meestal worden betere prestaties gehaald als een lange, continue reeks gegevens benaderd wordt dan wanneer de laserstraal voortdurend van plaats moet veranderen (in dat geval zijn magnetische schijfgeheugens sneller dan optische)
- De opslagcapaciteit van een CD situeert zich rond de 700 MB
- Digital Versatile Disks (DVD) maken gebruik van meerdere semi-transparante lagen en halen opslagcapaciteiten van meerdere GB
- Blu-ray Disks (BD) maken gebruik van een andere lasertechnologie waardoor de focus van de laserstraal preciezer is en er dus meer gegevens op de schijf opgeslagen kunnen worden (ongeveer vijf keer zoveel als bij DVD)

Massa-geheugen

Flashgeheugen

- Massageheugen zonder mechanische (= bewegende) component
  - Niet meer nadeel van tragere snelheid
- Bit = elektron in minuscuul vakje siliciumdioxide
  - Niet vluchtig voor vele jaren
  - Robuust tegen schokken
  - Random access
  - Maar, vaak wijzigen beschadigt de vakjes
- Flash drives (vb. USB stick, Solid State Disk (SSD)) hebben capaciteiten tot enkele honderden/duizenden GB (512 GB voor USB, 2 TB voor SSD)
- SD, SDHC en SDXC geheugenkaarten hebben de grootte van een kleine postzegel met capaciteiten van respectievelijk 2GB, 32 GB tot over een TB
- Vandaag: SSD meest courant als massa-geheugen in consumenten-pc's (groot verschil in termen van "gevoel van snelheid")


V-NAND SSD
970 EVO
NVMe M.2
SAMSUNG ELECTRONICS CO., LTD.
500GB
SAMSUNG

Gegevens voorstellen met bitpatronen

- We kunnen nu uitgaan van geheugen om bits (lange reeksen) in op te slaan
  - Maar hoe slaan we tekst, nummers, … op?

Gegevens voorstellen met bitpatronen

Tekst

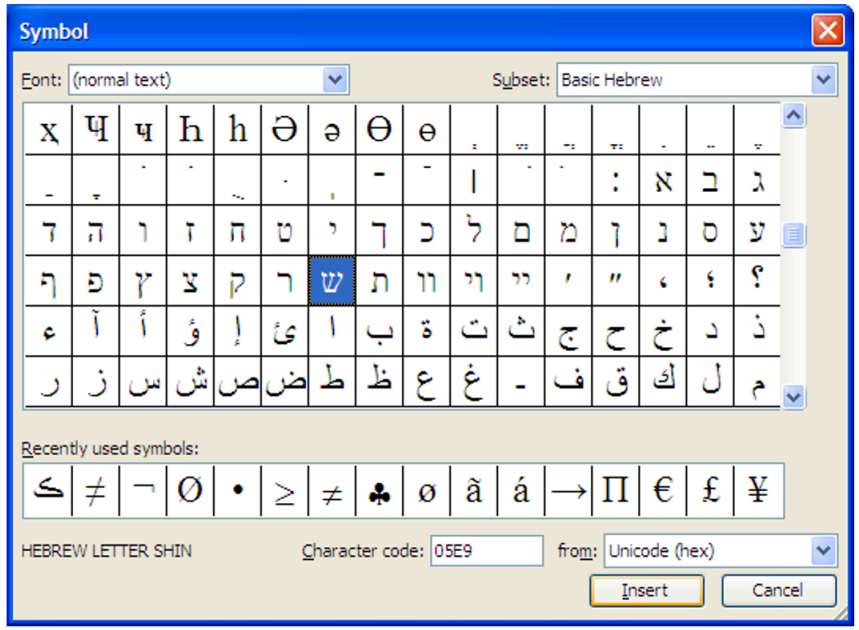
- Code met uniek bitpatroon per tekstsymbool
  - ASCII
    8 bits → 2^8 = 256 unieke patronen
  - Unicode
    16 bits → 2^{16} = 65536 unieke patronen
  - Text file
    Bestand met een lange reeks symbolen gecodeerd met ASCII of Unicode

Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex
line feed	00001010	0A	>	00111110	3E	^	01011110	5E
carriage return	00001011	0B	?	00111111	3F	_	01011111	5F
space	00100000	20	@	01000000	40	`	01100000	60
!	00100001	21	A	01000001	41	a	01100001	61
"	00100010	22	B	01000010	42	b	01100010	62
#	00100011	23	C	01000011	43	c	01100011	63
$	00100100	24	D	01000100	44	d	01100100	64
%	00100101	25	E	01000101	45	e	01100101	65
&	00100110	26	F	01000110	46	f	01100110	66
'	00100111	27	G	01000111	47	g	01100111	67
(	00101000	28	H	01001000	48	h	01101000	68
)	00101001	29	I	01001001	49	i	01101001	69
*	00101010	2A	J	01001010	4A	j	01101010	6A
+	00101011	2B	K	01001011	4B	k	01101011	6B
,	00101100	2C	L	01001100	4C	l	01101100	6C
.	00101101	2D	M	01001101	4D	m	01101101	6D
/	00101110	2E	N	01001110	4E	n	01101110	6E
0	00101111	2F	O	01001111	4F	o	01101111	6F
1	00110000	30	P	01010000	50	p	01110000	70
2	00110001	31	Q	01010001	51	q	01110001	71
3	00110010	32	R	01010010	52	r	01110010	72
4	00110011	33	S	01010011	53	s	01110011	73
5	00110100	34	T	01010100	54	t	01110100	74
6	00110101	35	U	01010101	55	u	01110101	75
7	00110110	36	V	01010110	56	v	01110110	76
8	00110111	37	W	01010111	57	w	01110111	77
9	00111000	38	X	01011000	58	x	01111000	78
:	00111001	39	Y	01011001	59	y	01111001	79
;	00111010	3A	Z	01011010	5A	z	01111010	7A
<	00111011	3B	[	01011011	5B	{	01111011	7B
=	00111100	3C	\	01011100	5C	|	01111100	7C
	00111101	3D	]	01011101	5D	}	01111101	7D

