Deel III: Technologie, historiek en economische aspecten van de informatica

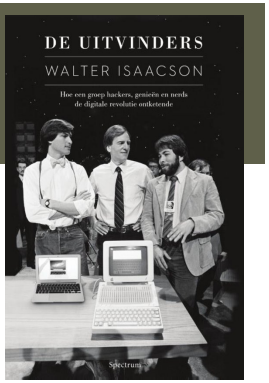
In dit deel bespreken we de technologische, historische en economische aspecten van de informatica. Het is niet de bedoeling te zeer in detail te gaan, maar wel willen we inzicht geven in de dingen die het verschil maken en maakten, zodat je een globaal beeld krijgt en de belangrijkste aspecten van informatica in het juiste perspectief kan plaatsen. Ook hopen we dat dit bijbrengt aan je algemene vorming als ingenieur.

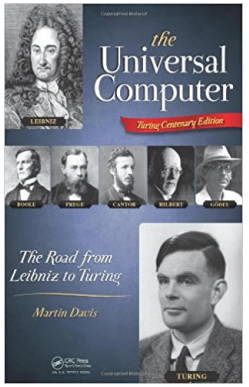
Fouten of opmerkingen:
jan.lemeire@vub.be

VUB VRIJE UNIVERSITEIT BRUSSEL

Hoeilaart, juni 2022

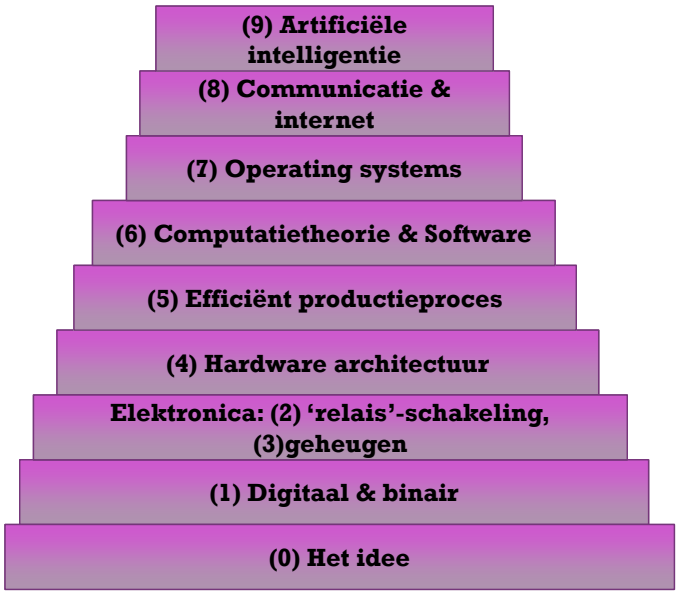
Referenties

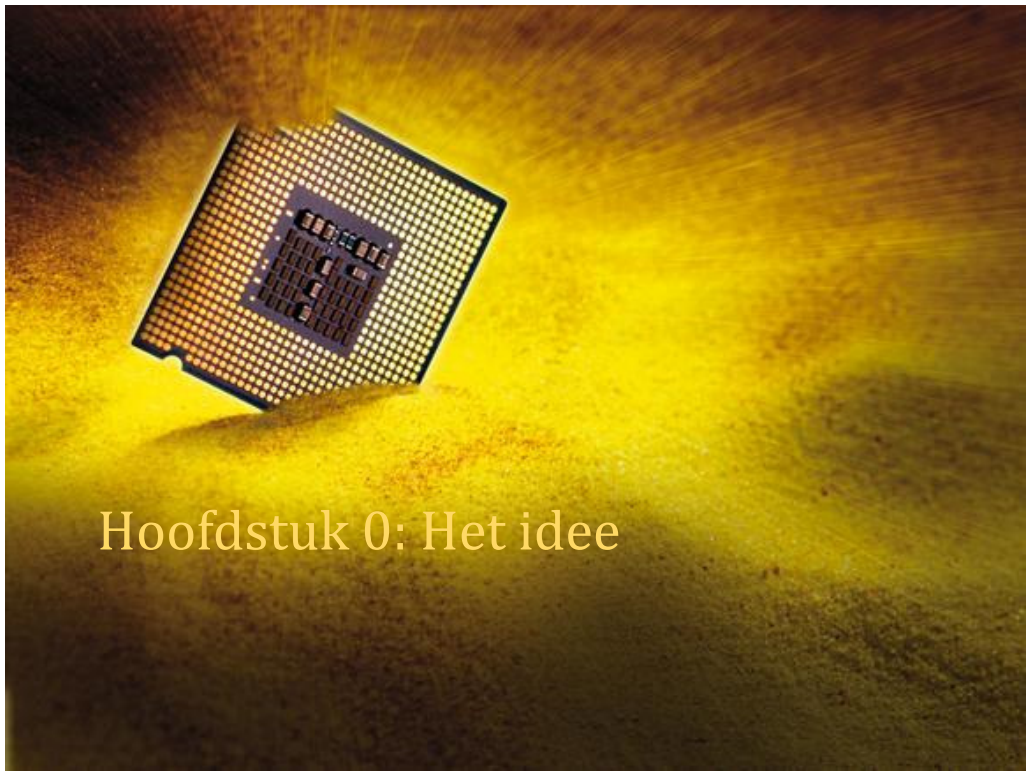




- "The Innovators: How a Group of Hackers, Geniuses and Geeks Created the Digital Revolution", by Walter Isaacson, 2014. *Aanwezig in de VUB-bibliotheek.*
  - In het Nederlands: "De uitvinders, hoe een groep hackers, genieën en nerds de digitale revolutie ontketende."
- "The Universal Computer. The Road from Leibniz to Turing", by Martin Davis, 2000. *Aanwezig in de VUB-bibliotheek.*
- Links op parallel.vub.ac.be

Waarmaken van Leibniz's droom





Calculemus!

Berechnen wir!

Leibniz

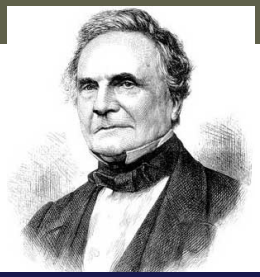
1646 – 1716

$$\frac{\partial f}{\partial x}$$

- Droomde van de "**Calculus ratiocinator**"
  - Een logisch denkend apparaat

Leibniz, een Duitse mathematicus en filosoof (maar duidelijk geen ingenieur) lanceerde het idee van een rekenapparaat die niet enkel kon rekenen met getallen (het telraam bestond al sinds de oudheid), maar ook kon 'rekenen' met symbolen die een betekenis hebben, wat we redeneren noemen. Als we de redeneerregels (cf. logica) formaliseren kunnen deze door een apparaat uitgevoerd worden. Leibniz poneerde dat een tiental wetenschappers/ingenieurs dit apparaat op enkele jaren konden maken.

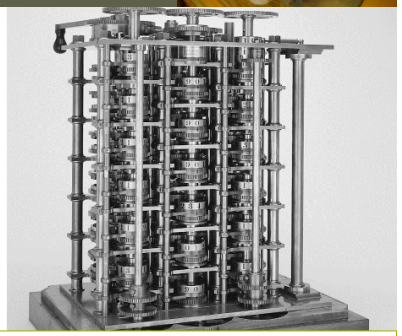
Eerste computer, mechanisch



Charles Babbage
1791-1871

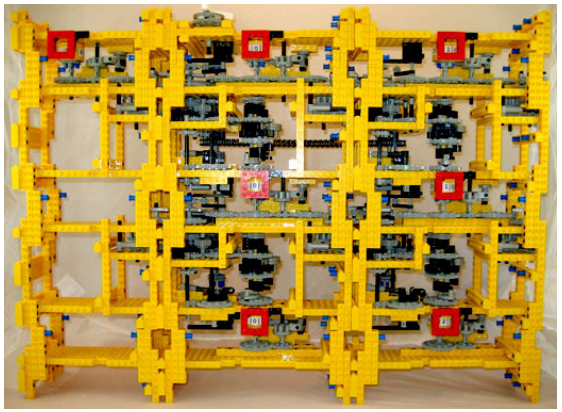
25000 mechanische onderdelen

Kostprijs: 17470 £



Babbage, een engelse wiskundige en ingenieur, heeft de eerste (mechanische) computer gebouwd. De 'difference engine' kon polynomiaalvergelijkingen oplossen. Vervolgens bedacht hij de 'analytical engine', een algemene rekenmachine die verschillende operaties kon uitvoeren op basis van geprogrammeerde instructies. Babbage begreep dat je complex taken kan opdelen in simpele instructies en dat de uitvinder degene is die de ponskaarten (zie hoofdstuk 3) maakt, de programmeur dus. Hij kon echter niet aan het geld komen om de machine te maken…
Zijn ideeën van zijn machine zijn later wel nuttig gebleken voor het bouwen van de eerste elektronische computers!

Babbage Difference Engine made with LEGO



- http://acarol.woz.org/

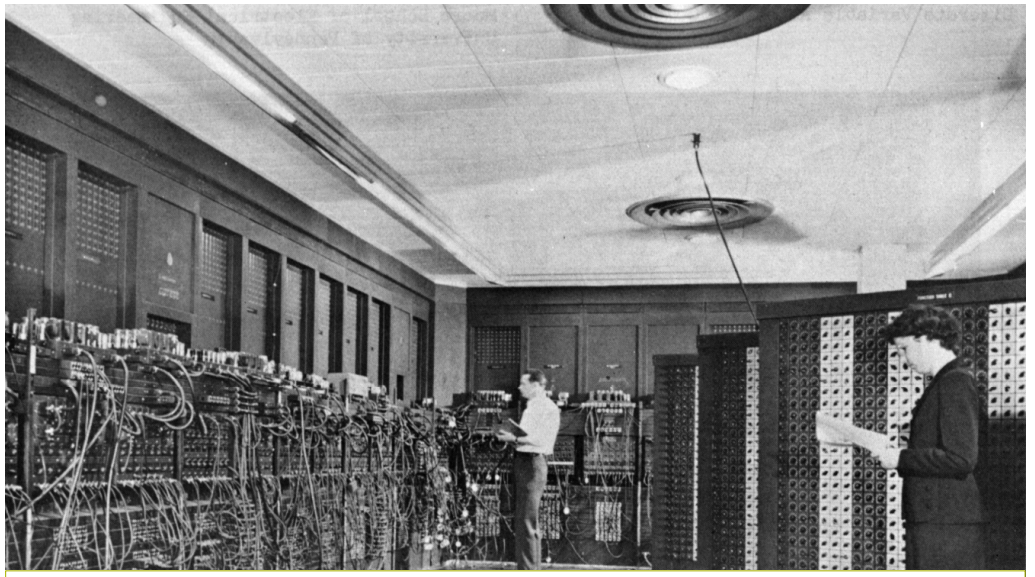
This machine can evaluate polynomials of the form $Ax^2 + Bx + C$ for x=0, 1, 2, …n with 3 digit results.

De eerste computer-wetenschapper & visionair

Ada Lovelace
1815 – 1852

In de 1980s is de programmeertaal 'Ada' ter ere van haar gemaakt

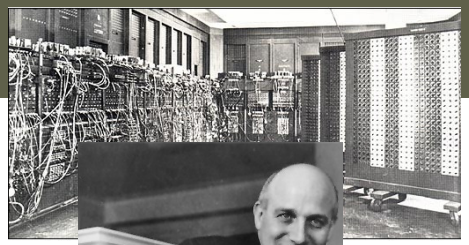
Als dochter van een exacte wetenschapster (moeder) en schrijver (Lord Byron) doet ze aan "poëtische wetenschap". Ze doorziet dat Babbage's ideeën zullen leiden tot een general-purpose machine die geprogrammeerd en geherprogrammeerd kan worden. Maar terwijl Babbage vooral denkt aan rekenen, beseft zij dat die machine ook met symbolen om kan gaan. Symbolen die alles kunnen voorstellen, dus ook mentale processen! Ze ziet in dat je meer kan doen met een computer dan enkel rekenen met getallen.
Ze beschrijft wat software is en een algoritme: stap-voor-stap bewerkingen doen om tot de oplossing te komen. Ze voorziet het inzetten van herbruikbare functies. Ze schrijft het eerste programma bestemd voor de universele computer van Charles Babbage. Ze vraagt zich af of "machines ooit zullen kunnen denken?" ze dacht van niet. Ada ving een glimp van de toekomst...



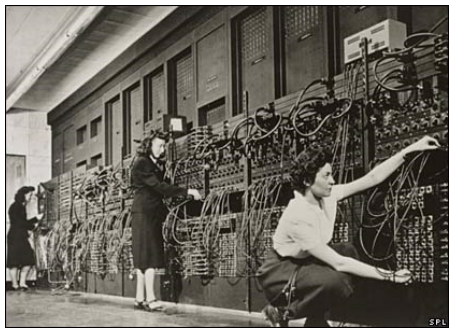
De ENIAC was de eerste operationale elektronische computer, gebouwd aan het einde van de tweede wereldoorlog door het Amerikaans leger. Mauchly en Eckert worden zo erkend als de 'founding fathers' van de computer. Na de oorlog zetten ze hun werk voort en namen verschillende patenten op onderdelen van de computer. Zo bleven computerfabricanten lang schatplichtig aan beide.

ENIAC

Eerste computer: WOII

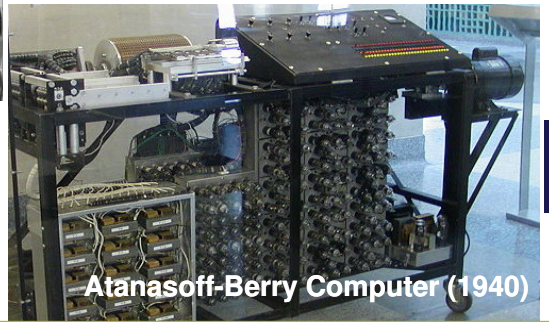


John Mauchly and John Eckert, 1945



De ENIAC en zijn opvolgers werd geprogrammeerd (bij de ENIAC nog door het versteken van draden) door een sterk team van vrouwen (Grace Hopper, ...). Zij bedachten handige programmeertalen en de kunst van gestructureerd en duidelijk programmeren. Zij werden daar toen niet voor erkend. Het was een mannenmaatschappij, en de mannen hielden zich vooral bezig met hardware.

John Atanasoff
1903 – 1995



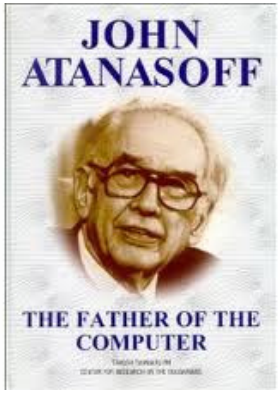
Atanasoff-Berry Computer (1940)

Clifford Berry
1918 – 1963

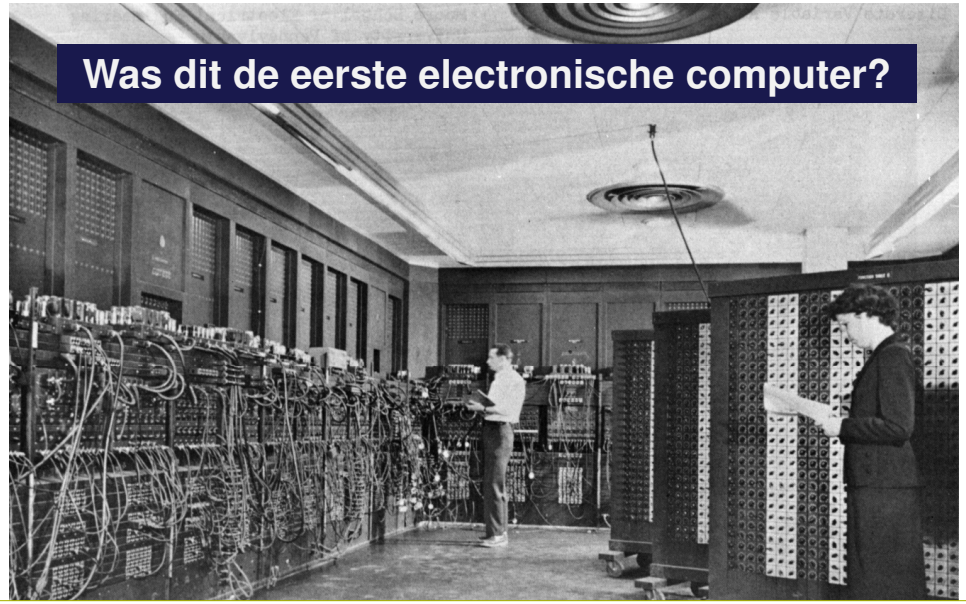
Pas twintig jaar later werd duidelijk dat Eckert ideeën gebruikt had van de computer van Atanasoff en Berry die bij het uitbreken van de tweede wereldoorlog hun werk hadden moeten staken. Er volgde in 1973 een groot proces waarbij de patenten van Mauchly en Eckert (overgekocht door Sperry Rand) warden aangevochten door computerfabrikant Honeywell. Na grondig onderzoek vernietigde de rechter de patenten en erkende dat belangrijke onderdelen uitgevonden zijn door Atanasoff en Berry. Deze laatste maakte het niet meer mee dat ze na vele jaren erkend werden als de grondleggers van de elektronische computer…

Honeywell v. Sperry Rand case, 1973

Honeywell v. Sperry Rand case, 1973



Pas in de 1980s de verdiende erkenning



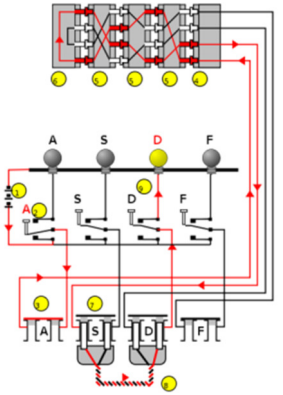
Maar nog is het verhaal niet gedaan, want in het allergrootste geheim hadden tijdens WOII de Britten eveneens een elektronische computer gebouwd voor het kraken van de Enigmacode van de Duitsers. Na de oorlog bleven ze dit geheim houden voor het geval een oorlog met de Russen zou uitbreken.

Winston Churchill

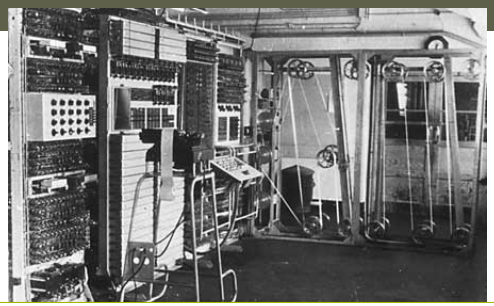
Bletchley Park

Op zoek naar "cribs"

De Enigma code



Colossus: the first operational electronic compu



Alan Turing 1912-1954

In tegenstelling tot de ENIAC die pas na de oorlog werd ingezet voor het berekenen van de banen van raketten, werd deze computer wel succesvol ingezet. De Enigmacode werd gekraakt en zo kon waardevolle tactische en strategische informatie van de Duitsers ontfutseld worden. Het bestaan van deze computer werd ook na de oorlog lang geheim gehouden om dit militair voordeel te kunnen behouden. Hierdoor had deze uitvinding echter niet veel invloed op de ontwikkeling van de moderne computer. De wiskundige Alan Turing was het brein achter de Colossus. Maar ook achter de theoretische informatica zoals we later zullen zien.