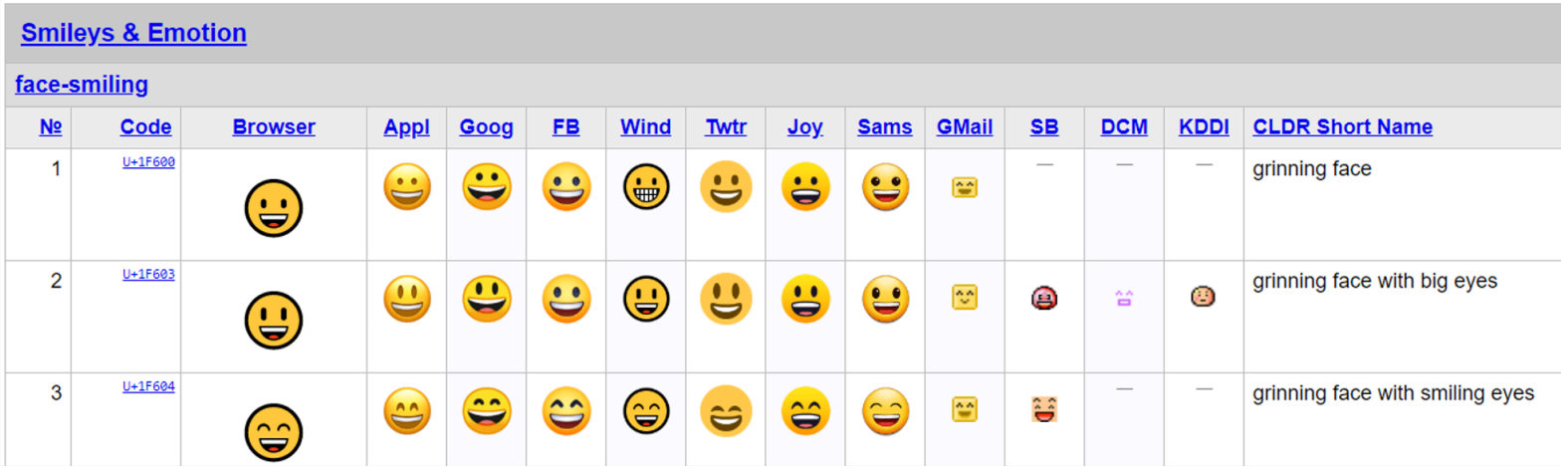
ASCII

01001000	01100101	01101100	01101100	01101111	00101110
H	**e**	**l**	**l**	**o**	**.**

Unicode



Karakter
ש
Unicode (hexadecimaal)
05E9
Unicode (binair)
0000010111101001



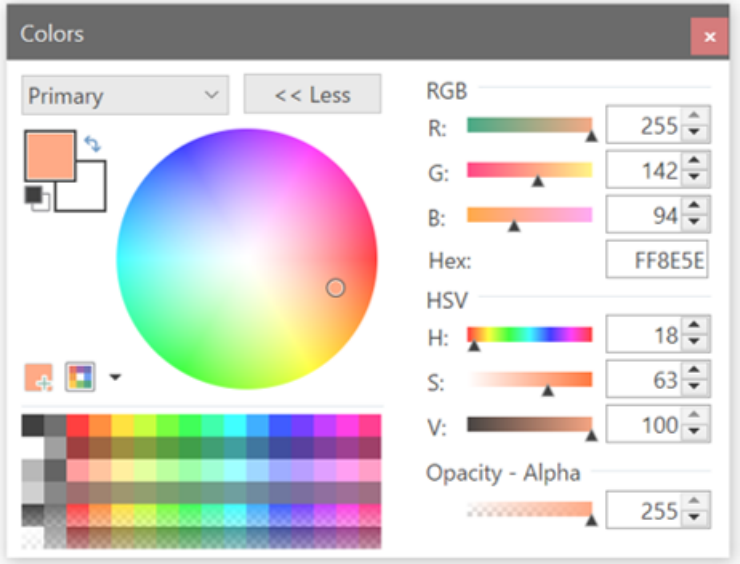
Smileys & Emotion														
face-smiling														
№	Code	Browser	Appl	Goog	FB	Wind	Twtr	Joy	Sams	GMail	SB	DCM	KDDI	CLDR Short Name
1	U+1F600										—	—	—	grinning face
2	U+1F603													grinning face with big eyes
3	U+1F604											—	—	grinning face with smiling eyes

https://unicode.org/emoji/charts/full-emoji-list.html

Gegevens voorstellen met bitpatronen

Afbeeldingen

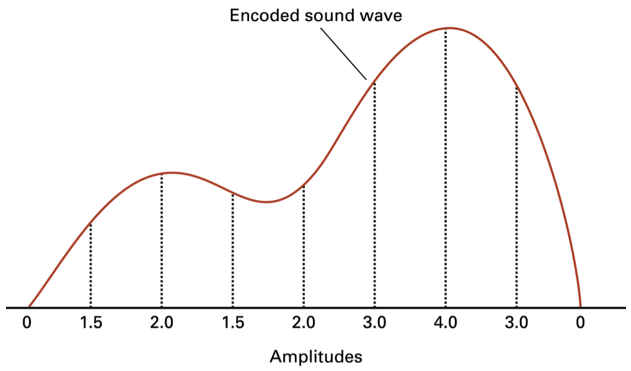
- Een afbeelding kan gezien worden als een verzameling puntjes (pixels)
  - Bitmap
    - Reeks bits die de pixels van een afbeelding codeert
    - Geen compressie
    - Zwart-wit: 1 bit per pixel
    - Kleur: meerdere bits per pixel
    - Vaak: 3 bytes per pixel
      - Kleurintensiteit voor rood, groen en blauw

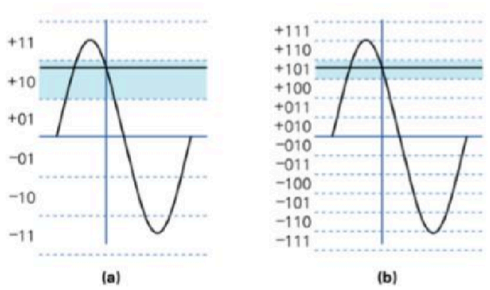


Gegevens voorstellen met bitpatronen

Geluid

- Sampling
  - Met een frequentie van 44100 keer per seconde (44KHz) wordt de amplitude van de geluidsgolf gemeten
  - Deze waarde wordt opgeslagen
    - 16 bits per sample
    - 32 bits per sample (voor stereo)