Analoog rekenen

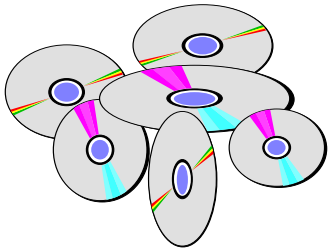
- Gebruik makend van fysische grootheden
  - Cf Babbage
- **Analoge electronica**

  http://www.chem.uoa.gr/applets/appletopamps/appl_opamps2.html

  http://www.falstad.com/circuit/ (***bekijken we later***)
- De rekenlat of 'slide rule' was lange tijd hèt rekeninstrument van de ingenieur

http://www.antiquark.com/sliderule/sim/n909es/virtual-n909-es.html

Muziek



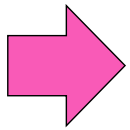
Analoog

informatie gecodeerd door middel van een continu-veranderlijke fysische grootheid

Digitaal

Muziek

Mens hoort tussen 20 en 20,000 Hz



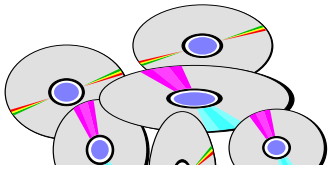
```
-096
+057
+164
+210
+219
+216
+165
-003
-117
-183
-138
-067
```

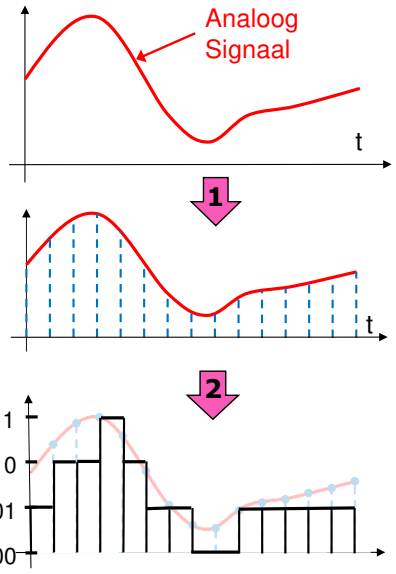
informatie gecodeerd door middel van een continu-veranderlijke fysische grootheid

Analoog

Digitaal (CD)

(44100 metingen/s)





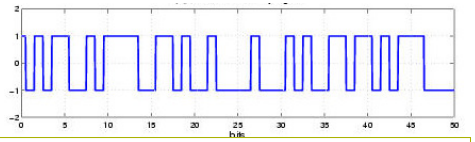
Digitaliseren

(1) Discretizeren: *samplen* in de tijd

(2) Quantizatie: de amplitude in elk samplepunt voorstellen als een binair getal

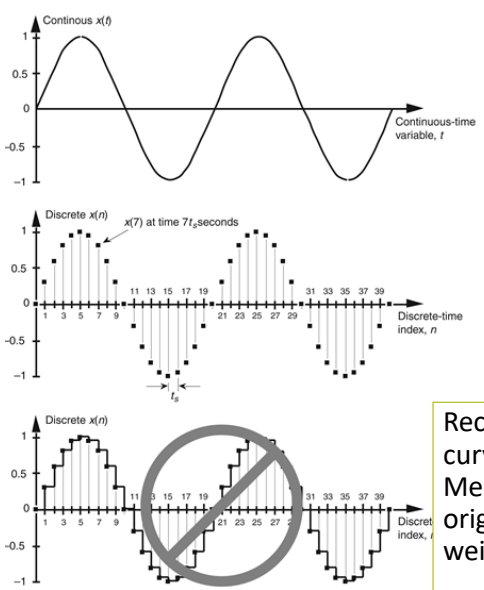
(3) Alle bits achter elkaar

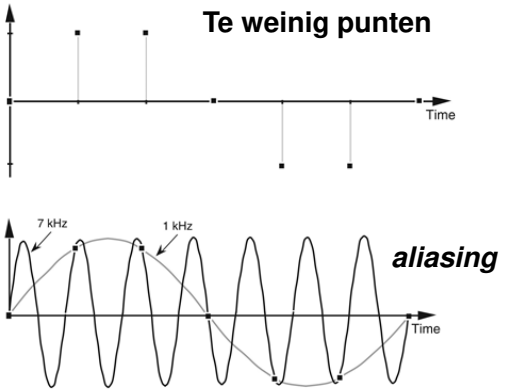
Digitaal signaal



Digitaliseren van een analoog signaal (bvb muziek) gebeurt door op geregelde tijdstippen (discretizeren) de amplitude van het signaal weg te schrijven als een aantal bits (quantizeren). Alle bits achter elkaar geven het digitaal signaal.

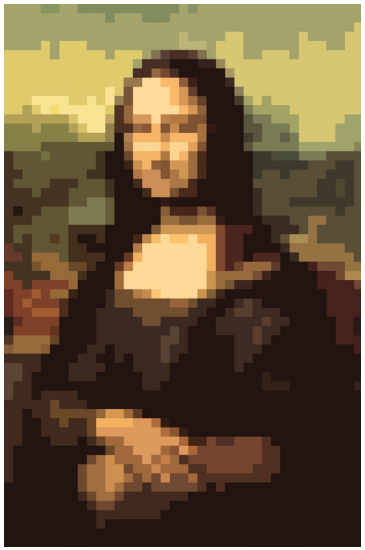
Reconstructie van analoog signaal





Reconstructie gebeurt door een 'vloeiende' curve door de digitale punten te laten lopen. Men moet wel genoeg punten hebben om het origineel te kunnen reconstrueren. Bij te weinig punten krijgt men 'aliasing'.

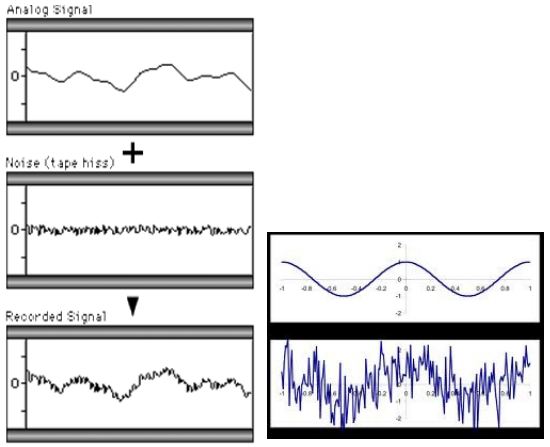
Digitalisatie van beelden



voor elk punt (pixel) de kleur opslaan
=> **bitmap**

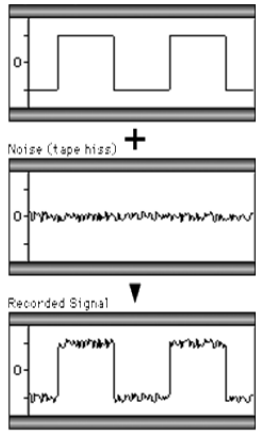
Effect van signaal-vervormingen of ruis (noise)

Analoog signaal



Informatieverlies

Digitaal signaal



Informatie nog te herkennen

Waarom digitaal?

- Unambiguë signalen, immuun voor ruis.
  - Als ruis onder een aanvaardbaar niveau is kan men steeds nog het origineel signaal herkennen. Het is immers een 0 of een 1.
- Perfecte copieën kunnen gemaakt worden.
- Bewerkingen zonder informatieverlies
  - *Digitale signaalverwerking is mogelijk!*
- Simpel, gemakkelijk te maken.
- ➡ Digitale componenten zijn goedkoop, klein, betrouwbaar en men kan er miljoenen op een klein gebied plaatsen.

Alles dat voorgesteld kan worden door één of ander signaal/patroon, kan voorgesteld worden door bits.

Go digital! Go binary!

- Van analoog naar digitaal en binair…
- Leibniz had er al aan gedacht (zoals te zien is in zijn nota's)
- Ook was dit de basis van de computer van Atanasoff en Berry

Waarom binair?

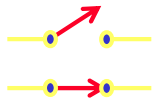
- *De ultieme essentie van informatie: 0 of 1 (to be or not to be)*
- In digitale toestellen gebruikt men meestal het binaire stelsel omdat binaire cijfers eenvoudig kunnen voorgesteld worden door fysische grootheden zoals een positieve of een negatieve spanning, een magnetisch veld in de ene of de andere richting of een open of gesloten schakelaar.
- Eenvoudige reconstructie: groter of kleiner dan een drempelwaarde (*threshold*) bepaalt de bitwaarde

Binaire representatie

• Een binair getal (bit) kan voorgesteld worden door 2 voltages die gegeven kunnen worden door een switch:

– Waarde 0 = 0 Volt = switch open

– Waarde 1 = 5 Volt = switch gesloten



• Een getal van n bits kan 2^n waarden aannemen

– 2 bits : 4 combinaties 00 01 10 11

– 3 bits : 8 combinaties 000 001 010 011 100 101 110 111

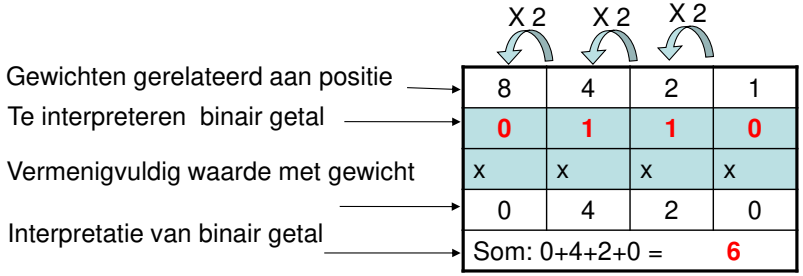
– 8 bits (= 1 byte) 256 combinaties

– 16 bits: 65 536 combinaties

– 32 bits: 4 294 967 296 combinaties

$2^{10} \approx 1000$
$2^{20} \approx 10^6$
$2^{30} \approx 10^9$

Binair talstelsel



128	64	32	16	8	4	2	1	Decimale Interpretatie
1	0	0	0	0	0	1	0	130
0	1	0	1	0	1	0	1	85
0	0	1	0	1	0	0	1	41
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	$\sum_{i=0}^{n} a_i \cdot 2^i$

Enkele oefeningen

128	64	32	16	8	4	2	1	Decimale Interpretatie
1	0	0	0	0	0	1	0	
0	1	0	1	0	1	0	1	
0	0	1	0	1	0	0	1	

Opgave 1: Bereken de decimale interpretatie van de bovenstaande binaire getallen.
Opgave 2: Zet om naar binair: 6, 17, 31, 255
Opgave 3: Zet uw geboortedatum om in binair formaat (dag, maand en jaar apart).
Opgave 4: Hoeveel decimale getallen kunnen er voorgesteld worden aan de hand van een 10-bit binair woord?

Grootte-ordes

		Bytes	
10^3	Kilo	kB	$\approx 2^{10}$= 1024
10^6	Mega	MB	$\approx 2^{20}$
10^9	Giga	GB	$\approx 2^{30}$
10^{12}	Tera	TB	$\approx 2^{40}$
10^{15}	Peta	PB	$\approx 2^{50}$
10^{18}	Exa	EB	$\approx 2^{60}$

Vergeet nooit meer: $2^{10} \approx 1000$

basisoperaties op bits en bytes

And

a \ b	0	1
0	0	0
1	0	1

Or

a \ b	0	1
0	0	1
1	1	1

Exclusive or

a \ b	0	1
0	0	1
1	1	0

a=60		0	0	1	1	1	1	0	0
b=13		0	0	0	0	1	1	0	1
~a	Not	1	1	0	0	0	0	1	1
a & b	And	0	0	0	0	1	1	0	0
a | b	Or	0	0	1	1	1	1	0	1
a ^ b	Exclusive or	0	0	1	1	0	0	0	1
a << 2	Shift left	1	1	1	1	0	0	0	0
a >> 2	Shift right	0	0	0	0	1	1	1	1

Zie programma `BitwiseOperators` in package `voorbeelden`

Programma van vorige tabel

Zie programma **BitwiseOperators** in package **voorbeelden**

```
public class BitwiseOperators {
   public static void main(String[] args) {
      int a = Integer.parseInt("00001101", 2); //2= basis van talstels
      int b = 13;/* 13 = 0000 1101 */       /* a = 0011 1100 = 60 */
      int c = 0;

      c = a & b;        /* 12 = 0000 1100  and */
      System.out.println("a & b = " + c );

      c = a | b;        /* 61 = 0011 1101  or */
      System.out.println("a | b = " + c );

      c = a ^ b;        /* 49 = 0011 0001  exclusive or (xor) */
      System.out.println("a ^ b = " + c );

      c = ~a;           /*-61 = 1100 0011  not */
      System.out.println("~a = " + c );

      c = a << 2;       /* 240 = 1111 0000  shift left */
      System.out.println("a << 2 = " + c );

      c = a >> 2;       /* 15 = 1111        shift right */
      System.out.println("a >> 2  = " + c );
   }
}
```

Enkele toepassingen

Gegeven: integers x & i

```
int x = 85, i = 4;
```

	7	6	5	4	3	2	1
x =	1	0	1	0	1	0	1
y =	0	0	1	0	0	0	0

1. "Is de (i+1)-de bit van x gelijk aan 1?"

```
int y = 1 << i; // y wordt een masker genoemd
if ((x & y) > 0)
   System.out.println("De "+(i+1)+"de bit van" +x+" is 1.");
else
    System.out.println("De "+(i+1)+"de bit van" + x +" is 0.");
```

2. "Zet de (i+1)-de bit van x op 1"

```
i = 1;
y = 1 << i; // een masker met een 1 op plaats i
x |= y; // is identiek aan x = x | y;
System.out.println("x = "+x+": "+(i+1)+ "de bit is nu 1.");
```

3. "Wat zijn de waarden van de 4e tot en met de 6e bit?"

```
int z = (x >> 3) & 7; // 7 is hier ook een masker: 111
System.out.println("Bits 4 tot en met 6 van "
+Integer.toBinaryString(x)+" zijn "+Integer.toBinaryString(z));
```

Hexadecimaal talstelsel

Om de notatie van binaire getallen te verkorten wordt ook de ***hexadecimale notatie*** gebruikt. Een **hexadecimaal cijfer** beschrijft 4 bits en wordt voorgesteld door 0, 1, ..., 9, A, ..., F.

Binair	Hex.
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binair	Hex.
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Vb: omzetting van 1110001101

1	1	1	0	0	0	1	1	0	1
3			8			D			

Om aan te duiden dat een getal in het hexadecimaal talstelsel is (bvb tijdens programmeren), voegen we de prefix '0x' toe: 0x38D.

Naar tiendelig talstelsel:
$0x38D = 3*16^2 + 8*16^1 + 13*16^0 = 3*256 + 8*16 + 13 = 869$
0x11 is dus gelijk aan 17

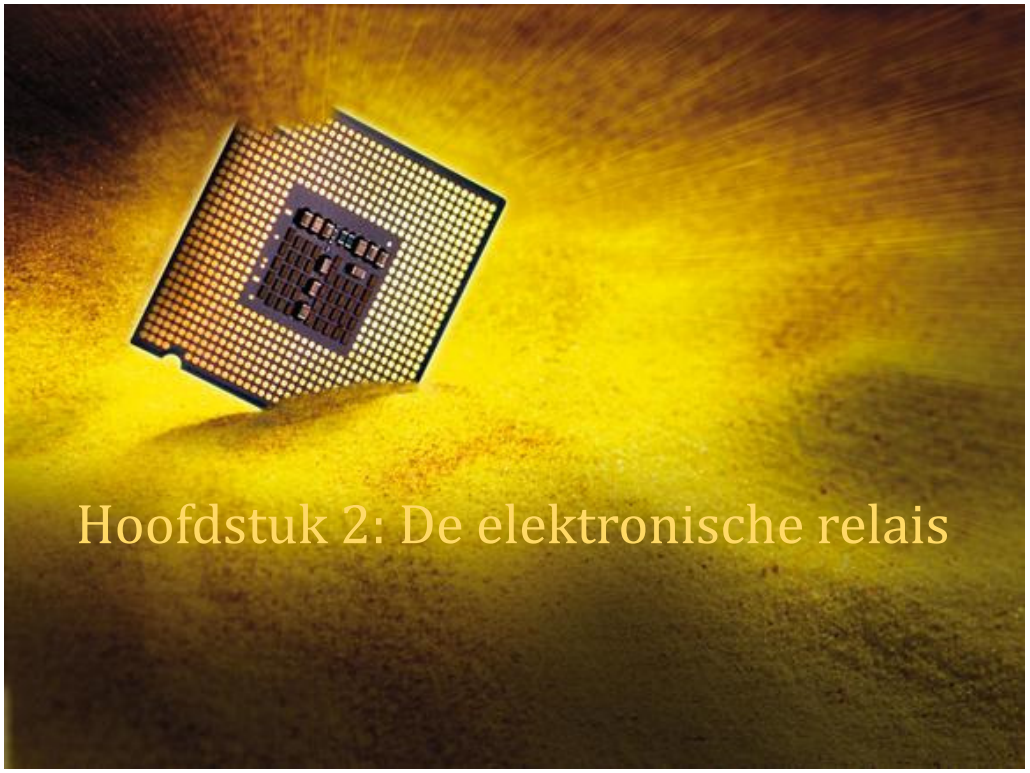
Voorstelling van karakters

De huidige computers gebruiken gestandaardiseerde tabellen waarin de alfanumerieke tekens die op het toetsenbord staan geassocieerd worden met binaire getallen. De meestgebruikte is de ASCII tabel.

De 128 tekens van hiernaast kunnen voorgesteld worden met 7 bits (2^7=128)

TABLE 3
ASCII CHARACTER CODES (DECIMAL)

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



Binair rekenen?

- 1 input => 1 output
  - Maar 1 niet-triviale functie: NOT
- 2 inputs => 1 output

	0	1
0	?	?
1	?	?

2^4 = 16 mogelijkheden

- Genoeg voor een basis: AND, OR, XOR, NOT

Alle functies (met meer inputs) zijn samen te stellen uit deze!!

2 inputs => 1 output: 16 mogelijkhed

	0	1
0	0	0
1	0	0

	0	1
0	0	1
1	0	1

	0	1
0	1	1
1	0	0

	0	1
0	1	1
1	1	1

	0	1
0	1	0
1	1	0

	0	1
0	0	0
1	1	1

triviaal

	0	1
0	0	1
1	1	0

	0	1
0	1	0
1	0	1

exclusive or (xor)

and

	0	1
0	1	0
1	0	0

	0	1
0	0	0
1	0	1

	0	1
0	0	0
1	1	0

	0	1
0	0	1
1	0	0

	0	1
0	0	1
1	1	1

	0	1
0	1	1
1	1	0

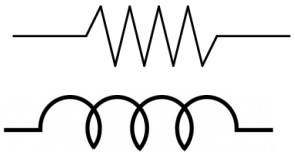
	0	1
0	1	1
1	0	1

	0	1
0	1	0
1	1	1

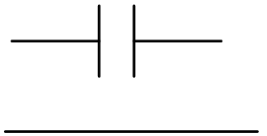
or

Hoe electrisch maken?

1. Geleider
   1. Weerstand
   2. Spoel
   3. Verbinden van componenten



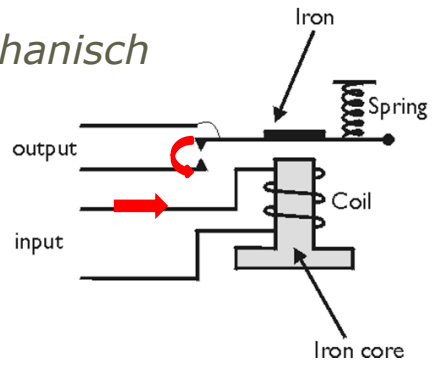
2. Isolator (*dielectric*)
   1. Condensator
   2. Scheiden van componenten

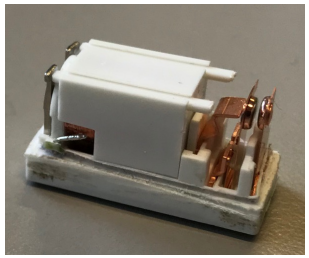


Extra component nodig…

Nodig: de relais

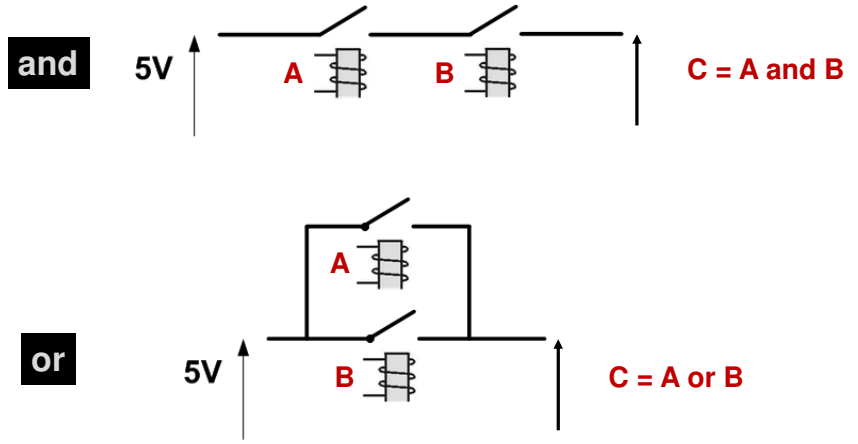
Electro-mechanisch





Naast de geleider, weerstand, capaciteit en spoel hebben we een component nodig met de werking zoals de elektromechanische relais. De relais zorgt ervoor dat een signaal een ander signaal kan beïnvloeden.
Het inputsignaal bepaalt of de output 'aan' of 'uit' is . Met deze component kunnen we elektrische schakelingen maken die binair kunnen rekenen. Gebaseerd op de elektromechanische relais bouwde de duitser Zuse een volledig werkende computer die gebruikt werd tijdens de tweede wereldoorlog.

Binair rekenen



Met deze componenten kan je alles berekenen!

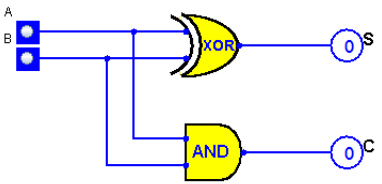
Complexe functies

- Algemeen: van *n* inputs naar *m* outputs
  - 2-naar-*m* kan je ontbinden als *m* keer 2-naar-1
  - *n*-naar-1 kan je ontbinden door 2 bij 2 samen te nemen
    - Bvb **{a, b, c, d} => x** door **{a, b} => e**, **{c, d} => f** en **{e, f} => x**
  - Beide samen geeft *n*-naar-*m*
    - *is misschien wel niet het efficientste*

Som van bits

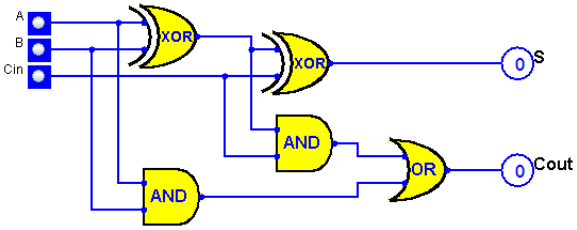
Half adder

- Som S van 2 bits A en B. De carriage (C) geeft de bit voor de volgende orde.