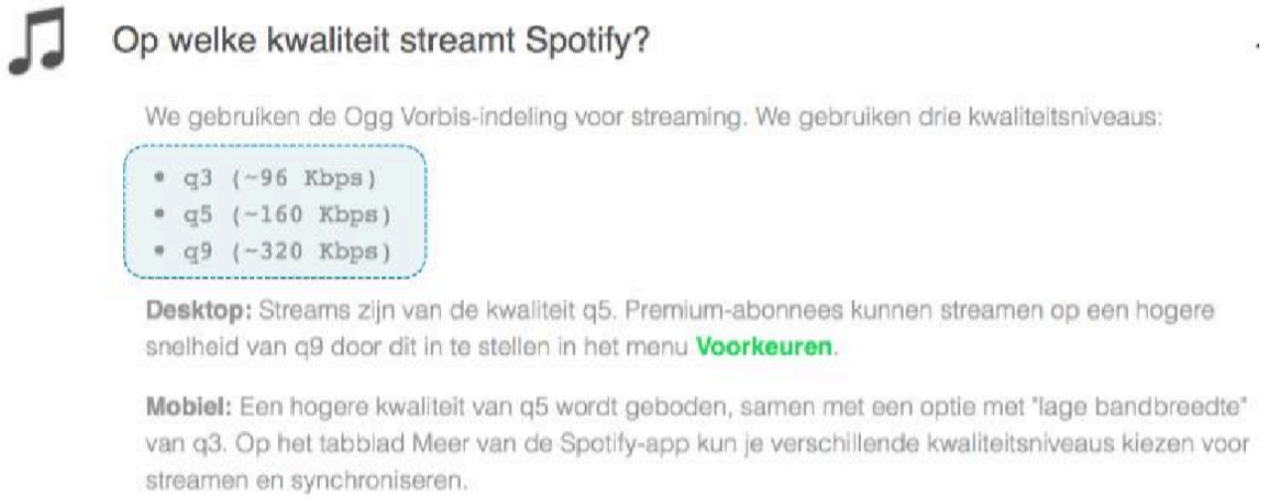
Gegevens voorstellen met bitpatronen

Geluid

- Bij CD-opnames worden 16 bits gebruikt, zodat 2^{16} = 65536 niveaus opgenomen kunnen worden
  - 2^{15} positieve en evenveel negatieve waarden
  - Geen compressie
  - 1 minuut digitale audio vereist 60 seconden x 44.100 samples x 16 bits x 2 (stereo) = 84.672.000 bits = 10.584.000 bytes = 10 MB
  - Een CD van ongeveer 1 uur vraag dus minstens 635 MB opslagruimte

Gegevens voorstellen met bitpatronen

Geluid



Op welke kwaliteit streamt Spotify?

We gebruiken de Ogg Vorbis-indeling voor streaming. We gebruiken drie kwaliteitsniveaus:

- q3 (~96 Kbps)
- q5 (~160 Kbps)
- q9 (~320 Kbps)

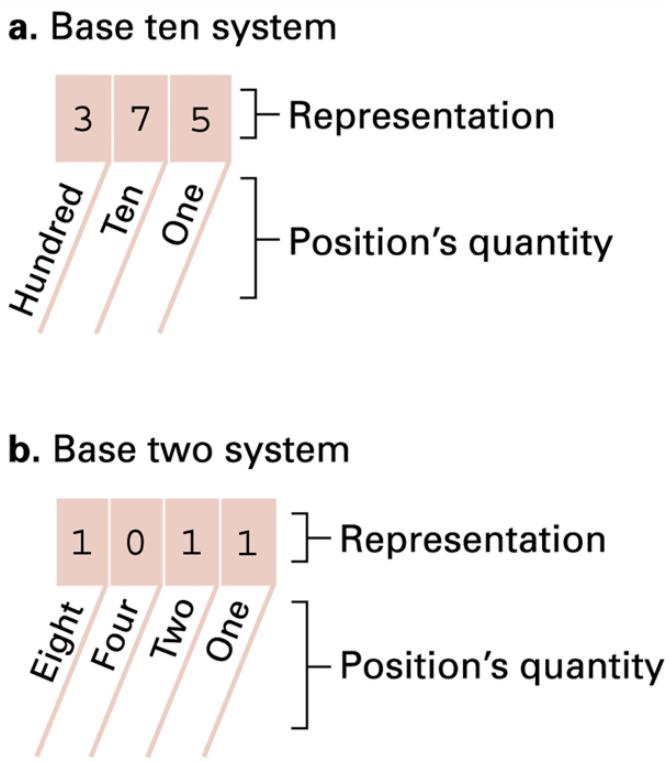
Desktop: Streams zijn van de kwaliteit q5. Premium-abonnees kunnen streamen op een hogere snelheid van q9 door dit in te stellen in het menu **Voorkeuren**.

Mobiel: Een hogere kwaliteit van q5 wordt geboden, samen met een optie met "lage bandbreedte" van q3. Op het tabblad Meer van de Spotify-app kun je verschillende kwaliteitsniveaus kiezen voor streamen en synchroniseren.

- 96 Kbps (≠ KB/s) = 96 Kbits per seconde = 12 KB / seconde
  - Wel compressie!

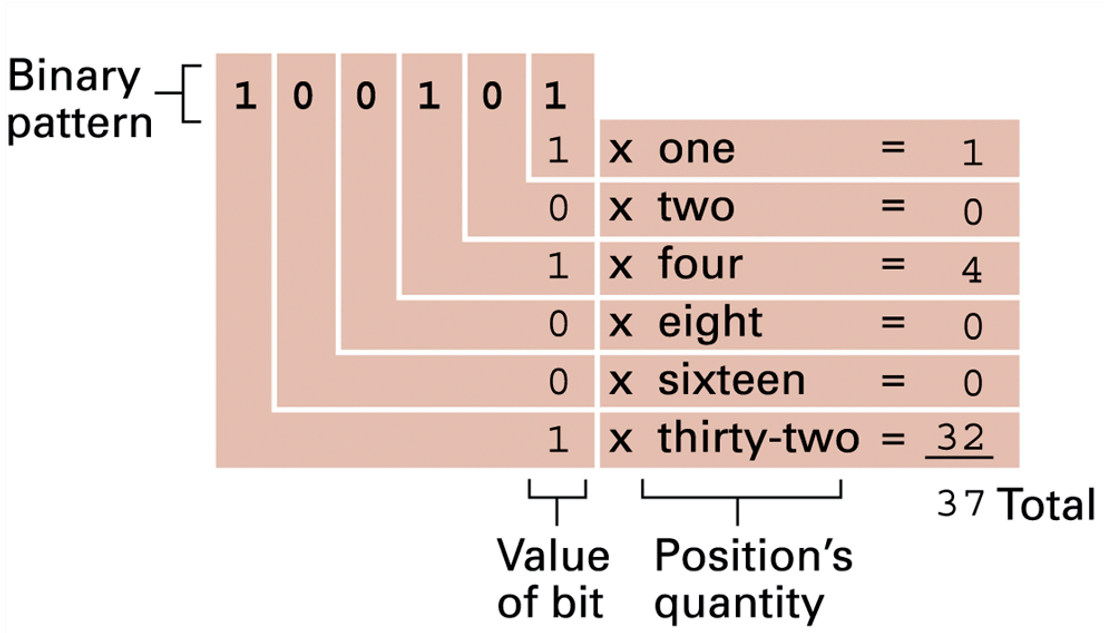
Het binair talstelsel

Numerieke gegevens voorstellen met bitpatronen…



Het binair talstelsel

Van binair naar decimaal



- Eenvoudig voor positieve gehele getallen

Het binair talstelsel

Van decimaal naar binair

1. Deel het getal (gehele deling) door 2 en noteer de rest
2. Zolang het quotient niet gelijk is aan 0, blijf doordelen door 2 en de rest noteren
3. Eenmaal de quotient gelijk is aan 0 kan de binaire voorstelling afgeleid worden door de resten van rechts naar links neer te schrijven

Het binair talstelsel

Van decimaal naar binair: voorbeeld

13 / 2 = 6 (rest 1)

6 / 2 = 3 (rest 0)

3 / 2 = 1 (rest 1)

1 / 2 = 0 (rest 1)

→ 1101

16	8	4	2	1
0				
	Past 8 in 13? Ja 5 blijft over			
		Past 4 in 5? Ja 1 blijft over		
			Past 2 in 1? Nee 1 blijft over	
				Past 1 in 1? Ja 0 blijft over
0	1	1	0	1

Het binair talstelsel

Van decimaal naar binair: voorbeeld

```
# Decimaal naar binair getal

decimaal = int(input("Geef een positief decimaal getal in: "))

binair   = ''
quotient = decimaal

while quotient != 0:
  quotient = decimaal // 2
  rest     = decimaal % 2
  decimaal = quotient
  binair   = str(rest) + binair

print("Het overeenkomstig binair getal is:", binair)
```

→ https://repl.it/@Seppevanden/InformaticaUGent

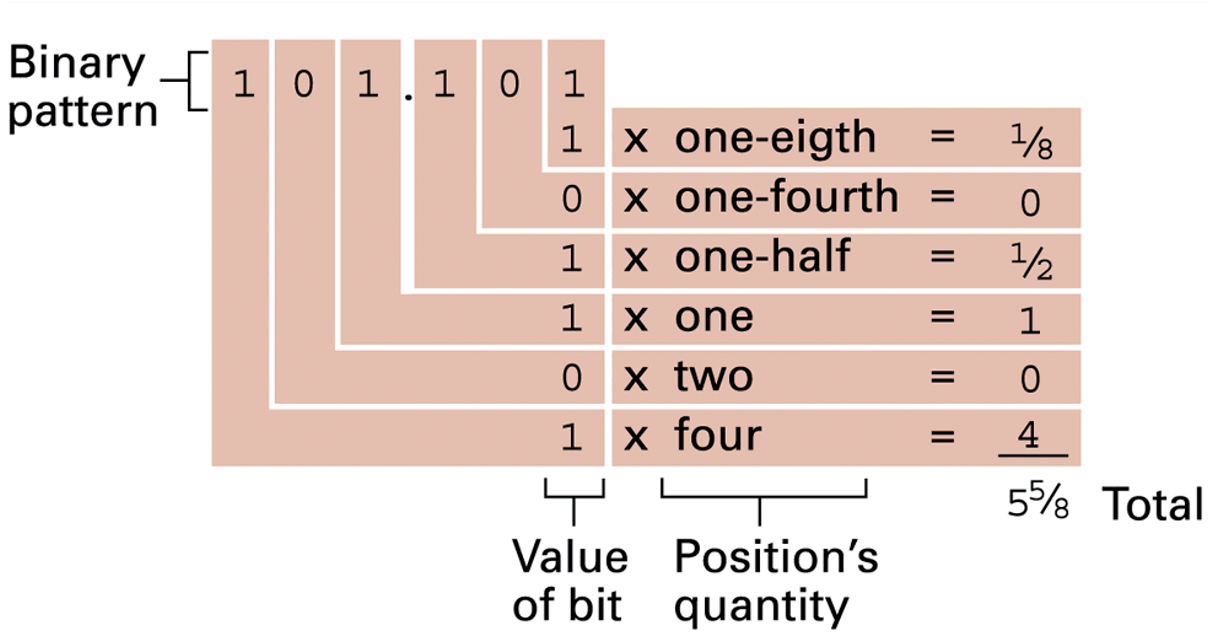
Het binair talstelsel

Optellen van binaire getallen

```
  0      1      0      1
+ 0    + 0    + 1    + 1
---    ---    ---    ---
  0      1      1     10
```

Het binair talstelsel

Binaire breuken / fracties



- Het punt dat het geheel deel van het binair getal scheidt van het breukdeel wordt het radixpunt genoemd