Full adder

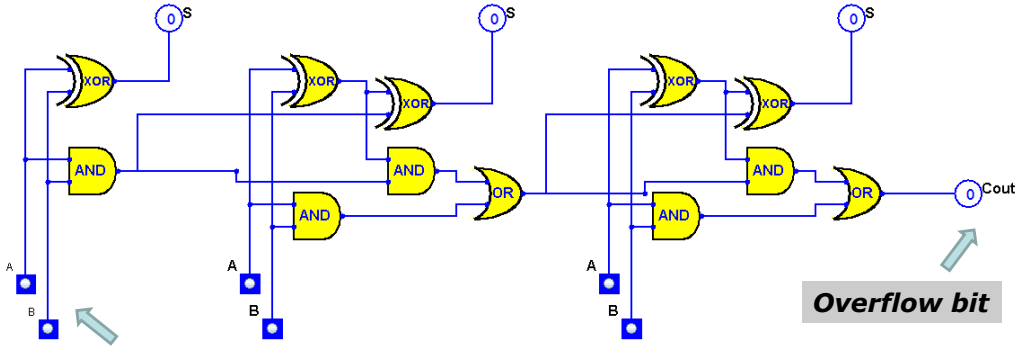
- Rekening houdend met de carriage bit (Cin) van de vorige orde.



Som van getallen

- Twee getallen (A en B) van 3 bits bij elkaar tellen (S)

Ripple carry adder

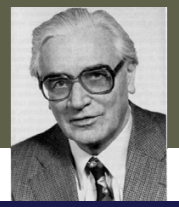


Overflow bit

Least-significant bits (bits van laagste orde)

Zie http://parallel.vub.ac.be/education/java/applets.html

WOII: de duitsers



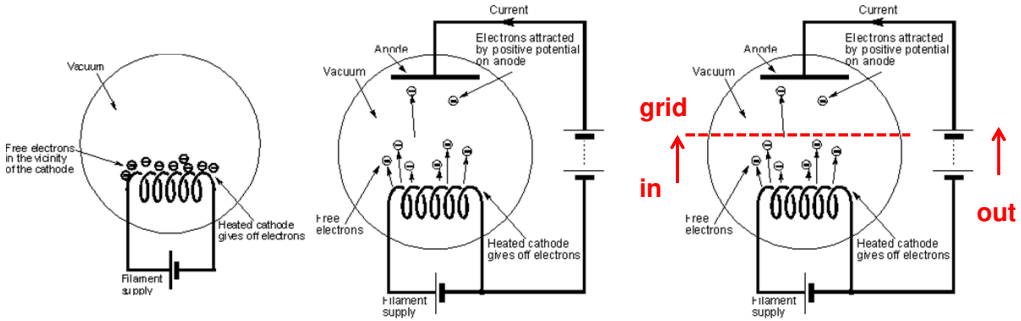
Konrad Zuse

- Een werkende electro-mechanische computer

Door de mechanische componenten beperkt tot vijftien à twintig operaties per seconde

De Vacuümbuis: elektronische relais

Vrije elektronen in vacuüm



gloeilamp **diode** **versterker**

De spanning (*in*) van het grid van de vacuümbuis bepaalt de geleidbaarheid tussen anode en kathode. Door een grote spanning op *out* te zetten, zal een klein inputsignaal *versterkt* worden. Een signaal met een klein vermogen wordt omgezet in een signaal met een groot vermogen.

Vacuümbuizen

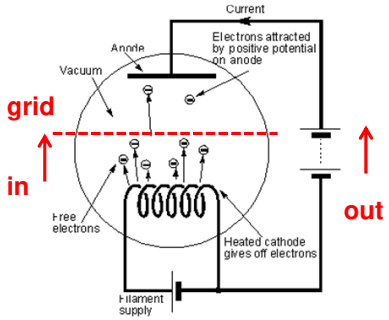
Veel gitaristen gebruiken lampenversterkers vanwege hun mooi 'natuurlijk' geluid.

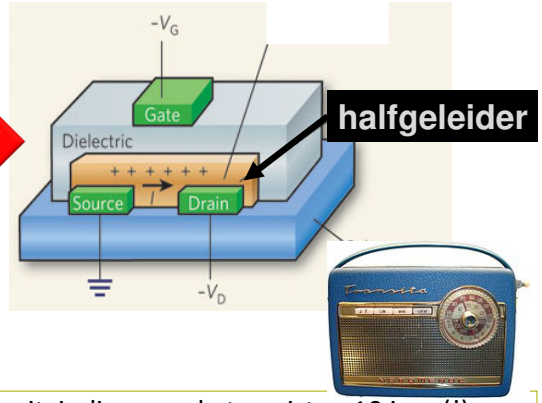
De eerste computer: ENIAC

John Mauchly and John Eckert, 1945

De ENIAC maakte gebruik van vacuümbuizen voor het rekenen. Deze zijn echter duur, groot, onhandig en gaan gemakkelijk stuk.

Van vacuümbuis naar transistor

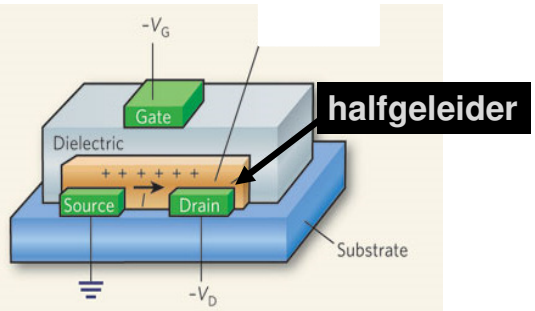


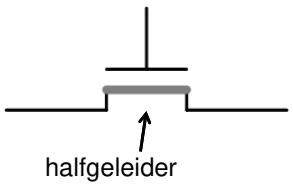


Maar de echte doorbraak kwam met de uitvinding van de transistor. 10 jaar (!) werd er in Bell Labs (van AT&T - American Telephone & Telegraph Company) actief gezocht naar een handige vervanger van de vacuümbuis. Het labo bestond uit praktische technici, theoretici (fysica, wiskunde, chemie) en creatieve geesten. De eerste succesvolle toepassing was de draagbare transistorradio.

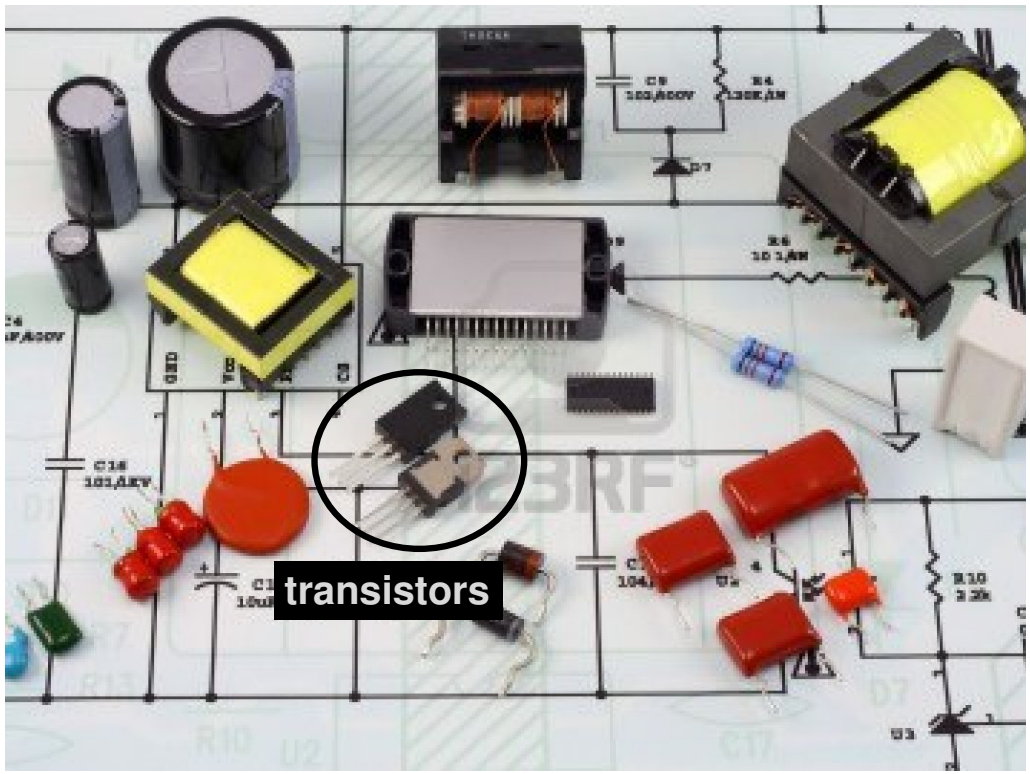
Transistor (MOSFET)

- De spanning V_G bepaalt de geleidbaarheid van de halfgeleider (semiconductor) tussen Source en Drain
  - Te gebruiken als versterker
  - Te gebruiken als relais, voor binaire operaties

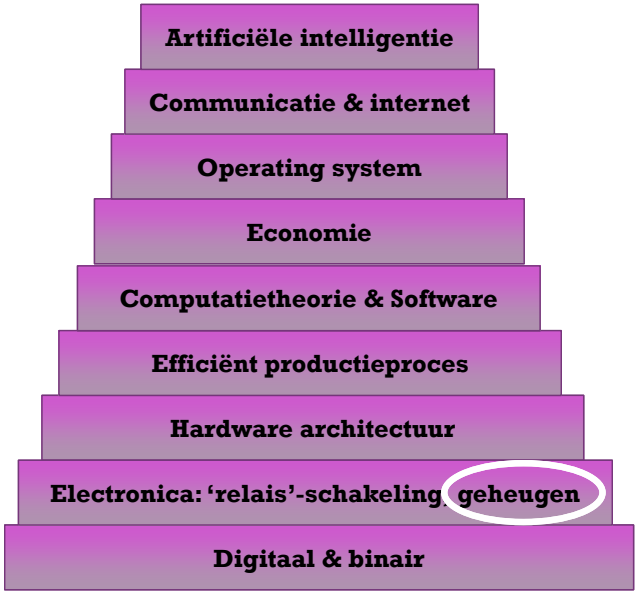




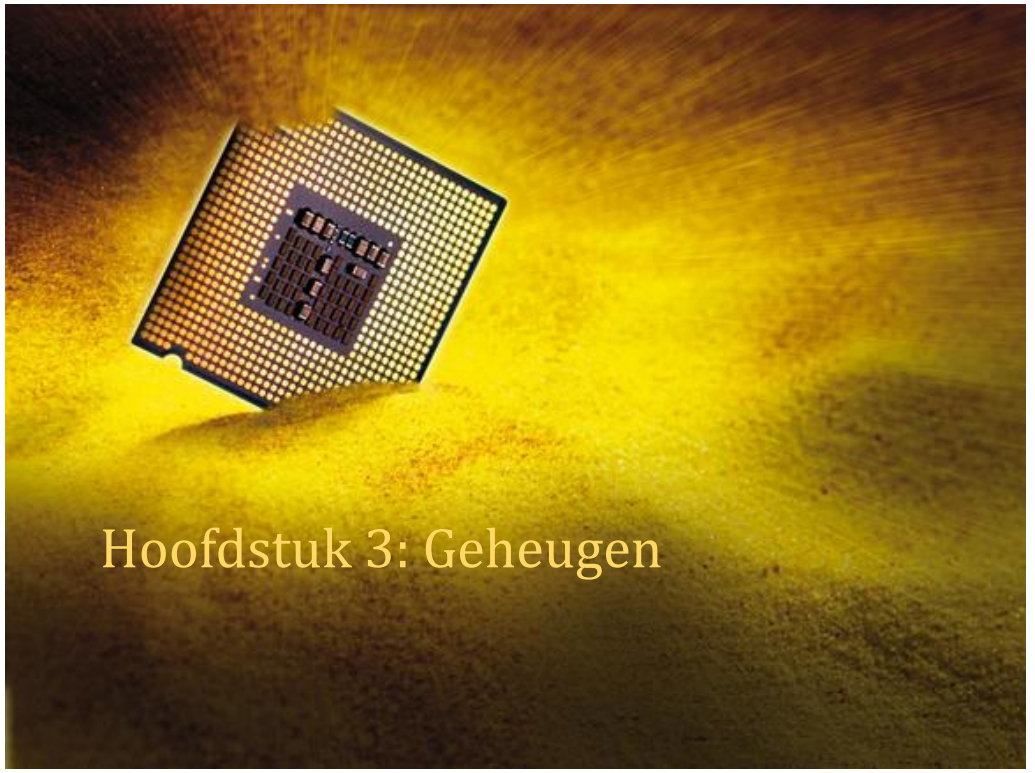
De halfgeleider gedraagt zich als een isolator als V_G=0 en als geleider bij V_G>1V. We hebben een elektronische relais!



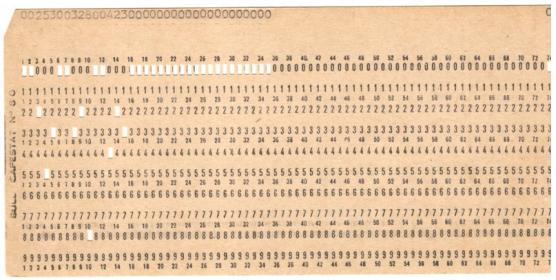
Waarmaken van Leibniz's droom



Informatica deel III: technologie, historiek en economische aspecten



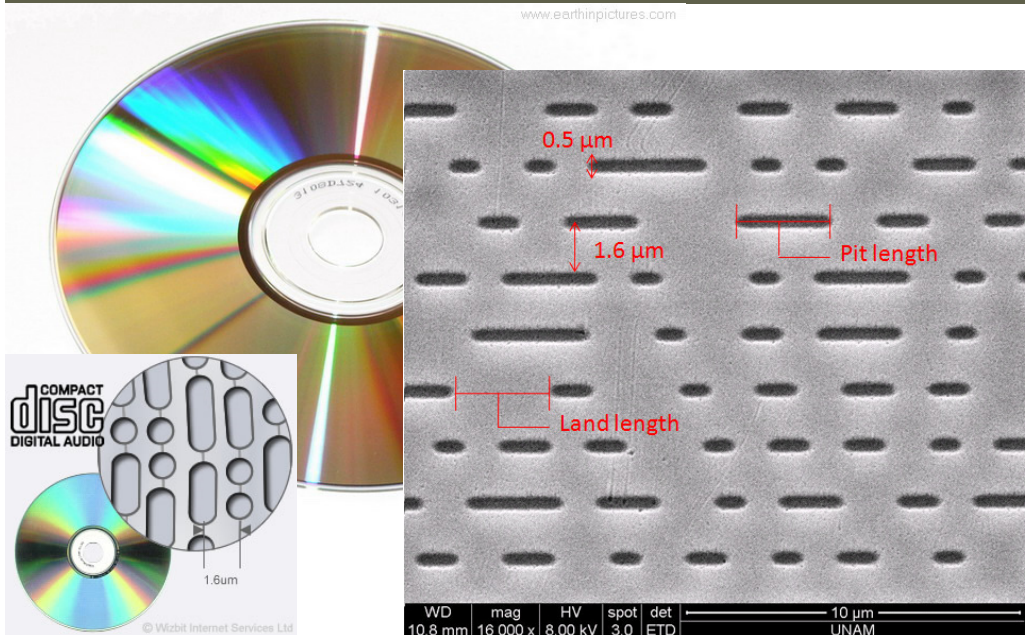
Ponskaarten



- Beschrijven de programma's

De programma's van de eerste computers werden beschreven door de instructies te coderen met gaatjes in een kartonnen kaart. Deze werden gemakt met een soort tijpmachine en dan gelezen door de computer.

(Moderne) ponskaart!



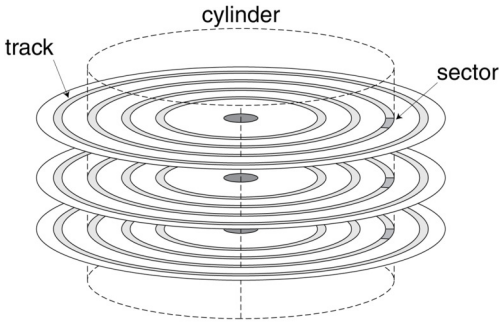
Magnetische banden

- zoals muziek- en videocassettes

Het opslaan van gegevens gebeurde met magnetische banden, wat nu gebeurt met een harde schijf of flashgeheugen.

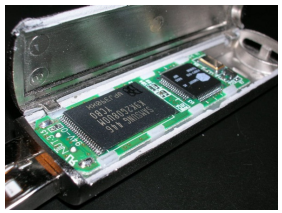
Harde schijf

- Permanent, rotating magnetic storage



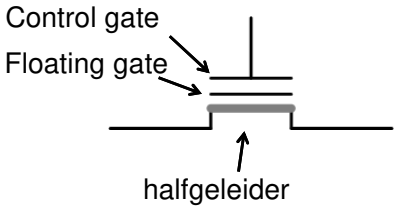
Flashgeheugen

- Permanent <u>semiconductor</u> storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)

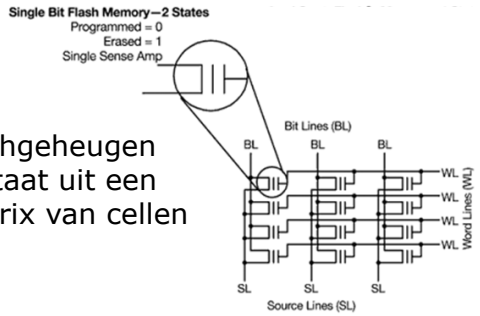


Hoe werkt Flashgeheugen?

- Elektrische lading wordt bewaard in een ingekapselde geleider *(de floating gate).*
  - 50.000 elektronen zijn voldoende, maar door lekken gaan die toch verdwijnen. Zeker meer dan 10 jaar OK, maar hoe lang flashgeheugen veilig is, is nergens te vinden.
  - De floating gate bepaalt geleidbaarheid van halfgeleider

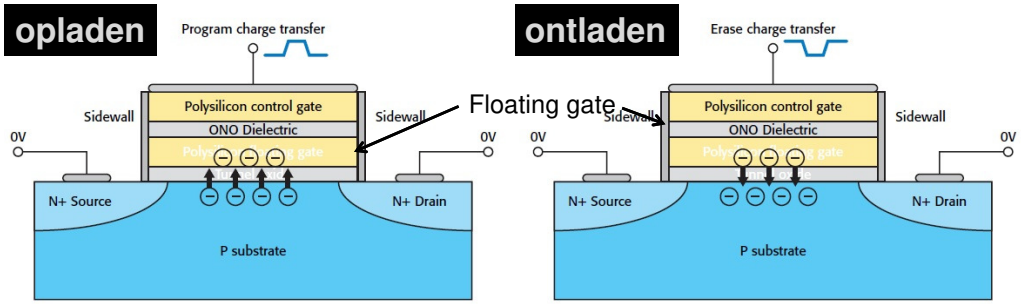


Flashgeheugen bestaat uit een matrix van cellen



Op- of ontladen flash bit

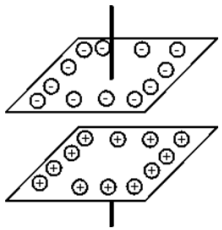
- Op- of ontladen van *floating gate* gebeurt door de isolator heen via ***tunneling*** (kwantum-mechanisch effect)
- Geactiveerd via hoge spanning op *control gate*



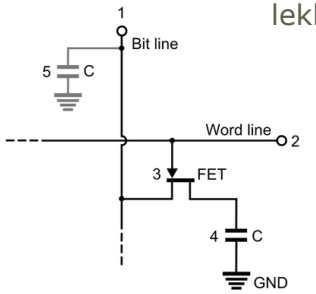
Permanent versus volatiel geheugen

- **Permanent**: "eeuwigdurend"
  - Optisch: Ponskaarten en CDs
  - Magnetisch: banden, diskettes en harde schijven
  - Flash-geheugen
  - Optisch & magnetisch enkel *sequentieel* leesbaar: je moet eerst 'doorspoelen' naar de juiste positie
- **Vluchtig of volatiel**: electriciteit nodig om gegevens aktief te behouden
  - Wordt gebruikt als *werkgeheugen*
  - Random Access Memory (RAM): elke byte kan onmiddellijk gelezen worden, is direct toegankelijk

Lading bewaren: condensator

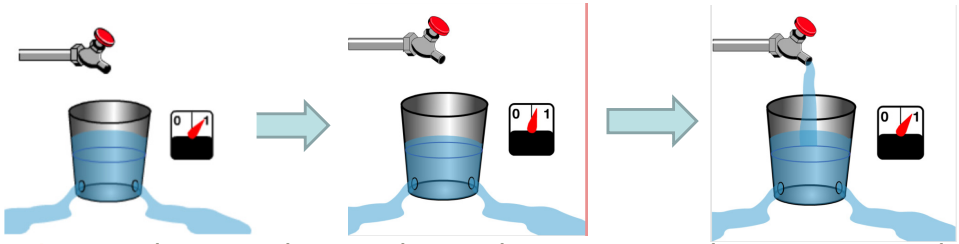


- Het geheugen dat de snelste transfer van bits van en naar de processor toelaat is dat type van geheugen waarbij elektrische schakelingen gebruikt worden waarbij 1 of 0 wordt voorgesteld als de aan- of afwezigheid van spanning over een geleider.
- Maar lading zal langzaam wegvloeien en lekken…



DRAM-cel die 1 bit kan opslaan met behulp van een condensator (C) en een transistor (MOSFET of FET)

Volatiel geheugen: refreshen!