Het binair talstelsel

Gehele getallen

- Gehele getallen kunnen positief of negatief zijn
- Conventie nodig voor het voorstellen en opslaan van het teken
  - `-1011` zou niet werken, want we kunnen enkel 0 en 1 waardes opslaan
  - 2-complementnotatie
    - Gebruikt een vast aantal bits, b.v. 32 bits → 2^{32} waarden in het bereik [-2147483648, 2147483648[
  - Excess-notatie
    - Variant waarbij het tekenbit omgekeerd wordt

Het binair talstelsel

Gehele getallen: 2-complementnotatie

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

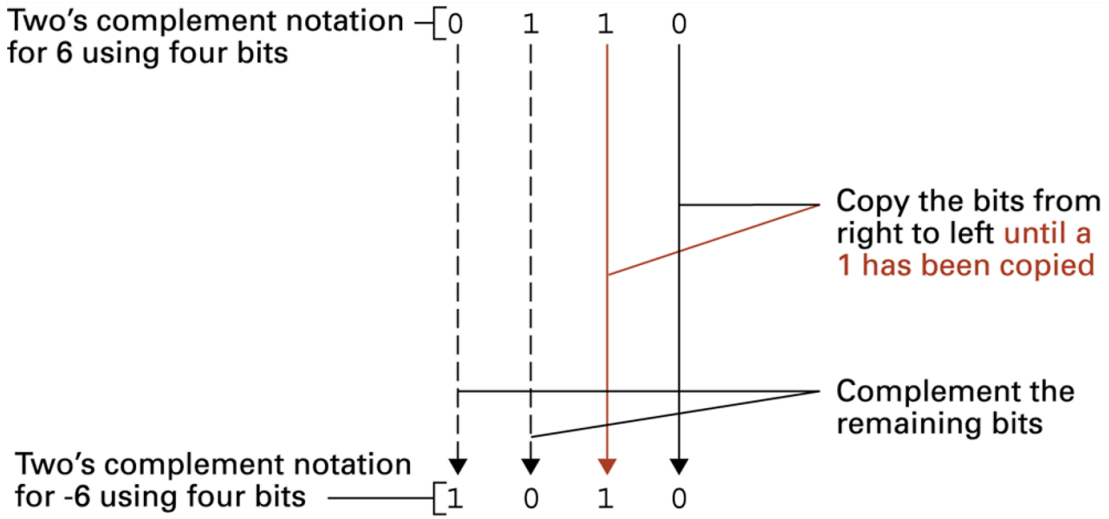
b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

- Meest significante bit geeft het teken weer
- Opgelet: -1 ≠ 101

Het binair talstelsel

Gehele getallen: 2-complementnotatie



- Maar negatief binair getal kan wel makkelijk afgeleid worden van positief
  - Ga van rechts naar links tot je een 1 tegenkomt en complementeer de rest
  - Of alle waardes complementeren en er dan 1 bijtellen
    - -1 en complementeren om van negatief naar positief te gaan (zie volgende slide)
  - Vast aantal bits!

Het binair talstelsel

Gehele getallen: 2-complementnotatie

- -1 en complementeren om van negatief naar positief te gaan

```
1010        −6
            −1
```

- Ofwel: -7 voorstellen als complement:

```
0111        +7
1000        complementeren
1001        1 bijtellen geeft −7
```

- Ofwel: -1 = + -1

```
0001        +1
1110        complementeren
1111        1 bijtellen geeft −1

     1010   −6
+    1111   + −1
= (1)1001   −7
```

- En dan complementeren…

```
1001        −7
0110        complementeren
            = +6
```

Het binair talstelsel

Gehele getallen: 2-complementnotatie

Problem in base ten		Problem in two's complement		Answer in base ten
3 + 2	→	0011 + 0010 = 0101	→	5
−3 + −2	→	1101 + 1110 = 1011	→	−5
7 + −5	→	0111 + 1011 = 0010	→	2

- 2-complement optellen en aftrekken (optelling met negatief getal)
- Geen aparte elektronische schakeling nodig voor aftrekking
  - Het volstaat wanneer schakelingen voorzien zijn voor optelling en voor het negatief maken van een getal

Het binair talstelsel

Overflow

- Cellen in een geheugen hebben een vast aantal tekens
- Met **overflow** wordt bedoelt dat een voor te stellen getal buiten het bereik valt
  - Bij negatieve getallen → tekenbit = 0 en overdracht van 1 naar niet-bestaande volgende positie
  - Bij positieve getallen → tekenbit = 1

Het binair talstelsel

Gehele getallen: excess-notatie

- Hier is het tekenbit 0 voor negatieve getallen
- Opnieuw met assumptie van vast aantal bits

```
000     0           -4
001     1           -3
010     2           -2
011     3           -1
100     4            0  -> halfweg in de lijst
101     5            1
110     6            2
111     7            3
```

Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	-1
010	-2
001	-3
000	-4

- Merk op dat de waarde gelijk is aan de positieve gehele voorstelling, -4
  - Meest significante bit = 1, rest 0
  - “Excess-4 notatie”

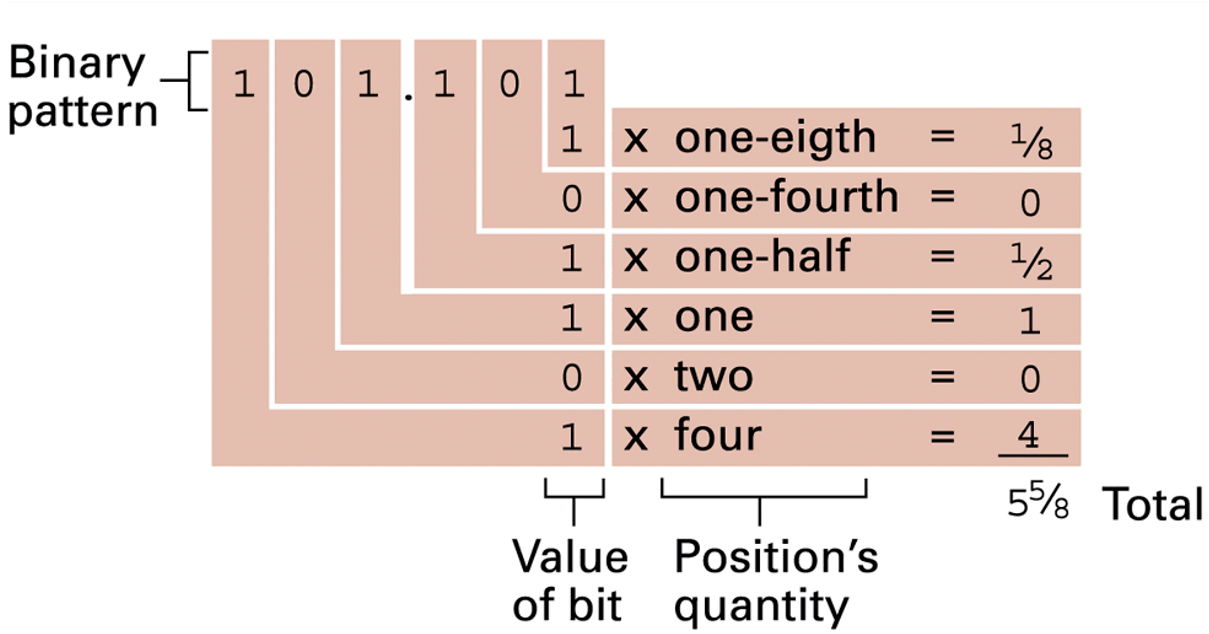
Het binair talstelsel

Gehele getallen: excess-notatie

Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

Het binair talstelsel

Reële getallen

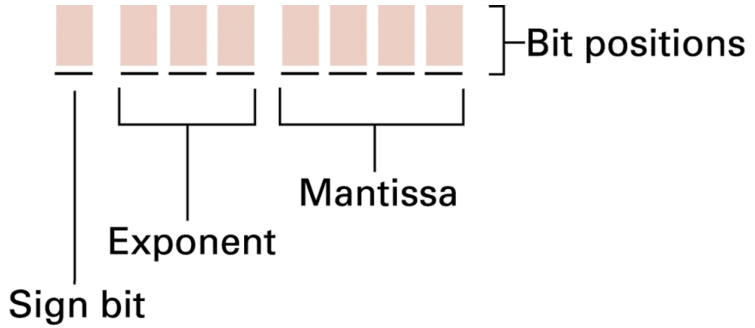


- `101.101` werkt ook niet helemaal, want hoe slaan we de komma op?
  - We kunnen enkel 0 en 1 waardes opslaan

Het binair talstelsel

Reële getallen: floating-pointnotatie

- Vast aantal bits wordt ingedeeld in drie delen:
  - Mantisse
    - Bitpatroon voor geheel deel en breukdeel
    - Begint altijd met 1 → genormaliseerde vorm
  - Exponent
    - Positie van het radixpunt (in excess-notatie)
  - Tekenbit (zoals bij 2-complementnotatie)



Het binair talstelsel

Reële getallen: floating-pointnotatie

Voorbeeld 1: welk reëel getal wordt voorgesteld door `10111001`?

- Mantisse = `1001`
- Exponent = `011`
  - Gewoon binair geeft 3 in decimaal, maar we werken met excess notatie dus 3 - 4 = -1
  - Radixpunt schuift in de mantisse één positie naar links
- Mantisse wordt dus `.1001` → `.01001` wat gelijk is aan 1/4 + 1/32 = 9/32
- Tekenbit = `1`
- Dus -9/32 = -0.28125

Het binair talstelsel

Reële getallen: floating-pointnotatie

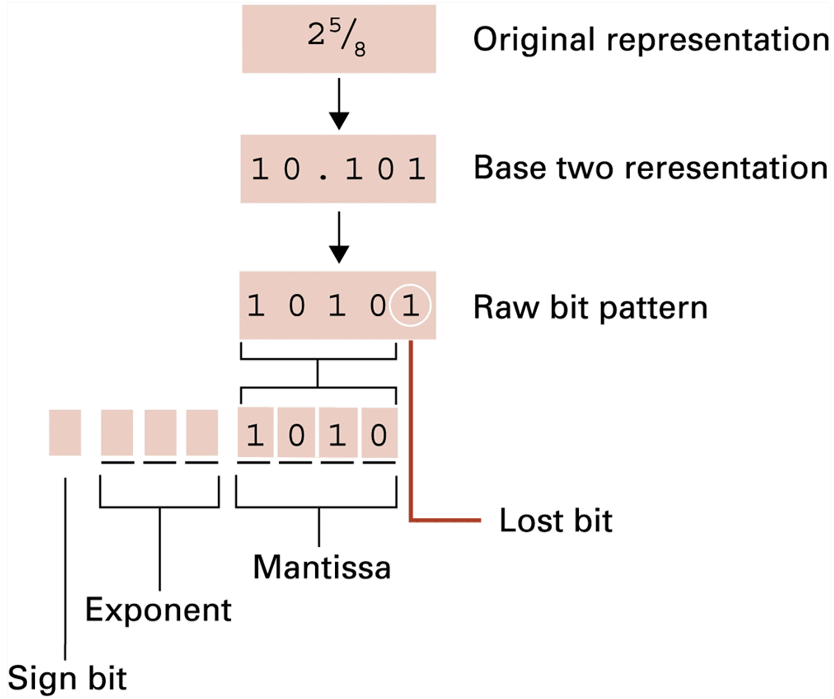
Voorbeeld 2: zet 2,75 om naar floating-pointnotatie met acht bits

- In breukvorm: `10.11`
- We kopieren dit naar de mantisse, beginnende met de meest linkse 1
  - Mantisse = `1011`
- Hoeveel plaatsen moeten we verschuiven om van `.1011` naar `10.11` te gaan?
  - 2 naar rechts, dus +2 = `010` in gewoon binair
  - Maar opgelet, excess-notatie, 2 + 4 = 6 = `110` in gewoon binair = +2 in excess-notatie
- Tekenbit = `0` (positief)
- Dus `01101011`