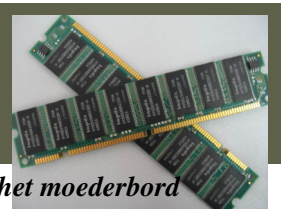


- Waarde van geheugen hou je bij op een condensator, maar die lading vloeit weg
- ➡ regelmatig weer opladen is noodzakelijk!
- **Dynamic Random Access Memory** (DRAM): elke cel slaat 1 bit op en bestaat uit een *capaciteit* voor de bitwaarde te bewaren, en een *transistor* voor refreshen/lezen/schrijven
- **Static Random Access Memory** (SRAM): een alternatief bestaande uit enkel transistors (*flipflops*). SRAM is sneller, maar neemt meer plaats in en is duurder.

Werk- versus perifeer geheugen

- **Werk- of primair geheugen**: rechtsstreeks bruikbaar in je programma's
  - Elk programma krijgt een deel van het geheugen toegewezen om in te 'werken'
  - Heeft elektriciteit nodig, bestaat enkel als computer opstaat, is dus vluchtig of volatiel, dus niet geschikt voor het opslaan van permanente gegevens
- **Perifeer of secundair geheugen**: om gegevens permanent op te slaan
  - Gebeurt via files
  - Vanuit programma lees en schrijf je van en naar *files*
  - Goedkoop, maar wel traag

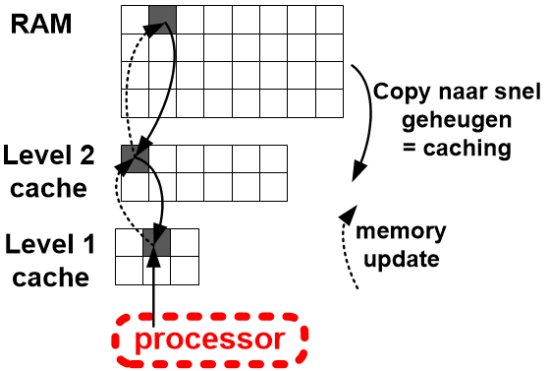
Werkgeheugen: RAM



DRAM-geheugenchips in te pluggen zijn in de DDR-slots van het moederbord

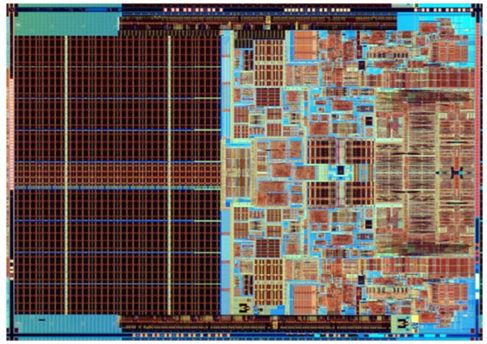
Het cachegeheugen (SRAM) is typisch in de chip van de CPU (processor) zelf ingebouwd en gebruikt hoge snelheidsconnecties zodanig dat de data erin opgeslagen extreem snel kan aangesproken worden. Het DRAM-geheugen daarentegen is op afzonderlijke chips ondergebracht en communiceert met de CPU tegen een lagere snelheid. De data aanwezig in het DRAM-geheugen kan dus minder snel aangesproken worden dan de data in het cachegeheugen, maar het DRAM-geheugen is veel goedkoper.
Men zal daarom kostprijs tegenover performantie afwegen en een klein gedeelte (meestal uitgedrukt in kilobyte) cachegeheugen voorzien voor de opslag van essentiële, veelgebruikte gegevens, terwijl men voor de overige gegevens RAM geheugen zal voorzien (meestal uitgedrukt in megabyte of gigabyte). Er is naast het RAM geheugen ook nog een beperkt gedeelte ROM (afkorting voor Read-only Memory) aanwezig. Read-Only-Memory (ROM) is geheugen waarvan de inhoud permanent is, het kan enkel gelezen worden en niet veranderd. Bovendien verdwijnt de inhoud van dit geheugen niet wanneer de stroom komt weg te vallen. Typisch wordt het ROM-geheugen gebruikt om permanente programma's in te stockeren zoals deze die nodig zijn wanneer de computer wordt aangezet.

Caching



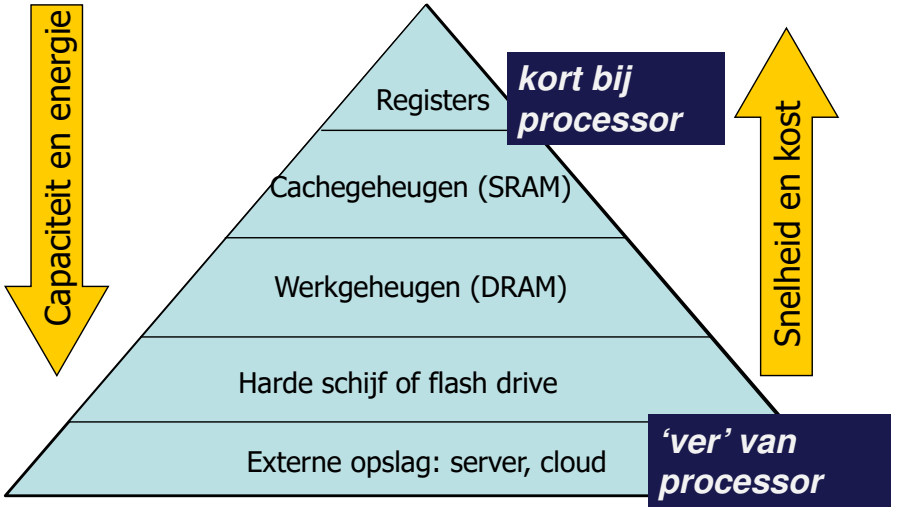
Caching bestaat uit het kopiëren van gegevens van traag geheugen naar snel geheugen. Na het kopiëren kunnen de gegevens snel gelezen en aangepast worden in het snel geheugen. Na aanpassing worden de gegevens eveneens aangepast in het origineel geheugen.
RAM-geheugen moet groot zijn (Gigabytes), maar kan daarom niet tegelijkertijd ook goedkoop en snel zijn. Dit lost men op door caching.

Processorchip



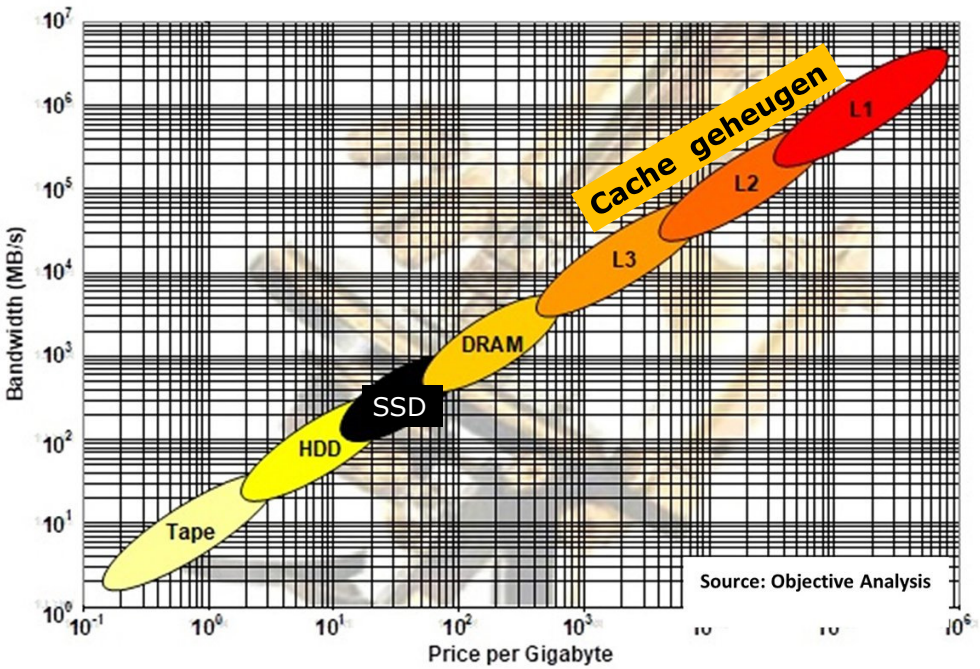
- Dual core chip met aanduiding in blauw van *registers*.
  - Dual core: 2 processors (quad core = 4 processors), je ziet dat de chip uit 2 symmetrisch-gelijke delen bestaat.
  - Registers: zeer snel, klein geheugen voor het bijhouden van tussentijdse waarden van berekeningen.
- Het *cache geheugen* bevindt zich in het donkere linkergedeelte.

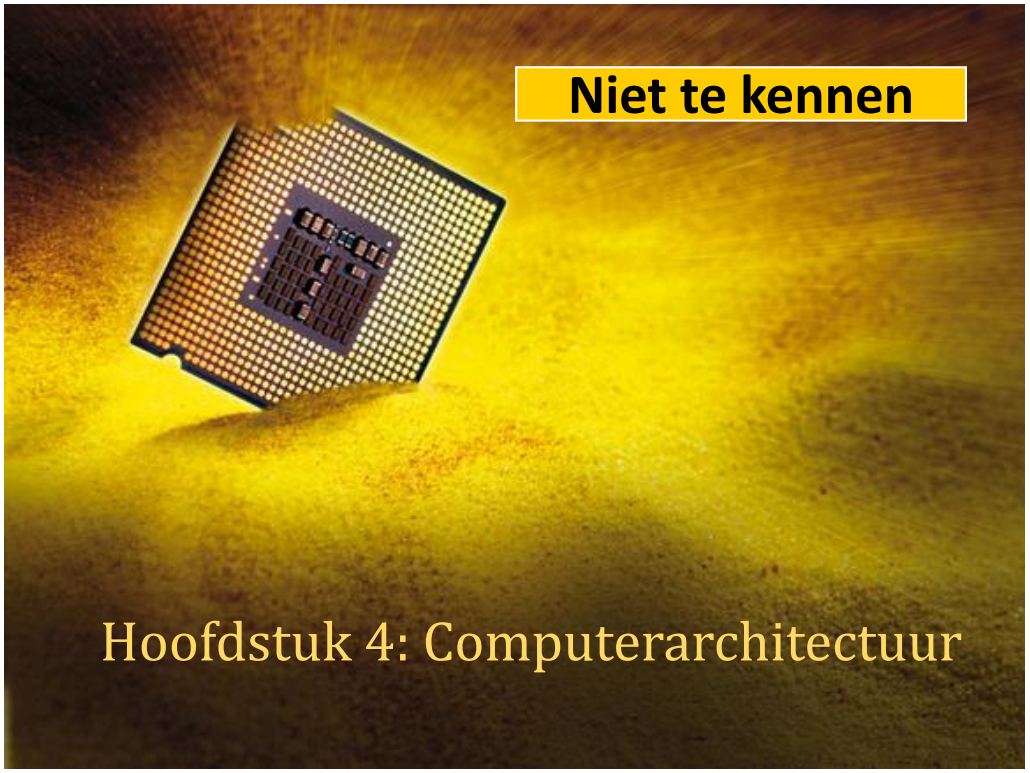
Geheugenhiërarchie



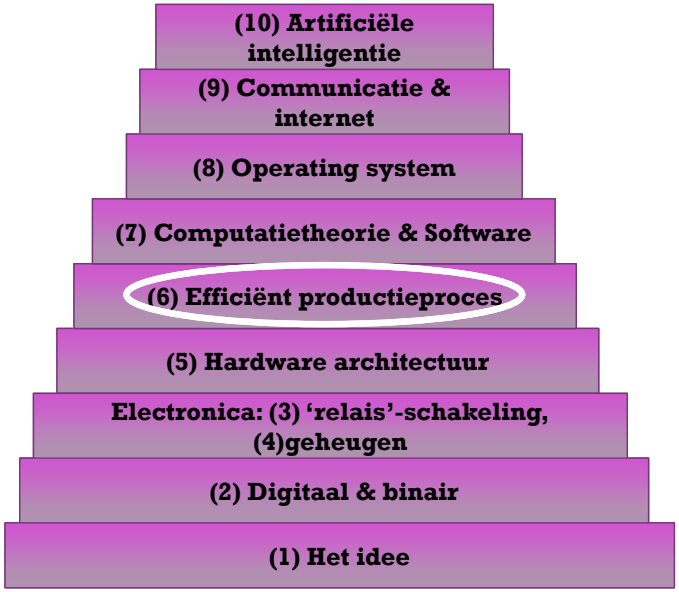
Geheugenhiërarchie

Ondanks het feit dat de von Neumann-architectuur het computergeheugen als één enkele component afbeeldt, gebruiken moderne computers een combinatie van geheugentypes, elk met z'n eigen karakteristieke performantie en kostprijs.
De trade-off tussen performantie en kost is afgebeeld op de volgende slide.
Aangezien de prijs van geheugens sterk afhankelijk is van de toegangstijd (dat is de tijd nodig om gememoriseerde informatie terug te vinden) wordt het geheugen meestal fysisch opgesplitst in delen met verschillende eigenschappen: duur, zeer snel geheugen voor wat snel bereikbaar moet zijn, en traag, goedkoper geheugen voor wat niet onmiddellijk nodig is. De snelle technologieën worden gebruikt voor registers en het primair werkgeheugen, terwijl de goedkope technologieën aangewend worden in de secundaire periferische geheugens.
Recentelijk is er de mogelijkheid tot 'remote storage', het extern opslaan van je gegevens in de cloud of op een server.

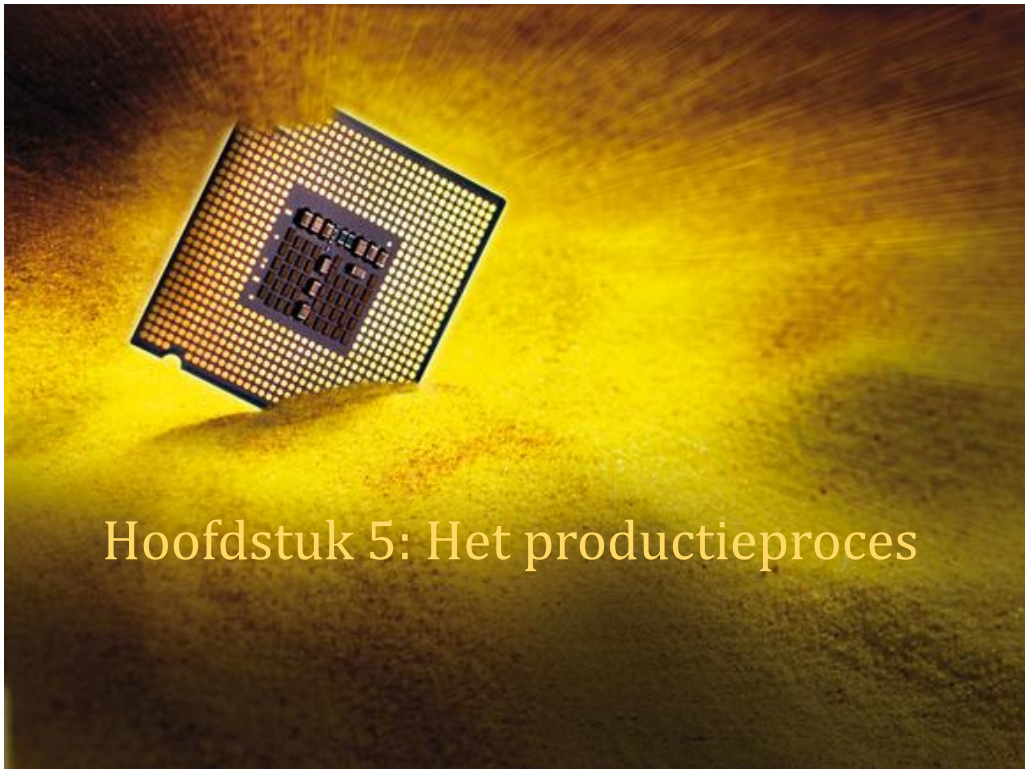



Niet te kennen
Hoofdstuk 4: Computerarchitectuur

Waarmaken van Leibniz's droom



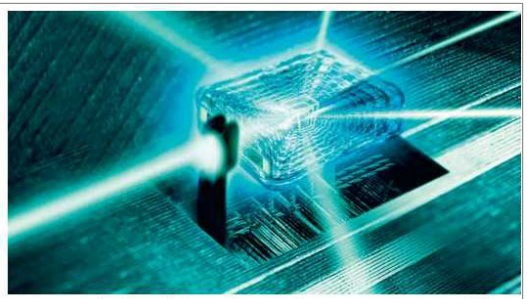
Informatica deel III: technologie, historiek en economische aspecten



Natuurkunde

Licht opsluiten

Door licht in cirkels te buigen kan het in piepkleine volumes verdwijnen.



FOTONICA *Lichtdeeltjes maken snellere computers mogelijk dan elektronica.*

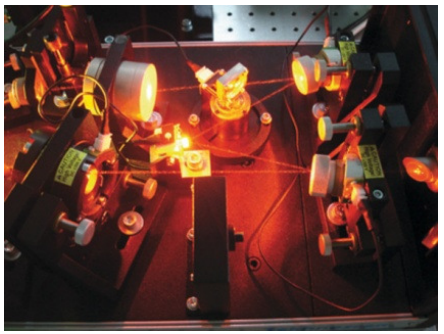
Omdat er niets is wat sneller beweegt dan licht, hopen wetenschappers van licht gebruik te kunnen maken om informatie over te dragen. Want dat zou dan nog sneller kunnen dan met de klassieke elektrische systemen van nu het geval is.

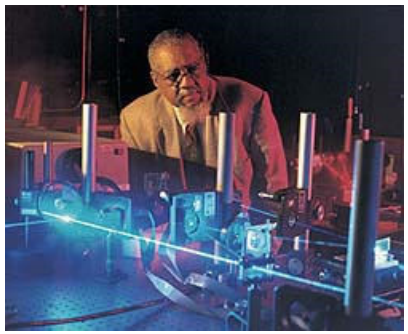
Maar lichtdeeltjes zijn minder gemakkelijk te controleren dan de elektronen uit onze dagelijkse elektronica, waardoor ze vooral op heel kleine schaal (zoals in de computerchipindustrie) moeilijk te gebruiken zijn. Daar komt echter stilaan verandering in. Fysicus Vincent Ginis van de Vrije Universiteit Brussel beschrijft met een aantal collega's in de *New Journal of Physics* een manier om licht op te sluiten in héél kleine eenheden, waardoor een verdere miniaturisatie van de fotonica (als tegenhanger van de elektronica) mogelijk wordt.

Hun idee is geïnspireerd door de recente ontwikkeling van een 'onzichtbaarheidsmantel', waarbij licht op zo'n vloeiende manier rond een voorwerp gebogen wordt dat het onzichtbaar wordt. Door dat principe om te keren kan licht op een vloeiende manier in cirkels worden gebogen, zodat het in piepkleine volumes kan worden opgesloten.

Hoe kleiner de componenten, hoe meer er op een chip kunnen.

Optische computer?





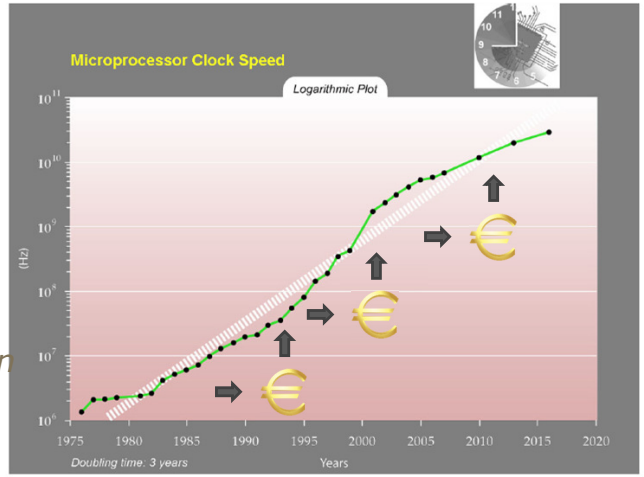
- *Maar…*

Ook aan de VUB wordt er onderzoek gedaan naar een optische computer. Ik zie echter twee grote problemen waardoor deze waarschijnlijk nooit op de markt zal komen.

Probleem 1: Electrische computer heeft al een hele ontwikkelingsproces doorgemaakt

Een alternatief moet wedijveren met de huidige processorkracht , de ontwikkeling van deze werd over de laatste 40 jaar echter gefinancierd door de verkoop van minder krachtige processoren.

Een alternatief heeft deze mogelijkheid niet … ze moet onmiddellijk wedijveren met de elektronische comuter



Aanverwant thema: de beste technologie is niet de beste optie voor de markt

- De eerste iPhone of iPad die Apple op de markt bracht was niet het beste wat Apple te bieden had, maar ze was wel goed genoeg om de markt te veroveren.
- Zo kon Apple later met een verbeterde versie uitpakken (bvb betere camera) en weer langs de kassa passeren…
  - Tip: wacht minstens tot 2e versie

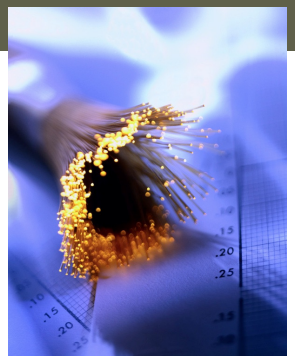
Probleem 2

Geen goedkoop, eenvoudig productieproces

- lenzen zijn foutgevoelig, niet te miniaturiseren

Elektrische computer heeft dat wel, gebruik makend van micro-elektronica

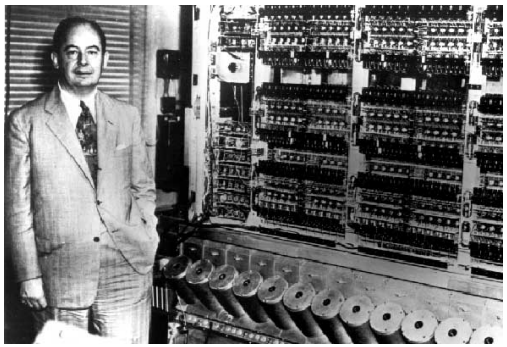
Wel: communicatienetwerken gaan via licht

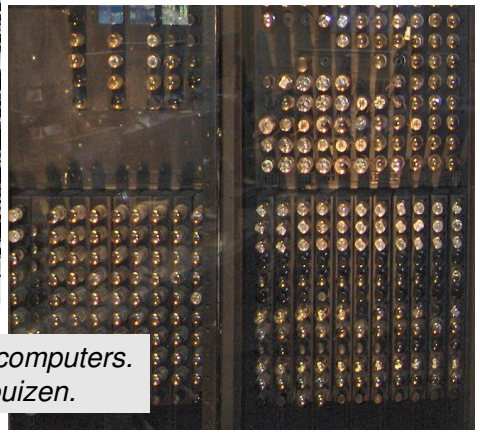


- **Glasvezel**
- The world's cable map: http://www.cablemap.info/
- Ook voor communicatie in computersystemen

Optische technologie wordt wel gebruikt voor communicatie van gegevens.

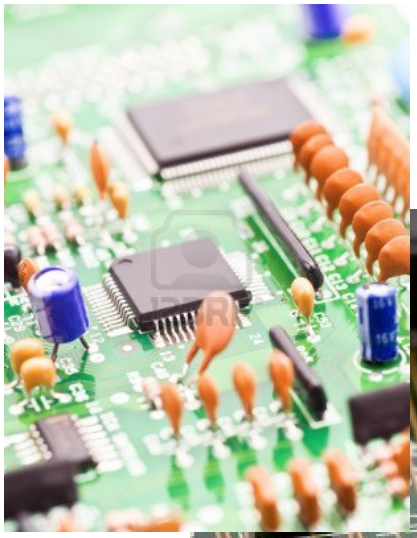
De doorbraak van de computertechnologie



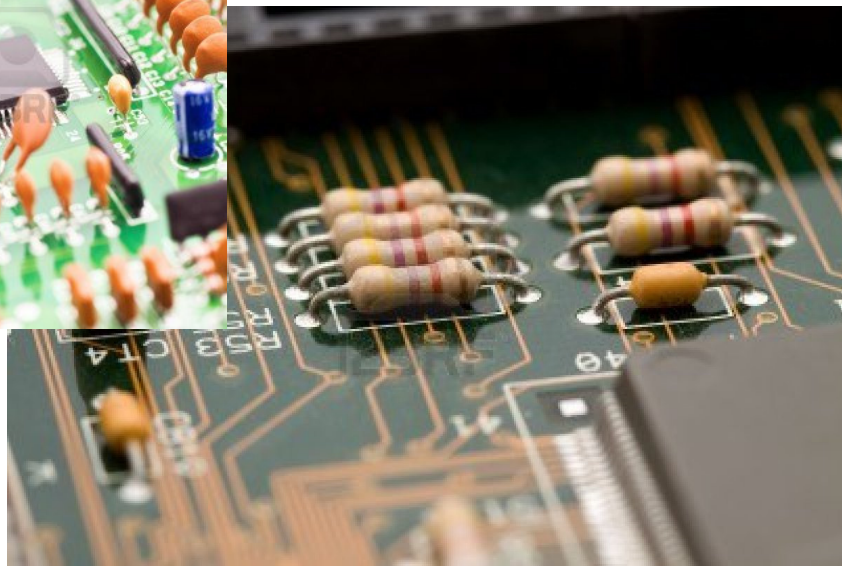


Von Neumann met één van de eerste computers. Rechts de achterkant met de vacuümbuizen.

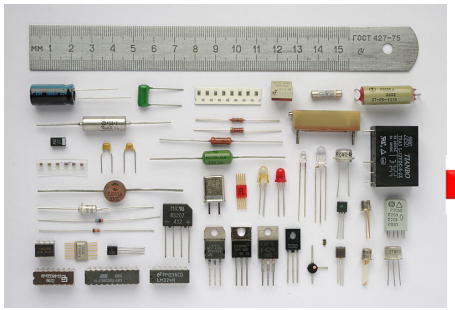
De vacuümbuizen worden vervangen door chips

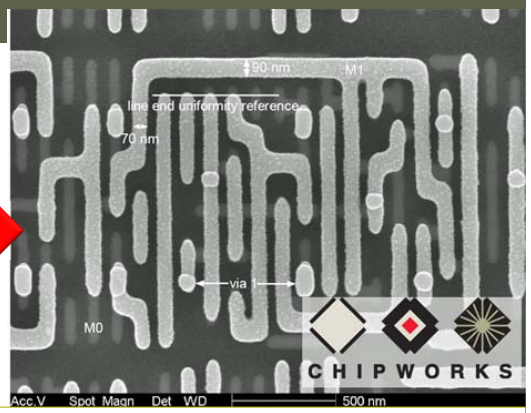


Componenten op een bordje met ingebakken connectielijnen



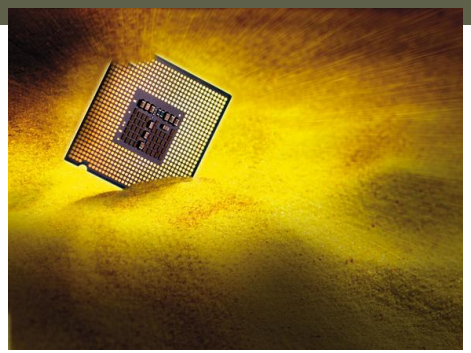
Alle componenten op een **chip**



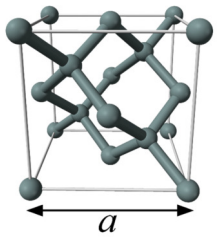


In plaats van losse componenten die geconnecteerd moeten worden, wordt er 1 plaatje gemaakt waarvan delen 'gemuteerd' zijn tot geleiders of halfgeleiders. Zo worden er weerstanden, capaciteiten en transistoren 'ingebakken' in het plaatje alsook de geleiders die de componenten connecteren. Er wordt gestart met een 'wafer' van silicium (element Si – atoomnummer 14). Vervolgens wordt er mbv mallen geëtst in het wafer en worden er andere elementen geïnjecteerd (dopering genoemd) zodat een geleider of halfgeleider ontstaat.

Zand…

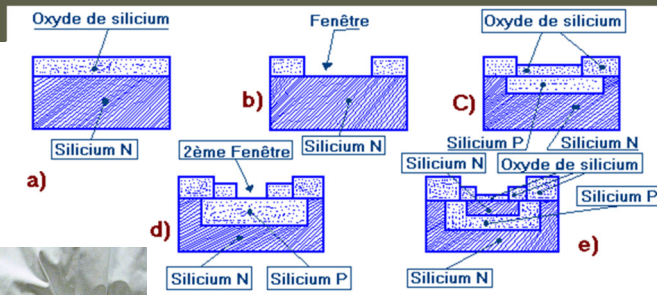


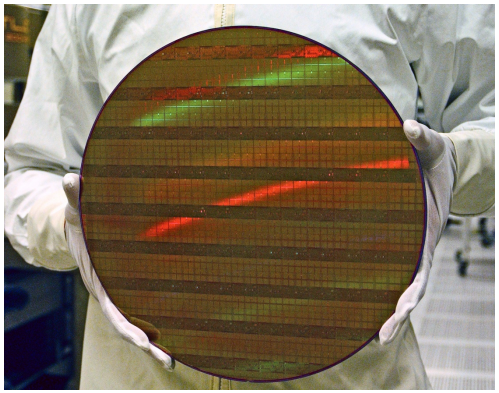
bevat het wonderlijke Silicium, de basis van de chip



a = 0.5431nm

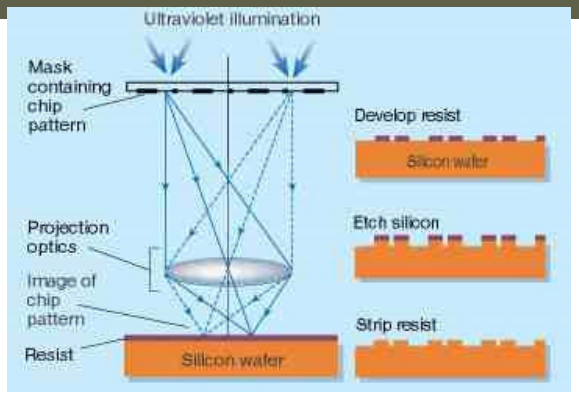
Silicium-’wafer’



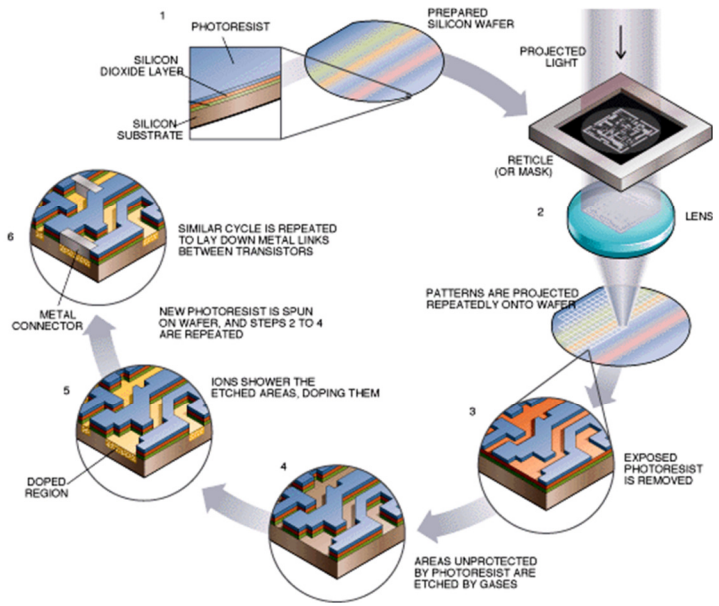


Etsen en doperen met geleidend en halfgeleidend materiaal

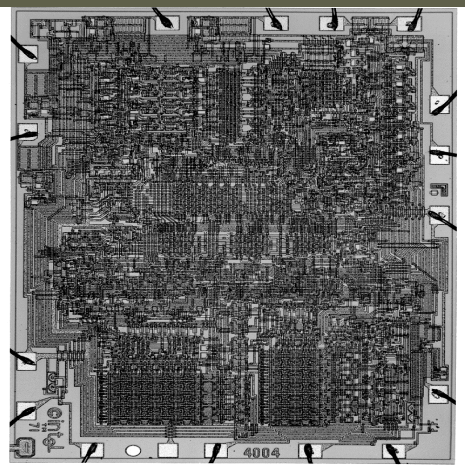
Etsen van wafer adhv mal (mask)



- ‘Eenvoudig’ productieproces
- te miniaturiseren tot 10 nanometer



Een ‘oudtje’: de Intel 4004 microprocessor (1971)

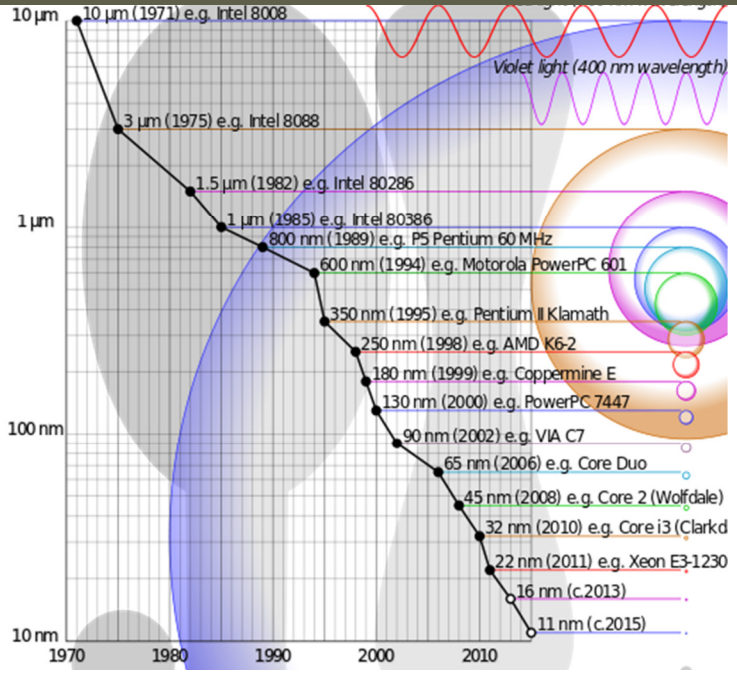


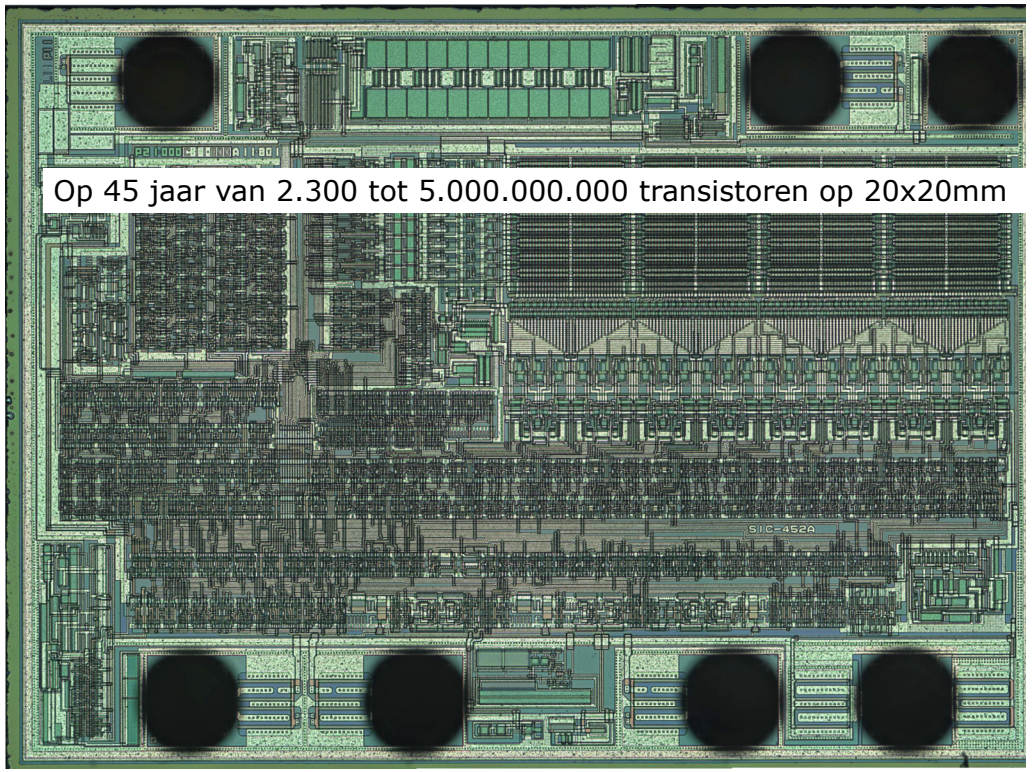
1000 transistoren op 1MHz klokfrekwentie

Intel was de eerste die een werkende processor kon maken op een chip. Intel werd opgericht door drie personen met totaal verschillende karakters: Moore (de uitvinder, naar ‘binnen gekeerd’), Noyce (de verkoper, naar ‘buiten toe’) en Grove (de manager, man van de actie).

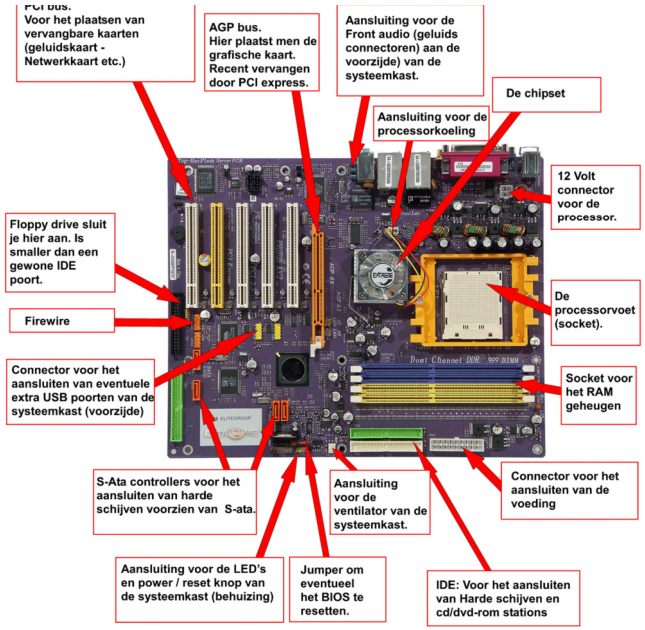
Miniaturisatie

Breedte van de geleidende lijntjes op de chip

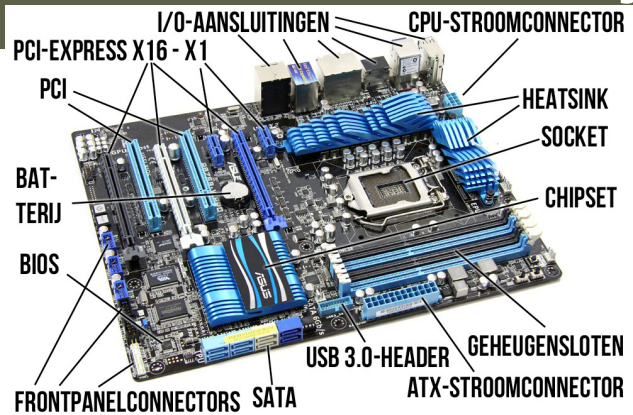




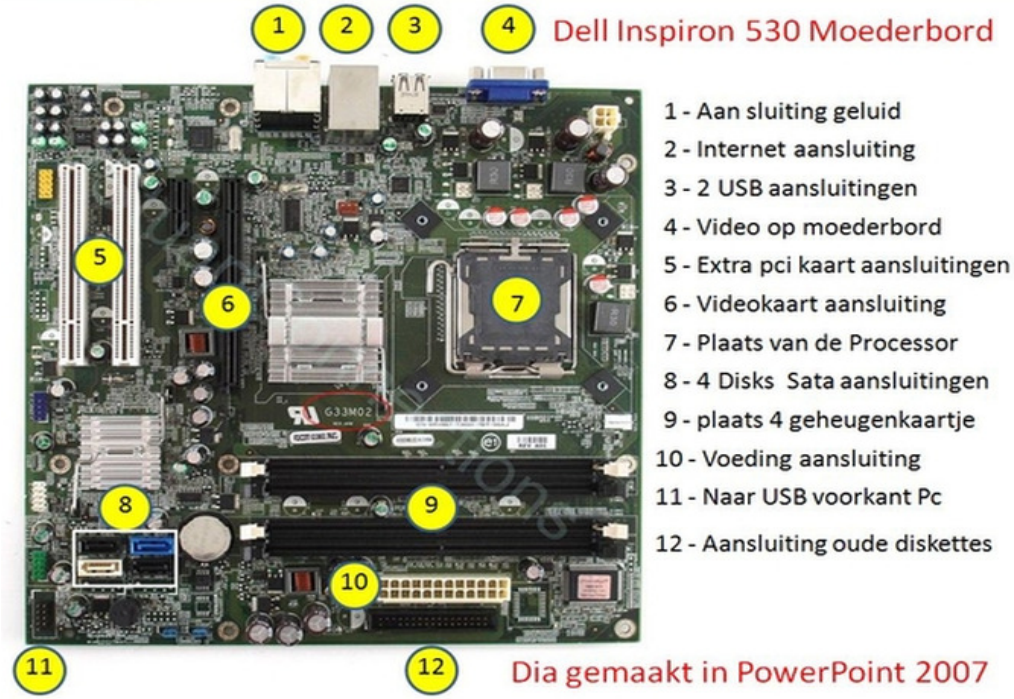
Op 45 jaar van 2.300 tot 5.000.000.000 transistoren op 20x20mm



Moederbord van je PC



De processor bevindt zich op het moederbord van je PC (schema: in de socket), waar het geconnecteerd wordt met het RAM-geheugen (schema: geheugensloten) en de connecties naar de buitenwereld (muis, toetsenbord en USB, PCI voor andere devices). Verder: de 'heatsink' neemt de warmte weg, de BIOS bevat de basisinstellingen voor de opstart van je computer.