Het binair talstelsel

Reële getallen: afkapfouten

- Exponent wordt 110 en tekenbit wordt 0 dus gehele bitpatroon wordt `01101010`
- Als je dit terug naar decimaal omzet krijg je 2,5
- Er is dus een afkapping van de minst significante bit gebeurd waardoor het getal 2,625 is afgerond naar 2,5 (afrondingsfout = 0,125)
- Om het effect van dergelijke fouten te reduceren worden 32 bits (Single Precision Floating Point: 1 tekenbit, 8 exponentbits, 23 mantissebits) of 64 bits (Double Precision Floating Point) gebruikt voor het opslaan van getallen in floating-pointnotatie
- De acht-bitindeling die hier in de cursus gebruikt wordt dient vooral als voorbeeld (ook de reden waarom oude 8-bit computers vaak geen ondersteuning hadden voor decimalen!)



```
00001000  =  0.03125
00001001  =  0.03515625
00001010  =  0.0390625
00001011  =  0.04296875
00001100  =  0.046875
00001101  =  0.05078125
00001110  =  0.0546875
00001111  =  0.05859375
00011000  =  0.0625
00011001  =  0.0703125
00011010  =  0.078125
00011011  =  0.0859375
00011100  =  0.09375
00011101  =  0.1015625
00011110  =  0.109375
00011111  =  0.1171875
00101000  =  0.125
00101001  =  0.140625
00101010  =  0.15625
00101011  =  0.171875
00101100  =  0.1875
00101101  =  0.203125
00101110  =  0.21875
00101111  =  0.234375
00111000  =  0.25
00111001  =  0.28125
00111010  =  0.3125
00111011  =  0.34375
00111100  =  0.375
00111101  =  0.40625
00111110  =  0.4375
00111111  =  0.46875
01001000  =  0.5
```

```
01001001  =  0.5625
01001010  =  0.625
01001011  =  0.6875
01001100  =  0.75
01001101  =  0.8125
01001110  =  0.875
01001111  =  0.9375
01011000  =  1
01011001  =  1.125
01011010  =  1.25
01011011  =  1.375
01011100  =  1.5
01011101  =  1.625
01011110  =  1.75
01011111  =  1.875
01101000  =  2
01101001  =  2.25
01101010  =  2.5
01101011  =  2.75
01101100  =  3
01101101  =  3.25
01101110  =  3.5
01101111  =  3.75
01111000  =  4
01111001  =  4.5
01111010  =  5
01111011  =  5.5
01111100  =  6
01111101  =  6.5
01111110  =  7
01111111  =  7.5
```

```
10001000  =  -0.03125
10001001  =  -0.03515625
10001010  =  -0.0390625
10001011  =  -0.04296875
10001100  =  -0.046875
10001101  =  -0.05078125
10001110  =  -0.0546875
10001111  =  -0.05859375
10011000  =  -0.0625
10011001  =  -0.0703125
10011010  =  -0.078125
10011011  =  -0.0859375
10011100  =  -0.09375
10011101  =  -0.1015625
10011110  =  -0.109375
10011111  =  -0.1171875
10101000  =  -0.125
10101001  =  -0.140625
10101010  =  -0.15625
10101011  =  -0.171875
10101100  =  -0.1875
10101101  =  -0.203125
10101110  =  -0.21875
10101111  =  -0.234375
10111000  =  -0.25
10111001  =  -0.28125
10111010  =  -0.3125
10111011  =  -0.34375
10111100  =  -0.375
10111101  =  -0.40625
10111110  =  -0.4375
10111111  =  -0.46875
11001000  =  -0.5
```

```
11001001  =  -0.5625
11001010  =  -0.625
11001011  =  -0.6875
11001100  =  -0.75
11001101  =  -0.8125
11001110  =  -0.875
11001111  =  -0.9375
11011000  =  -1
11011001  =  -1.125
11011010  =  -1.25
11011011  =  -1.375
11011100  =  -1.5
11011101  =  -1.625
11011110  =  -1.75
11011111  =  -1.875
11101000  =  -2
11101001  =  -2.25
11101010  =  -2.5
11101011  =  -2.75
11101100  =  -3
11101101  =  -3.25
11101110  =  -3.5
11101111  =  -3.75
11111000  =  -4
11111001  =  -4.5
11111010  =  -5
11111011  =  -5.5
11111100  =  -6
11111101  =  -6.5
11111110  =  -7
11111111  =  -7.5
```

Het binair talstelsel

Reële getallen: afkapfouten

```
print(.1 + .2)
```

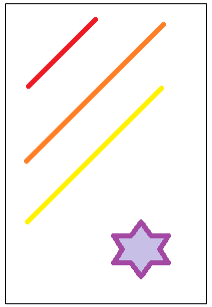
```
0.30000000000000004
```

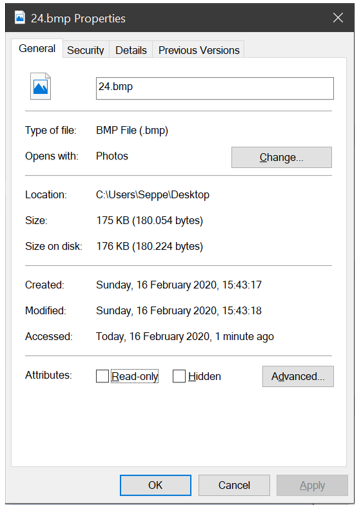
→ https://0.30000000000000004.com/

Het binair talstelsel

Compressie

- 200 x 300 pixels
- 24bit kleuren
- BMP: 200 x 300 x 24 = 1440000 bits = 180000 bytes

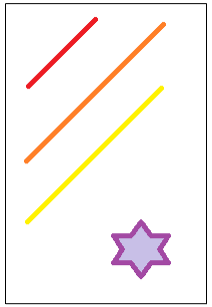


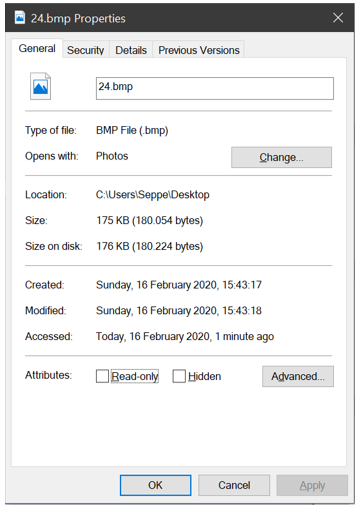


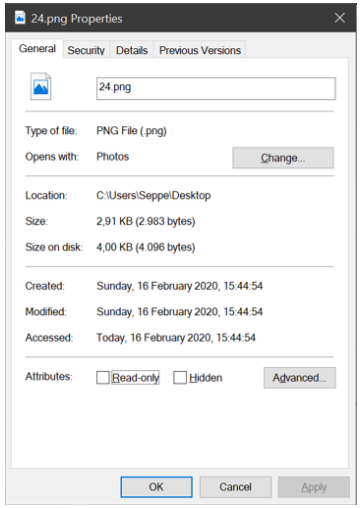
Het binair talstelsel

Compressie

- 200 x 300 pixels
- 24bit kleuren
- PNG?