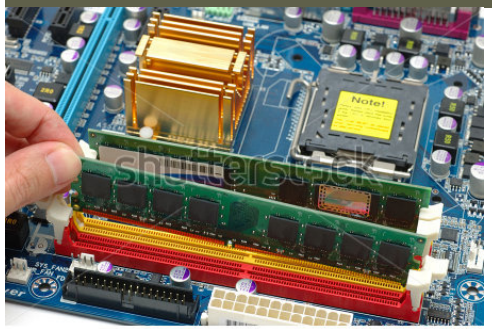


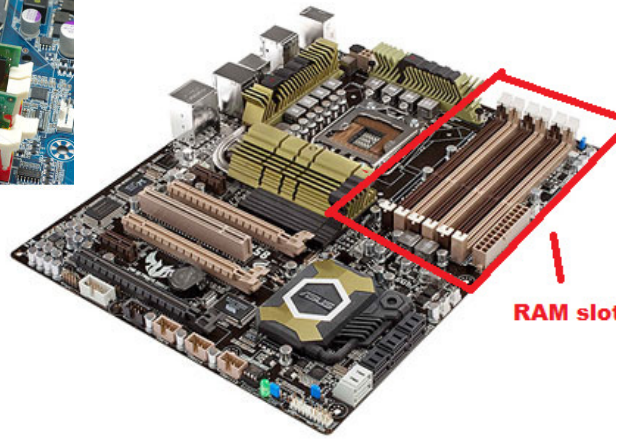
Dell Inspiron 530 Moederbord

1 - Aan sluiting geluid
2 - Internet aansluiting
3 - 2 USB aansluitingen
4 - Video op moederbord
5 - Extra pci kaart aansluitingen
6 - Videokaart aansluiting
7 - Plaats van de Processor
8 - 4 Disks Sata aansluitingen
9 - plaats 4 geheugenkaartje
10 - Voeding aansluiting
11 - Naar USB voorkant Pc
12 - Aansluiting oude diskettes

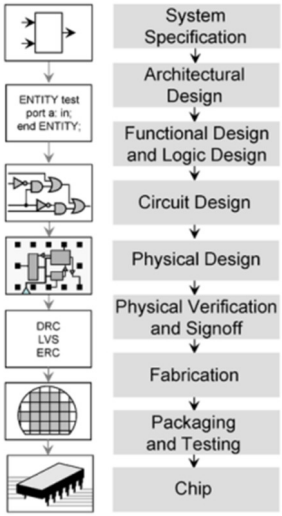
Dia gemaakt in PowerPoint 2007

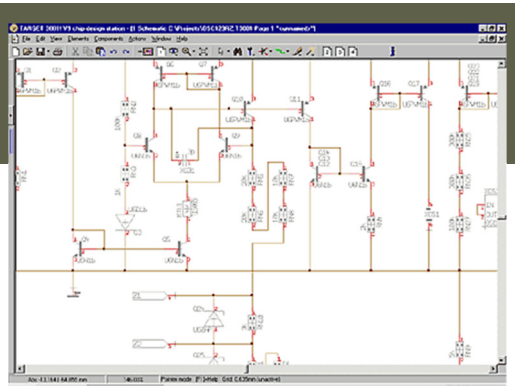
Zelf toevoegen van RAM-geheugen

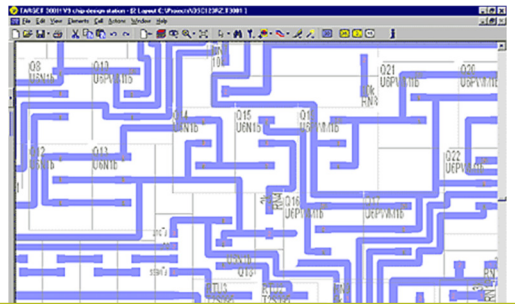




Chip design



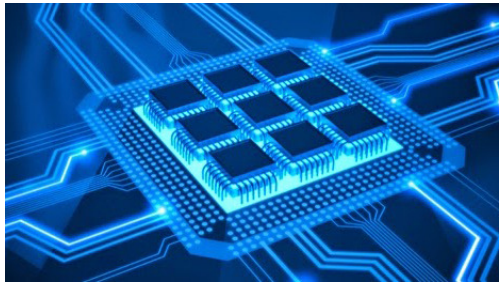


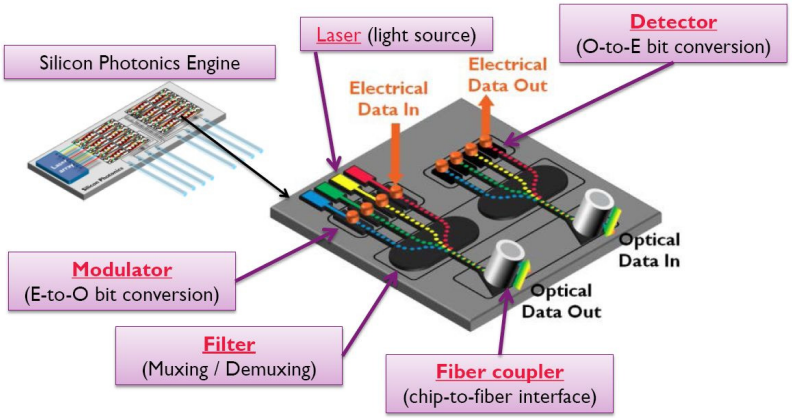


Het aanmaken van een chip gebeurt volautomatisch vertrekkende van een ontwerp van het electronisch circuit. Voor het ontwerp bestaat modern software.

Wèl toekomst voor licht in de computer

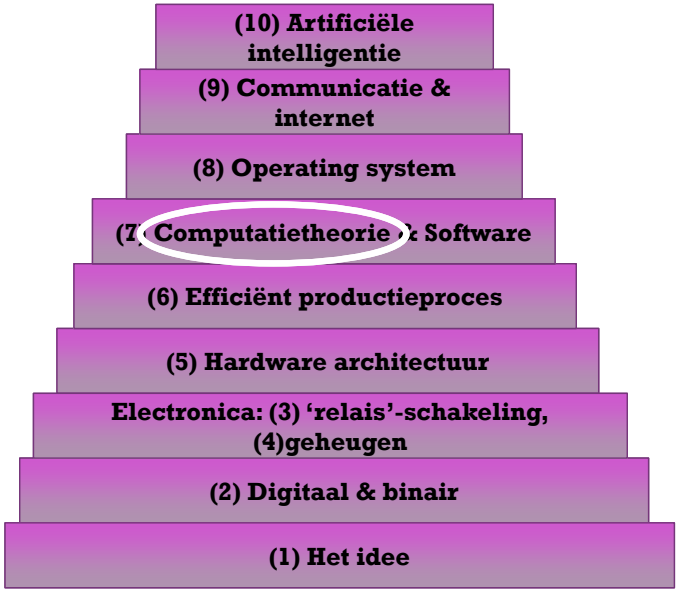
Photonic Integrated Circuit: geen lichtprocessor, maar wel licht die in **dezelfde chip** interageert met de electronen.
Kan wel niet zo klein gemaakt worden: de processor blijft werken met elektronen, de comm



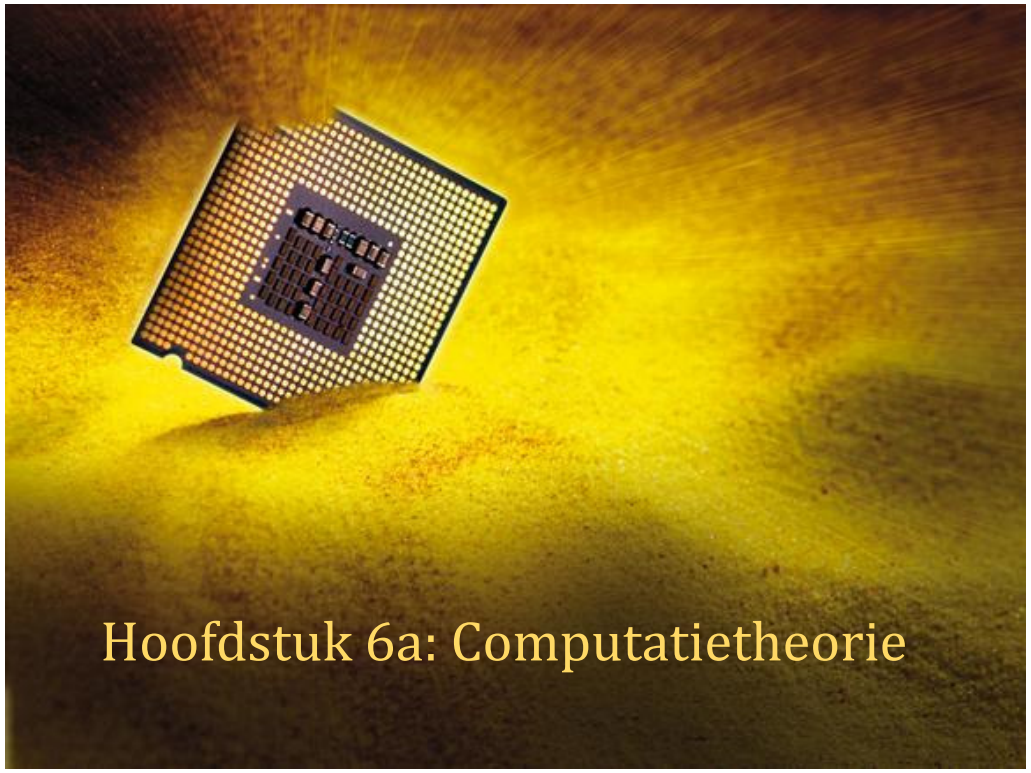


Interuniversitair Micro-Elektronica Centrum is het grootste onafhankelijke Europese onderzoekscentrum op het gebied van micro-elektronica, nanotechnologie

Waarmaken van Leibniz's droom



Informatica deel III: technologie, historiek en economische aspecten



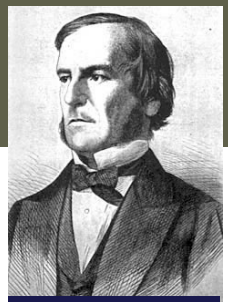
Leibniz' droom

1646 – 1716

- De "**Calculus ratiocinator**"
  - Een logisch denkend apparaat
- Om met de regels van de logica ondubbelzinnig te kunnen vaststellen of een statement waar of vals is
- Deductief systeem waarin alle regels afgeleid zijn van een kleine verzameling axiomas

Leibniz was één van de eerste die begreep dat als je filosoferen, denken, redeneren, … in formele regels kunt gieten, je dit ook door een apparaat kunt laten doen.

George Boole



1815 – 1864

- Logica herleid tot Booleaanse algebra

Boole zette een belangrijke stap met het ontwikkelen van een algebra voor het 'rekenen' met statements over waar/onwaar.

Gottlob Frege

1848 – 1925

- Beschrijft hoe wiskundigen en logici denken
- 1879: *de universele taal van de logica*
- 1903: brief van Betrand Russel die hem op een onvolledigheid wijst
  - Extra-ordinaire verzameling = verzameling dat een element is van zijn eigen
    - Vb: "verzameling van alle dingen die geen spreeuw zijn"
  - *Maar*: de verzameling ***E*** van alle ordinaire verzamelingen
    - En wat is ***E***? Ordinair of extra-ordinair

Eind 19e eeuw dacht men dat alle natuurwetten binnen handbereik waren (positivisme). Zo begreep men met de Maxwell-vergelijkingen electriciteit en magnetisme. De relativiteitstheorie van Einstein en de kwantummechanica maakte een einde aan dit optimism in de fysica. Maar ook in de logica bleek niet zo iets te bestaan als een mooie verzameling regels die niet tot contradicties leidt.

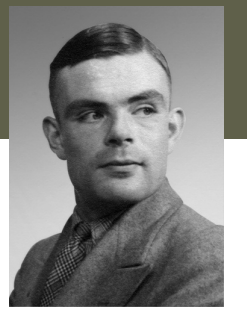
David Hilbert

1862 – 1943

- "Wir mussen wissen; wir werden wissen"
  - We moeten weten; we zullen weten
- Elke logische of mathematische vraag is oplosbaar (en zal opgelost worden)
- *Nodig*: een expliciete procedure om vanuit premises na te gaan of een conclusie volgt of niet (Hilbert's **beslissingsprobleem**)

Hilbert's beslissingsprobleem zou leiden tot een doorbraak in de theoretische informatica.

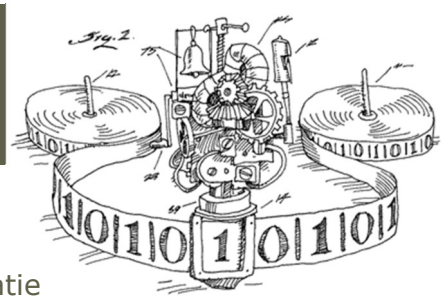
Alan Turing



- Hilbert's procedure: **algoritme**!
- Als we een 'mechanische' lijst van regels hebben om de oplossing van een mathematisch probleem op te lossen, zouden wiskundigen geen werk meer hebben…
- Bestaat er zo'n algoritme?
- Om tegendeel te bewijzen moest Turing een machine maken dat een algoritme kan uitvoeren

Alan Turing wierp zich op Hilbert's probleem. Hij bedacht dat er hiervoor een algoritme gemaakt moest worden (die nagaat of een statement klopt of niet), en vervolgens een machine die het algoritme kan uitvoeren. De 'Turing machine' werd geboren, een theoretisch model van een computer.

De Turing machine



- **Data**: oneindige lange tape voor het lezen of schrijven van 0/1/spatie
  - Kan 1 vakje naar links of rechts bewogen worden
- **Staat (toestand)**: een aantal binaire variabelen
- **Gedragsregels:**

Staat	Gelezen bit	Nieuwe staat	Te schrijven bit	Bewegen
0	0	0	0	>
0	1	1	1	<
1	0	1	_	>
1	1	HALT		

Deze machine is alleen maar van theoretisch nut. Ze is bijvoorbeeld niet gemakkelijk programmeerbaar.

Voorbeeld

- **Applet van studenten 2012**

  http://parallel.vub.ac.be/education/java/studentenprojecten/Turing Machine.html
  - "5-state B.B.": met input '_11' stopt machine, voor andere inputs niet

- **Java applet:**

  http://ironphoenix.org/tril/tm/

Oplossing van Hilbert's **beslissingsprobleem?**

- Met de machine nagaan of een logisch statement waar/vals is:
  - Programma runnen, als de machine stopt is statement bewezen (waar).
  - Je moet dus te weten komen of het programma ooit zal stoppen
- '**Halting problem**' (tegelijkertijd ontdekt door Alonzo Church en Alan Turing)
  - Gegeven een Turing machine en programma, ga na of dit programma zal stoppen of oneindig lang zal doorgaan.
  - We hebben een ander algoritme nodig die dit nagaat.
  - Turing bewees dat zo'n algoritme niet bestaat.
- ➡ *Hilbert's beslissingsprobleem is onoplosbaar*

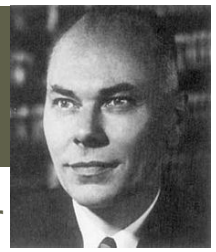
Het resultaat van Turing was weer een knak voor het optimisme van de wetenschap. Je kan niet van alles bewijzen of het waar of niet waar is. Hieruit onstond mede de moderne relativistische kijk op de wereld. De wetenschap kan niet alles oplossen/verklaren… De VUB blijft echter geloven dat wetenschap veel kan oplossen. **Scienta Vincere Tenebras**!

De universele computer

- Elk algoritme kan er op uitgevoerd worden = Universele Turing Machine
- Kan alles berekenen wat berekenbaar is
  - Rekenen, maar ook om logisch te denken…
  - ➡ Universeel
- Als je met een andere computer deze computer kan simuleren, is hij ook universeel
  - ➡ Turing bewees de gelijkwaardigheid van computers

Turing's resultaten blijken wel belangrijk voor de informatica. Misschien wel het belangrijkste resultaat van Turing is het idee van een universele computer. Eén computer (hardware) die alles kan berekenen wat maar te berekenen valt. Dit was revolutionair, zeker omdat velen in die tijd niet konden geloven dat je met 1 machine totaal verschillende berekeningen kan doen. Alsof je met een wasmachine ook kan autorijden.

Misvattingen omtrent computers



- Howard Aiken, bouwde de eerste computer voor IBM in 1944 en zei in 1956:

  "*If it should turn out that the basic logics of a machine designed for the numerical solution of differential equations coincide with the logics of a machine intended to make bills for a department store, I would regards this as the most amazing coincidence I have ever encountered*"
- Studie van 1947:

  "*slechts zes elektronische digitale computers zullen voldoende zijn om al het rekenwerk van de gehele Verenigde Staten te kunnen doen*."

Basisprincipe

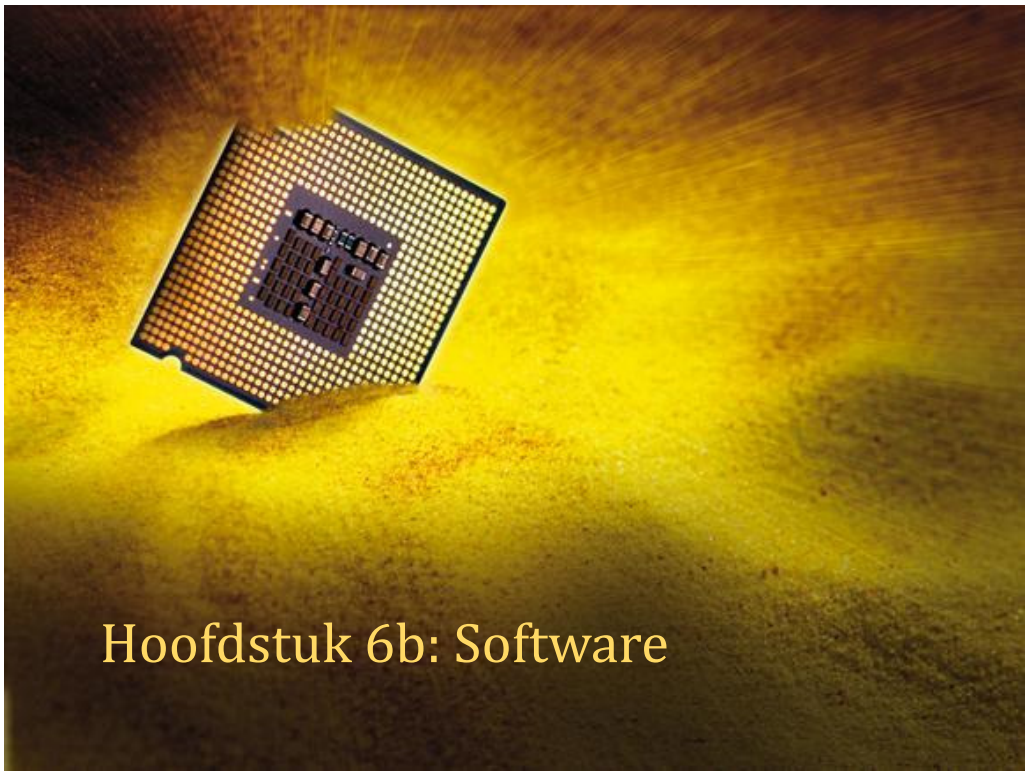
- 1 universele computer kan elke andere computer simuleren
- Zo is je programma ook maar data die zegt wat de computer moet doen. Voor dat programma heb je geen aparte machine nodig, toch?
- Nogal logisch, niet?
- Dit kan je ook toepassen op programmeertalen: je wilt een taal waarmee je alle mogelijke algoritmes kan programmeren, een universele programmeertaal.

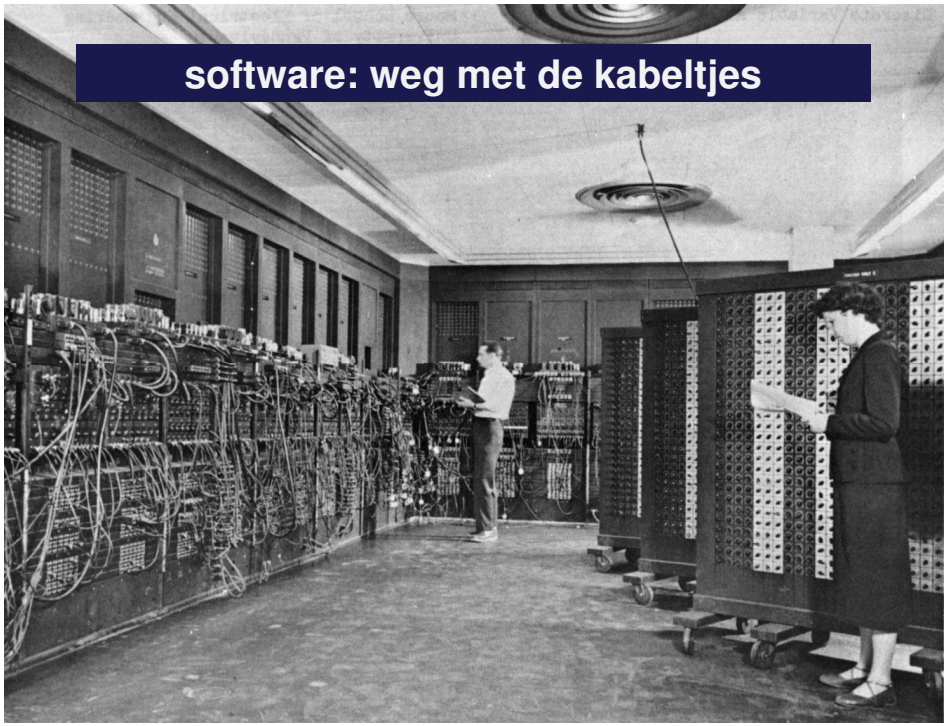
Als je met 1 computer een andere kan simuleren (nabootsen), kan je alles doen wat die andere ook kan. Als die andere universeel is, is de jouwe ook universeel. Hiermee bewijs je de equivalentie van universele computers. En ook de compleetheid van programmeertalen. Java & Python zijn complete talen, je kan er alles mee programmeren wat je maar kan bedenken.

Computatietheorie: bewijzen dat een programma correct is…

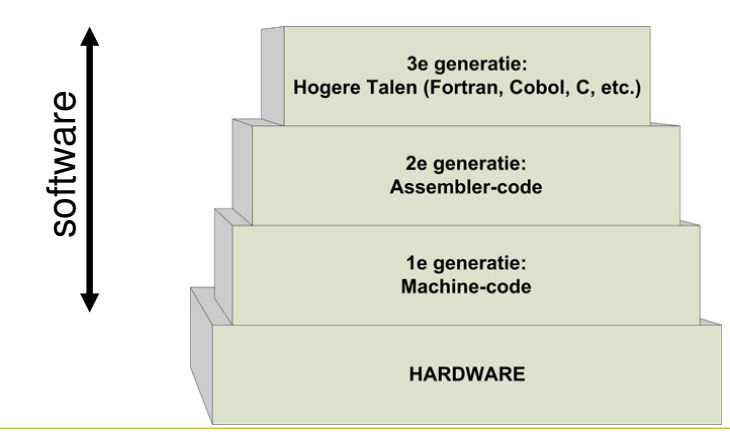
- Heel omslachtig!
- Enkel mogelijk voor eenvoudige algoritmes.
- *Dus*: ad-hoc testen noodzakelijk, zorgvuldig programmeren, goed opdelen, …
- Maar: belangrijk! Denk maar aan zelfrijdende autos kerncentrales of satelieten: weten dat de software correct werkt en nooit crasht.

Anderzijds zijn er nog vele gaten in de theoretische informatica. Zo zou het toch handig zijn om te bewijzen dat je programma correct is. Dit blijkt heel omslachtig te zijn, zelfs voor simple programma's. Informatica blijft dan ook gedeeltelijk een 'praktische' wetenschap. Maar wij ingenieurs malen daar toch niet om, he? Als het maar nuttig is, die computer.



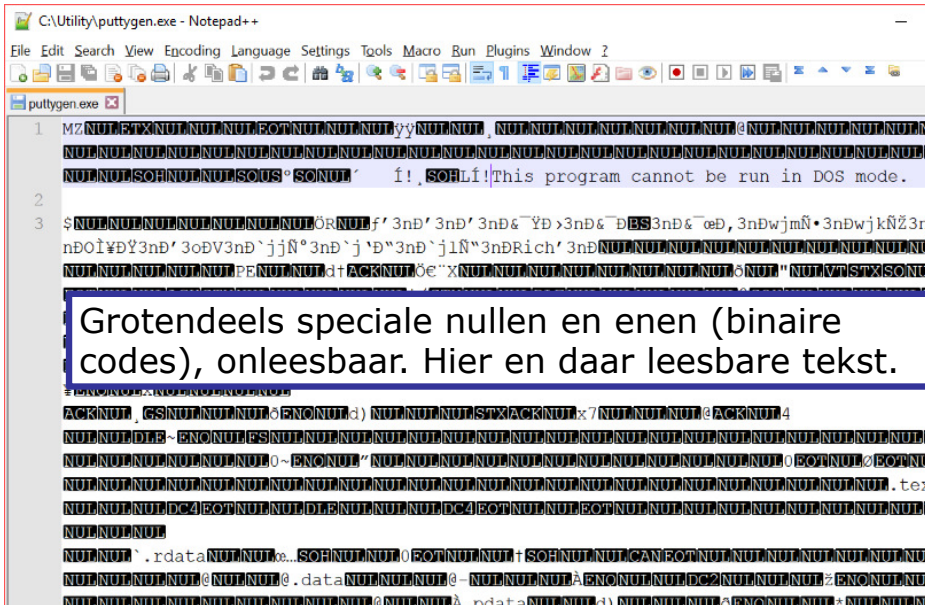


Programmeertalen

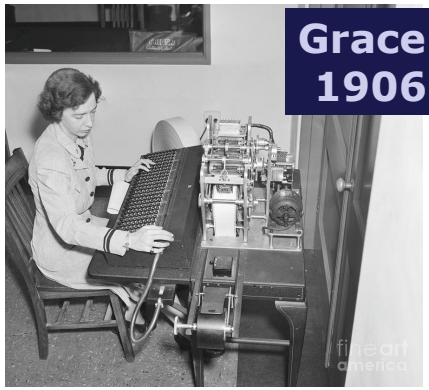


Een processor biedt een aantal instructies aan die hij begrijpt en kan uitvoeren. Deze instructies zijn binair en noemt men *machinecode*. Voor de leesbaarheid biedt *assembler* mnemonische afkortingen aan voor de binaire instructies, alsook variabelen en labels van codelijnen. De assembler zet alle afkortingen om naar binaire machinecode die dan uitgevoerd kunnen worden.

Executable = Machinetaal



Machine-onafhankelijk programmeren



Ontwikkeling programmeertalen die op elke machine uitgevoerd kunnen worden. De programma's worden daarna omgezet in specifieke machinetaal. Grace Hopper was hiervoor de grondlegger met de taal COBOL, die nog steeds voorkomt!

Voorbeeldprogramma (zie hfst 4)

```
Scanner scanner = new Scanner(System.in);
System.out.print("Geef een getal:");
int x = scanner.nextInt();
int m = x;
int t = 0;
do {
      System.out.println(m);
      m = m * x;
      t++;
} while (m < 1000 && t < 20);
```

Machinetaal bestaat uit 3 soorten instructies

Hoofdstuk 4

1. Berekeningen en logische bewerkingen
   - Berekeningen: optellen, vermenigvuldigen, bitwise operaties
   - Logisch: vergelijken, and/or/xor/not
2. Controle-instructies
   - De program counter wordt verhoogd/verlaagd: het programma gaat dus niet verder met de volgende instructie, maar springt naar een andere positie
3. Lezen en schrijven van gegevens
   - Van en naar geheugen
   - Van input en naar output

In assembler (javacode vanaf lijn 3)

```
     lw  $t0, $s0      # load word
     mv  $t1, $t0       # move value
     mv  $t2, 0
l1:  sw  $t0, $s1      # store word
     mul $t1, $t1, $t0 # multiply
     add $t2, $t2, 1   # addition
     lt  $t3, $t1, 1000 # less than
     lt  $t4, $t2, 20   # less than
     and $t5, $t3, $t4 # and
     beq $t5, 1, l1    # jump to l1 if equal
```

Lezen van toetsenbord en schrijven naar scherm gebeurt door te lezen en schrijven naar geheugenadressen $s0 en $s1. Variabelen x, m en t staan in registers $t0, $t1 en $t2. Voor de conditie hebben we 3 extra registers nodig ($t3, $t4 en $t5). 'l1' is een label van een instructie, om er later naar toe te kunnen springen. Resultaat van een bewerking komt in eerste operand.

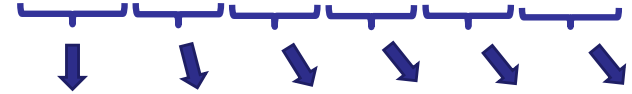
Machinetaal (binair)

```
add $t0, $s1, $s2
```

assembler

00000010001100100100000000100000

machinetaal



000000	10001	10010	01000	00000	100000
special	$s1	$s2	$t0	0	add
0	17	18	8	0	32

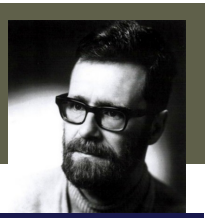
1 instructies = 32 bits met specifieke betekenis

3e generatietaal

- Onafhankelijk van de hardware
- ≈ beschrijving van het algoritme
  - Gebruik van variabelen ipv geheugenadressen
  - Geen JUMP, enkel for/while/if/switch
    - Jumps leidt tot spagetticode, je mag van overal naar overal springen
  - Functies/methodes mogelijk
- Omzetting naar assembler: *compileren*
  - Checkt je code op syntaxfouten!

Programmeren in assembler is niet produktief. Het is redelijk omslachtig, je maakt snel fouten en de code is heel onleesbaar. Daarom heeft men programmeertalen ontwikkeld die minder code vergen, meer aansluiten bij algoritmes, meer gestructureerd zijn, leesbaarder en minder foutgevoelig. De Nederlander Edgser Dijkstra was het voornaamste genie achter de 3e generatietalen.

Edsger Dijkstra (NL)

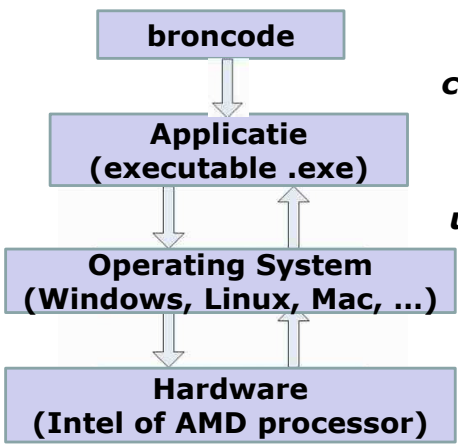


1930 – 20(

Welke constructies moet een programmeertaal minimaal bevatten?

- **While/for**
  - **Goto** is onnodig. In oude talen kon je zeggen 'spring naar lijn 111', naar elke willekeurige lijn van je programma. Dit leidde echter tot onoverzichtelijke 'spaghetti'-code. Dijkstra toonde aan dat een while voldoende is
- **Functies**
  - In het begin waren er om praktische redenen veel beperkingen op het gebruik van functies.
- **Recursie**
  - Hier was veel tegenkanting tegen, omdat het praktisch tamelijk 'duur' is (vergt veel van de processor)

Programma => computer



Hogere taal

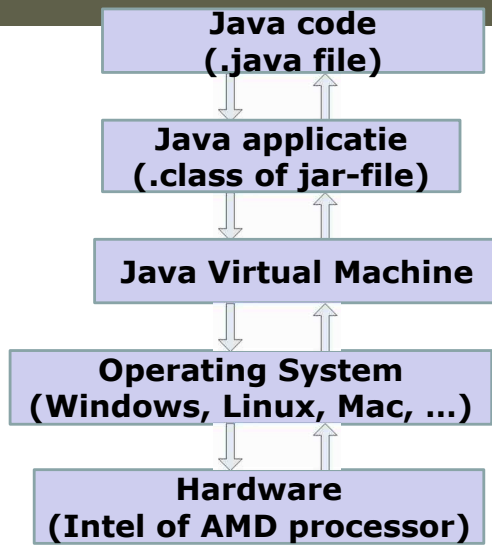
compileren

Machinetaal

uitvoeren

Nieuwe compilatie nodig voor elk systeem (operating system + hardware)!

Java: via virtuele machine



compileren

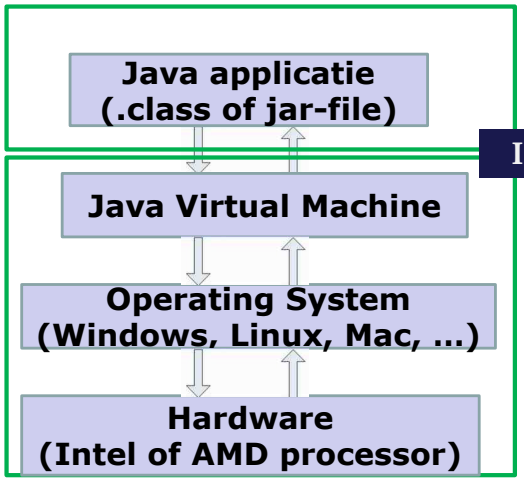
Specifieke tussentaal die iets hoger is dan assembler

uitvoeren

Tijdens het uitvoeren wordt instructies van de class-files omgezet naar machinetaal

Javacode werkt op elk operating system!
Je moet wel de virtuele machine installeren

Ideaal voor internet



Jou code staat op **webserver**

Via virtuele machine runt hij op elke **lokale machine ('client')**

Doordat Javacode via een virtuele machine (die wel specifiek is voor elk systeem) tijdens het uitvoeren wordt omgezet naar machinetaal, kan je programma overal uitgevoerd worden.

Python & Mathlab

- ***Geïnterpreteerde talen***
- Je code wordt tijdens het uitvoeren omgezet naar machinetaal.

Je moet Python/Matlab dus geïnstalleerd hebben

- Javacode wordt wèl gecompileerd, *java-files* worden omgezet naar *class-files*.
  - Javacode wordt dus gecontroleerd op fouten. Maar Python en Matlab niet, waardoor je programma zal crashen bij fouten met de types van variabelen bijvoorbeeld
  - Java Virtual Machine gebruikt een **Just-in-time compiler** om toch optimale machinecode te verkrijgen
    - Pypy is een just-in-time compiler voor Python

Snelheid matrixmultiplicatie

Product van twee 200x200 matrices

	runtime (ms)	faster?
C (executable)	**8,4**	
java	**28**	**0,30**
python	**10822**	**0,0008**
python met numpy library functie	**11,5**	**0,73**
Python met just-in-time compiler pypy	**82**	**0,10**
matlab	**172**	**0,049**
matlab library functie	**0,713**	**11,8**
gpu (intel)	**0,375**	**22,4**
gpu (AMD tahiti)	**0,161**	**52,2**
gpu (nvidia geforce GTX 650)	**0,077**	**109,1**

4e generatietaal

- High-level specification languages: programmeren *wat* je wilt, niet *hoe* het moet gebeuren
  - Vb: GUI aanmaken met een tool

    Te integreren in Eclipse
    - **Google Windowbuilder Pro**
    - **Visual Editor**

    Genereert java-code die GUI aanmaakt

(Nog) niet echt succesvol

Enkel voor specifieke applicaties (zoals GUIs)

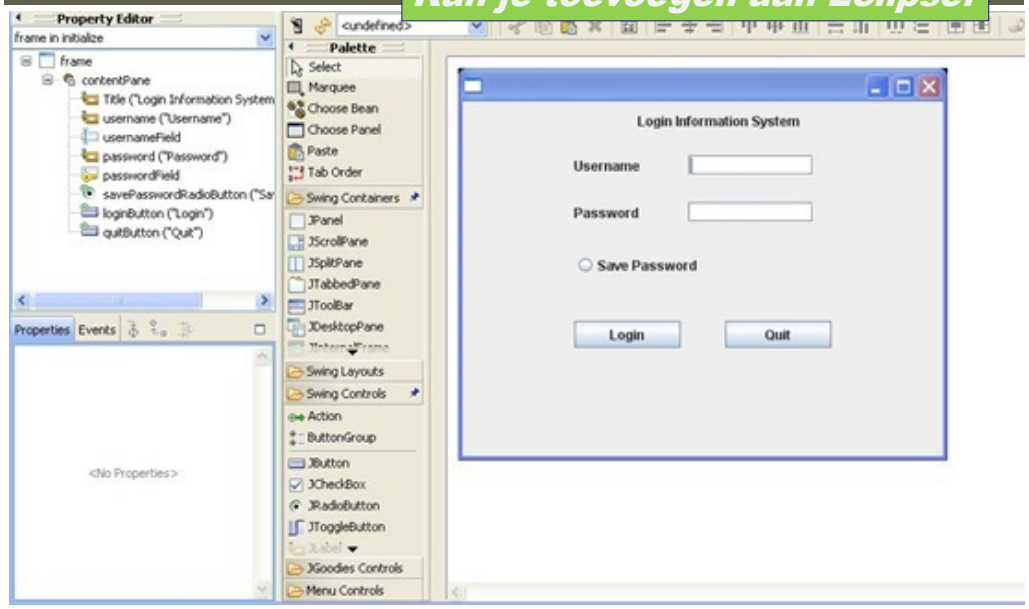
- Mijn mening: nog geen artificiële intelligentie

=> computer kan nog niet begrijpen wat je wilt

Een GUI aanmaken kan via een tool waarbij je niets moet programmeren. Je bouwt als het ware je applicatie door aan te geven hoe deze er uit moet zien. Dit werkt echter nog niet voor algemene applicaties…

Tool om GUIs te maken

Kan je toevoegen aan Eclipse!