Het binair talstelsel

Compressie

- HD-film:
  - Een beeldje = 1920 x 1080 pixels
  - 24bit kleuren per pixel
  - 30 beeldjes per seconde
  - 1u film = (60 * 60) * 30 * 1920 * 1080 * 24
    = 5374771200000 bits
    = 671846400000 bytes
    = 671846 Megabytes = 671 Gigabytes
- Op Blu-ray past een film van meerdere uren op 50GB?

Het binair talstelsel

Compressie

- Compressie: benodigde opslagruimte verkleinen
  - Lossless: zonder verlies van informatie
    - B.v.: ZIP-files, FLAC, PNG
  - Lossy: met verlies van informatie
    - B.v.: MP3, MP4

Het binair talstelsel

Compressie

- Simpele techniek: “run-length encoding”:
  - xxxxxxxyyxxyyy
  - 7x2y2x3y
- Dictionary encoding:
  - Maak eerst een cataloog aan van veel voorkomende langere patronen en verwijs daarnaar

Het binair talstelsel

Compressie

- Lossy compressive maakt vaak gebruik van inzichten in hoe mensen horen en zien
  - B.v. beeld (JPEG): mensen zien slechter blauw dan andere kleuren
  - Geluid: hoge frequenties kunnen meer gecomprimeerd worden
  - Video: dupliceer geen delen van de afbeelding die min of meer gelijk blijven

Het binair talstelsel

Compressie

- Afweging grootte en kwaliteit

Het binair talstelsel

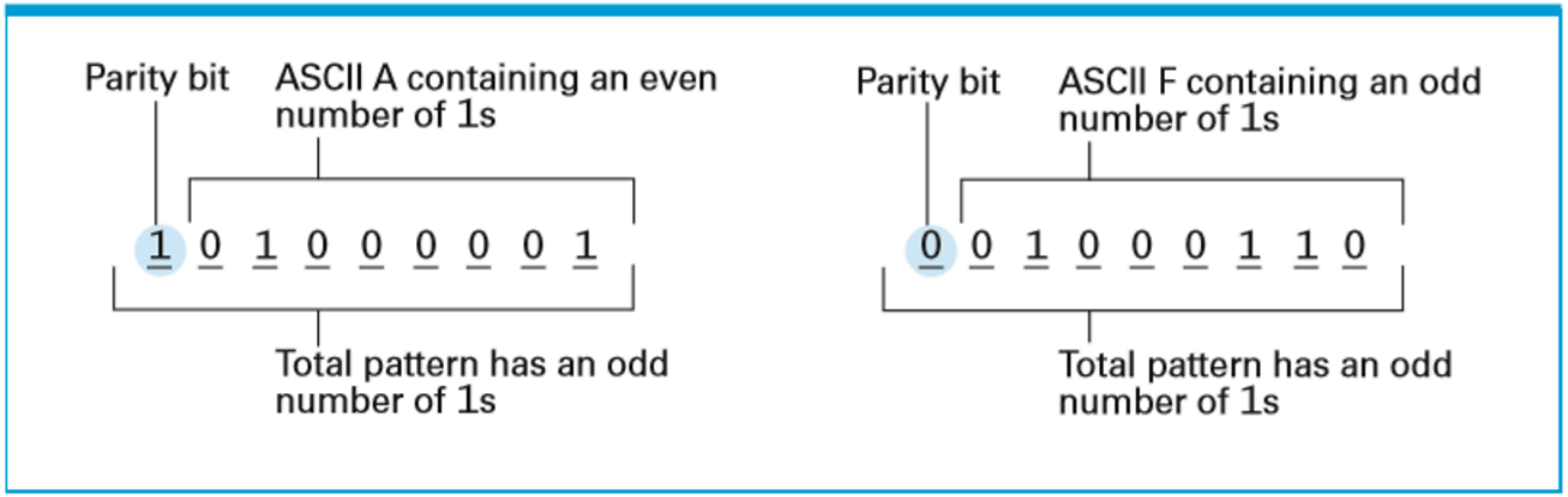
Communicatiefouten

- Informatie kan verloren gaan of fouten bevatten bij communiceren (zelfs “intern” in een machine)
- Systeem nodig voor controle en indien mogelijk correctie

Het binair talstelsel

Communicatiefouten

- Pariteitsbits
  - Extra bit = 1 als aantal 1-en in bericht even is: “odd parity”
  - “Even parity” ook mogelijk



Het binair talstelsel

Communicatiefouten

- Checksums
  - B.v. Belgisch Rijksregisternummer YY.MM.DD-NNN.CC
    - CC is een controlegetal op basis van de 9 voorafgaande cijfers. Berekend door het getal van negen cijfers te delen door 97. De rest van deze deling (“modulo”) wordt van 97 afgetrokken
  - B.v. CRC: ter controle voor langere bitreeks
    - https://crccalc.com/

Het binair talstelsel

Communicatiefouten

- Zelf-corrigerende codes
  - Hamming distance van grootte 3:

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Character	Code	Pattern received	Distance between received pattern and code
A	0 0 0 0 0 0	0 **1** 0 **1** 0 0	2
B	0 0 1 1 1 1	0 **1** **0** 1 **0** **0**	4
C	0 1 0 0 1 1	0 1 0 **1** **0** **0**	3
D	0 1 1 1 0 0	0 1 **0** 1 0 0	**1** — Smallest distance
E	1 0 0 1 1 0	**0** **1** 0 1 **0** 0	3
F	1 0 1 0 0 1	**0** **1** **0** **1** 0 **0**	5
G	1 1 0 1 0 1	**0** 1 0 1 0 **0**	2
H	1 1 1 0 1 0	**0** 1 **0** **1** **0** 0	4