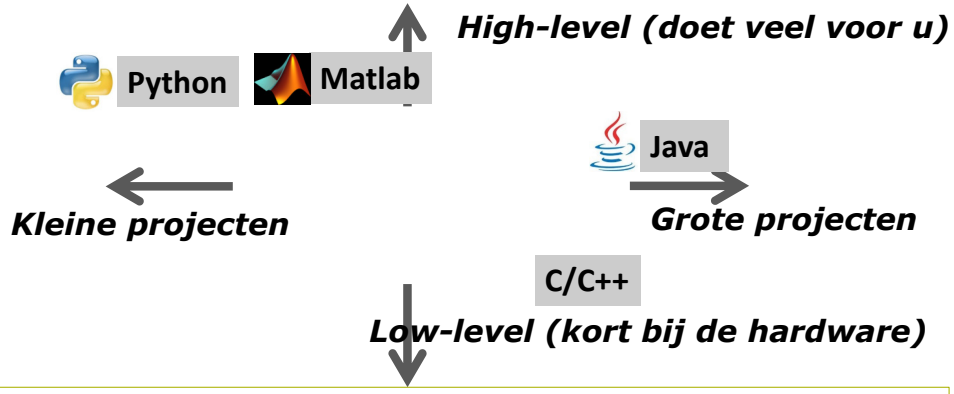


Object-georiënteerde talen

- Worden als 3e generatietalen aanzien
- Toch is de code-organisatie fundamenteel verschillend
  - De code is opgesplitst in klasses ipv functies.
  - Vergt een andere manier van denken
  - Is dè uitdaging van dit semester, zowel voor jullie als voor ons (lesgevers)

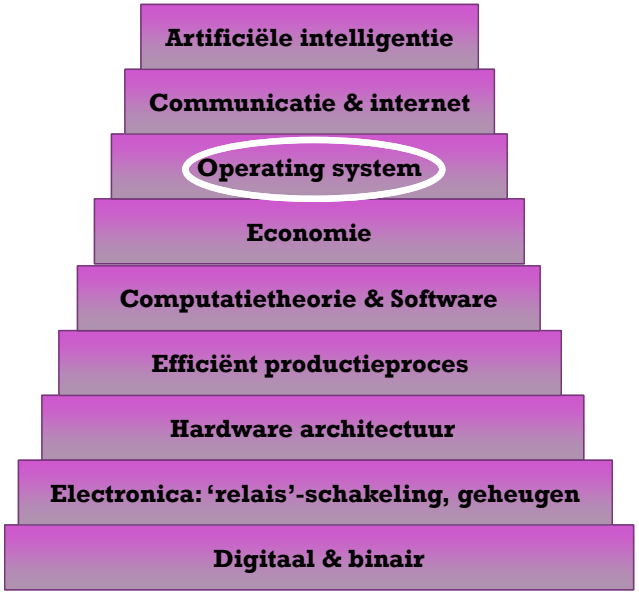
Voor ingenieur: welke taal?

moet handig en efficiënt zijn, geschikt voor de taak

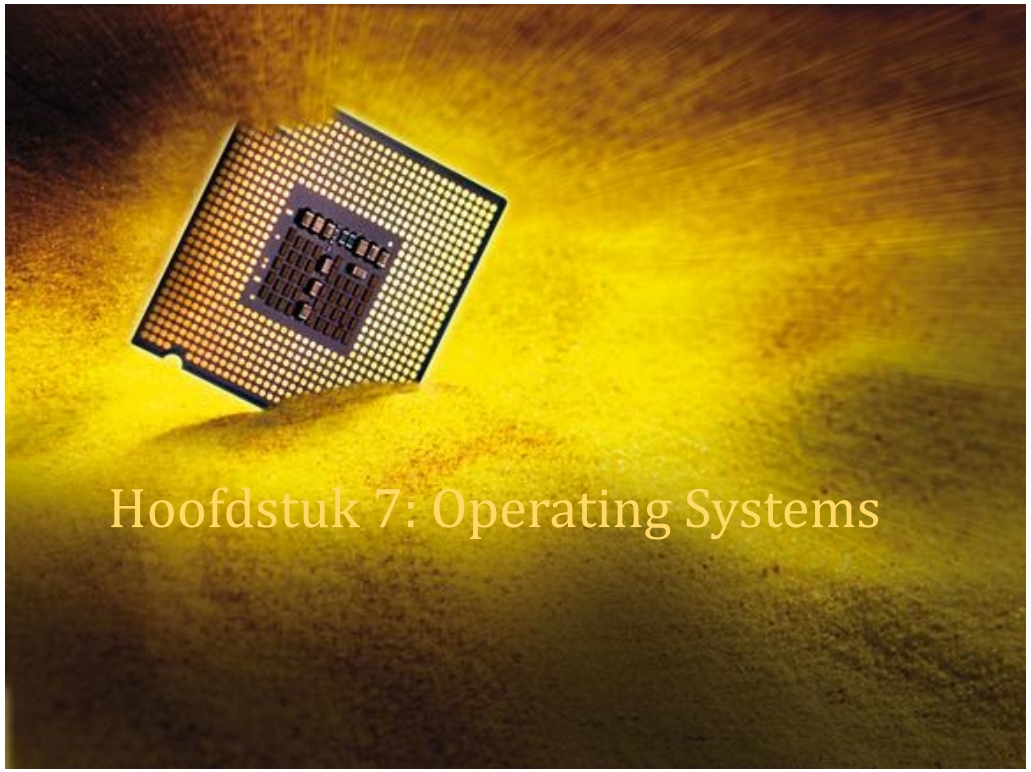


Zonder al te diep in te gaan over wat nu de beste taal is (een eeuwige steeds-hoog-oplopende discussie onder informatici), in dit schema een kleine poging tot conclusie. C/C++ zullen velen later nog zien, het is een veel-gebruikte taal die kort bij de hardware staat, daarvoor uitermate geschikt voor het programmeren van systemen (chips, robots, kritische applicaties, …)

Waarmaken van Leibniz's droom



Informatica deel III: technologie, historiek en economische aspecten



Bedrijfscomputers

Initieel waren er voornamelijk bedrijfscomputers

- Centrale computer
  - "mainframe" systemen
  - Voornaamste fabrikant: **IBM**

Interactiviteit via *Terminal*

- Scherm van 24 lijnen van 80 tekens groot

Gebruikers werken op de centrale computer via 'domme' terminals die enkel het scherm tonen en de input van de gebruiker (via toetsenbord) doorgeven.

Toepassingen mainframe

- Gegevens (bvb boekhouding) van banken, bedrijven, winkels etc
- Gegevens worden bijgehouden in een *database*
  - *Database = gestructureerd bijhouden van gegevens*
- Eigenschappen mainframe:
  - *Betrouwbaar (heel belangrijk)*
  - *Robuust (crasht bijna nooit)*
  - *Veilig (security) van gegevens (bvb bankgegevens)*

1970s: IBM gaat voor Personal Computer

- Computer voor "thuis"
- Kan op eigen kracht werken (niet geconnecteerd met centrale computer)
- De PC is geboren!
- IBM: op dat moment het grootste informaticabedrijf
  - Concentreert zich op hardware

De PC ontketent een nieuwe revolutie in de informatica… ontwikkeld door IBM, maar in feite was het Rank Xerox die de meeste uitvinden hebben gedaan (de muis, de GUI, ...). Zij konden het echter niet omzetten in succesvolle producten, zij bleven bij de kopieermachines...

Microsoft Albuquerque Group, December 7, 1978. Top row: Steve Wood, Bob Wallace, Jim Lane. Middle row: Bob O'Rear, Bob Greenberg, Marc McDonald, Gordon Letwin. Front row: Bill Gates, Andrea Lewis, Marla Wood, Paul Allen. Missing from photograph are Ric Weiland and Miriam Lubow. Photo courtesy Microsoft Archives.

IBM heeft Operating System nodig

- IBM gaat langs bij Bill Gates en zijn 'hippie'vrienden
- Zitten thuis te programmeren
  - Bill Gates en Paul Allen waren er al sinds hun 16e van overtuigd dat software de toekomst is!
- **MicroSoft** is geboren
- Steken snel DOS (Disc Operating System) in elkaar
- Nog steeds terug te vinden in Windows
  - Cmd-window
  - Programma's start je met commando (en eventueel argumenten – dit zijn de "String[] args" van de main)

- *Commando-based:*

Je geeft commando's

in de terminal of shell


```
                    Welcome to MS-DOS 2011

For a list of simple commands, please type HELP
>HELP

LOC - Displays the location of the currently operating program
OPEN <directory> - Opens and displays contents of the directory entered
EXECUTE <directory/filename> - Executes a file within that directory
REBOOT - Reboots current session (Confirmation will be required to prevent
accidental reboots)
SYS - Displays system properties
>RUN: person .EXE

What was that? I can't understand you.
>realityconfig

Attempting to configure your reality...
Enter password:
>^C to abort.

Incorrect password. Exiting function.
>SYS

Main Harddrive: Q:/
Space Used: 105246/6152374 KB
Processing Power: 10 GH
RAM: 3000000 KB
>OPEN GAMES

What are you, stupid? GAMES isn't a directory! Try Q:/!
>EXECUTE adventure.EXE

Try adding a directory to that, before you seem any less intelligent.
>_
```

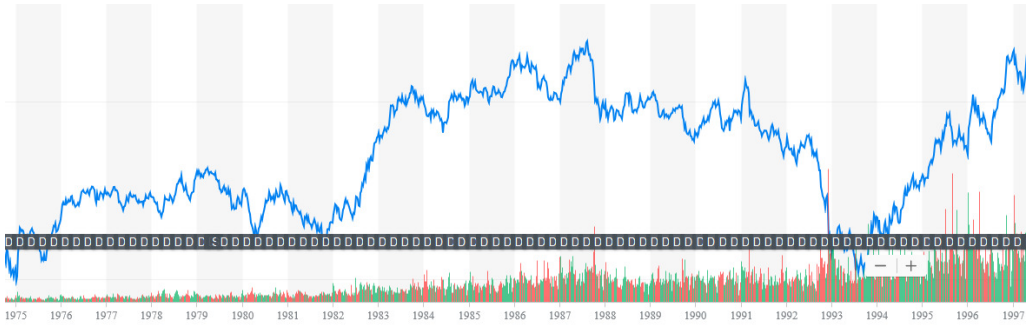
Bill Gates wordt rijkste man ter wereld

Macht ligt vanaf nu bij software en vooral het besturingsysteem

1984

IBM maakt historische vergissing door op hardware te blijven mikken
Ze mist de softwareboot compleet…

IBM



Lancering PC

Microsoft wint het pleit

IBM heeft definitief hegemonie verloren

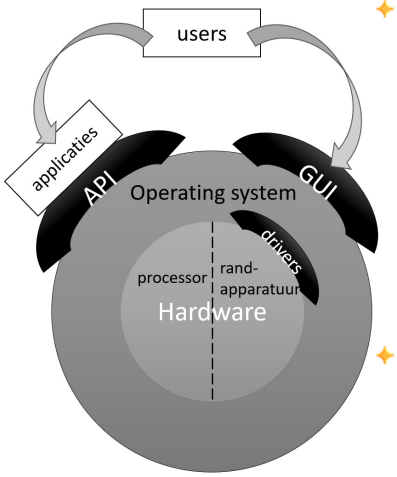
Besturingsysteem

- = Operating System (OS)
- OS regelt en organiseert de computer
- OS wordt van harde schijf gestart bij het *booten*
  - BIOS zorgt hiervoor

Een besturingssysteem (*Operating System* of afgekort OS) is een programma (meestal een geheel van samenwerkende programma's) dat na het opstarten van een computer in het geheugen geladen wordt en alle mogelijkheden van de computer aan de gebruiker aanbiedt. Het OS biedt ook de functionaliteiten aan om andere programma's - applicaties genoemd - uit te voeren.

Application Programming Interface (API)

- OS verstopt de details van de hardware voor de gebruiker en voor applicaties dmv een API



  - Deze worden op een uniforme wijze aan de applicaties aangeboden. De API abstraheert de toegang tot de verschillende randapparatuur, zonder OS moet elk programma zelf instaan voor het aansturen van randapparatuur (zoals printer, beeldschermen, harde schijf). Een gebruikersprogramma is enkel afhankelijk van het OS, niet van de randapparatuur.
  - Het communiceren van het OS met het randapparaat gebeurt via een *driver* (die wel specifiek is voor elke apparaat zoals printer).

Interactiviteit

- Geen interactiviteit: batch programma
  - Programma en gegevens worden op voorhand klaargemaakt
  - Tijdens de uitvoering kan je niet interageren met het programma
  - Resultaten worden op het einde als geheel gepresenteerd
- Commando-gebaseerd
  - Cf DOS, linux shell
  - De commando's worden 'geïnterpreteerd' en het programma gestart
  - Je kan een lijst van commando's doorgeven ('batch')
- Grafische User Interface (GUI)
  - Windows, muis, toetsenbord, touch screen

Hoofdtaken OS (vervolg)

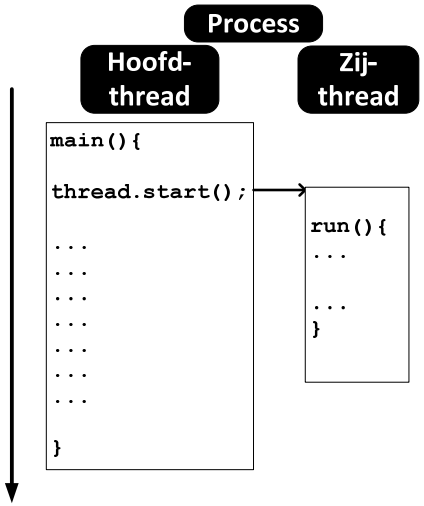
- Verdelen van toegang tot systeembronnen (RAM-geheugen, opslag, printer etc.) tussen de aktieve programma's
  - Elk programma krijgt een deel van het werkgeheugen toegewezen (java: standaard 64MB, je kan dit hoger instellen)
  - OS voorkomt dat een programma buiten zijn eigen deel gegevens kan lezen of schrijven (beveiliging!)
- Aanbieden gegevens (files) en applicaties aan gebruiker
  - OS beheert het filesysteem (georganiseerd in een boomstructuur dmv folders)
- Verdelen van processortijd over de aktieve programma's
  - Zie verder

Task Manager

- Windows: start via Control-Alt-Delete
- Toont actieve *applicaties* en *processen*, alsook processorgebruik
  - Applicaties: van gebruiker
  - Processen: naast de processen van de applicaties, ook processen en achtergrondprocessen ('services') van operating system en applicaties
- Operating System verdeelt cycles van processor (CPU) over de verschillende processen (*process scheduling*)

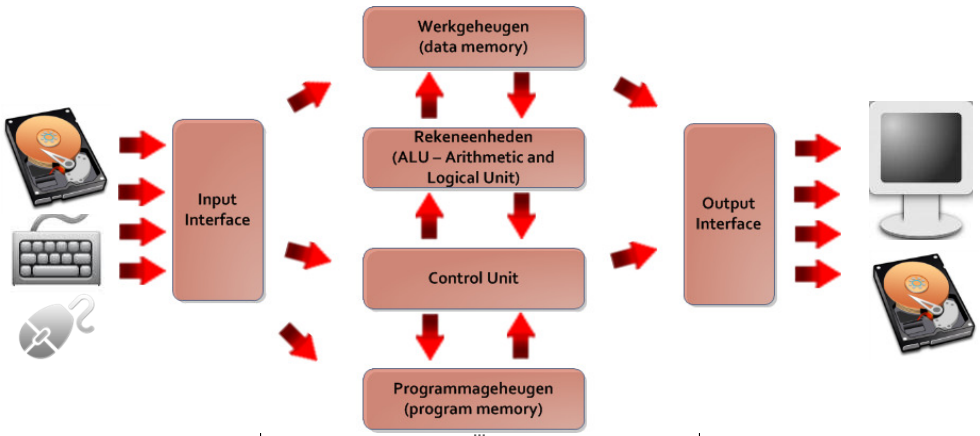
Op een modern OS kan je meerdere programma's tegelijk draaien. Het uitvoeren van een programma resulteert in een proces. Een achtergrondproces is in feite ook een gewoon programma, maar ze is niet zichtbaar voor de gebruiker. De achtergrondprocessen zorgen voor het beheer van het systeem of bieden services aan (zoals het sharen van je (muziek-)files en het checken van je mailbox).

Threads van processen



Elk programma dat uitgevoerd wordt komt overeen met 1 onafhankelijk *proces*, maar elk proces kan bestaan uit meerdere *threads*. Elke thread voert een sequentie van instructies uit gespecifieerd door de code. Een sequentie kan je zien als een 'draad', vandaar de benaming. Vele moderne toepassingen zijn zelf opgebouwd uit een aantal threads die simultaan uitgevoerd worden. Het tekstverwerkingsprogramma *Word* bv. gebruikt verschillende threads om teksten en figuren op het scherm te tonen, zodanig dat wanneer men snel door een tekst wenst te lopen men niet hoeft te wachten op het tekenen van alle figuren. Een andere thread zal tegelijkertijd je taalfouten opsporen (de *spelling checker*). Threads van eenzelfde proces werken dus met dezelfde gegevens, terwijl elk proces zijn eigen gegevens heeft (word document, excel sheet, mail, chat, …).

Processor kan maar 1 instructiesequentie tegelijk uitvoeren



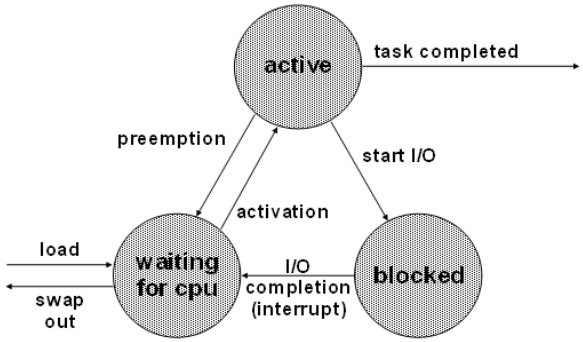
23	Lees x van toetsenbord
24	Initialiseer macht m = x
25	Initialiseer teller t
26	Schrijf macht m
27	Verhoog macht: m = m * x
28	Verhoog teller: t = t + 1

Process scheduler

De *process-scheduler* is het deel van het besturingssysteem dat op elk ogenblik bepaalt welk van de programma's toegewezen wordt aan de centrale verwerkingseenheid (CPU). Het hart van de computer, de processor of CPU, kan immers maar 1 programma tegelijk uitvoeren. De processor is immers opgebouwd volgens de *Von Neumann-architectuur*. Karakteristiek voor deze architectuur is dat hij één programma stap-voor-stap uitvoert. De CPU zal om beurten de programma's uitvoeren.

Moderne computers hebben intussen meerdere processorcores (dual core, quadcore, …). Dat zijn dan in feite 2, respectievelijk 4 onafhankelijke processoren die elk één proces tegelijkertijd kunnen uitvoeren.

De *process-scheduler* zal per core de processortijd verdelen over de lopende processen en threads: elk proces/thread krijgt een periode ('time slice') toegekend.



Toestanden van een proces

De lopende processen, kunnen zich, op elk ogenblik in drie toestanden bevinden:

1) *actief*: de centrale verwerkingseenheid (CPU) is aan het proces aan het werken; dwz. dat instructies van dat programma opgehaald worden door de stuureenheid en worden uitgevoerd.

2) *geblokkeerd*: een in- of uitvoeroperatie (Input/Output of I/O) is aan de gang en het proces moet wachten tot het einde ervan;

3) *wachtend*: het proces zou kunnen uitgevoerd worden, maar de centrale verwerkingseenheid is niet beschikbaar, ze is instructies aan het uitvoeren van een ander proces. Het proces moet wachten tot de proces-scheduler processorcycles ter beschikking stelt.

Overgang van 1 proces naar een ander

- Als het actieve proces op I/O (input/output) moet wachten
  - Dikwijls wordt bij I/O het OS geactiveerd, omdat die de I/O beheert (bvb toegang tot files controleert)
- Of als de toegekende 'time slice' die een proces toegekend krijgt op is
  - Bij starten van een proces wordt ook een timer gestart die na de time slice een *interrupt* geeft
  - *Interrupt* zorgt dat de processor met het actieve proces stopt en de scheduler van het OS wordt gereactiveerd. Deze beslist welk proces nu een time slice toegekend krijgt.
- Overgang naar een nieuw proces*: context switch*
  - de staat van het oude proces wordt opgeslagen en dat van het nieuwe wordt geladen

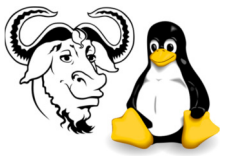
Interrupts

Met een *Interrupt* ('onderbreking') wordt een lopend programma onderbroken. Deze is voorzien in de hardware van de processor. Je kan immers niet verwachten dat een lopend programma zelf zal checken of hij verder mag gaan. Ook kan het OS er niet van uitgaan dat elk programma in een redelijke tijd stopt. Het OS moet op elk moment een programma kunnen onderbreken. Anders zou een 'oneindige lus' volledig beslag leggen op je processor zonder dat je er iets aan kan doen (behalve dan de computer herstarten).

Let op: het OS is ook niets meer dan een programma. Er is geen 'big brother' in je computer die toekijkt wat de processor doet. Controle over het actieve proces wordt verkregen door *interrupts.* Processoren bevatten een hardwareschakeling waardoor een programma kan onderbroken worden door middel van een *interrupt*. Vervolgens bepaalt de *interrupt handler* welk programma uitgevoerd moet worden. Aan de hand van de oorsprong van de interrupt veroorzaakt deze een sprong naar het gewenste programma. Als er bijvoorbeeld input binnenkomt (muisklik, internetbericht). Of voor de controle door het OS. Stel dat een programma een file wil openen, dan wordt het OS even geactiveerd om te controleren of het programma dat wel mag. Als de file nog open is in een ander programma bvb, wordt de toegang geweigerd.

Unix & Linux

- UNIX operating system: zoals mainframe initieel bestemd voor bedrijfscomputers
  - User moet inloggen
  - Heeft eigen files op *server* (in zijn *home*)
  - Enkel administrator kan dingen aan systeem veranderen
- Linux: Open Source-versie van UNIX
  - Open Source volgt de GNU-regels: de code mag vrij (gratis) gebruikt worden zolang er geen geld voor gevraagd wordt
- *Apple* gebruikt nu ook een Unix-versie
- *Android* van Google is java op Linux



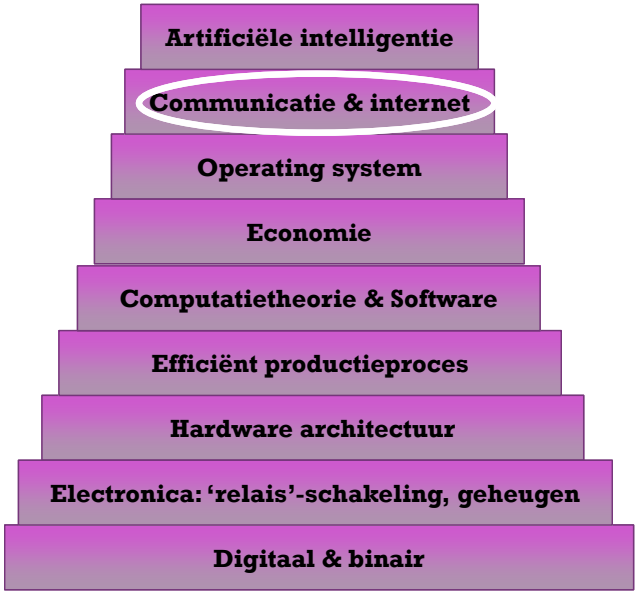
Conclusie: in de consumentenmarkt is het Windows of Linux.

Strategie: open versus gesloten

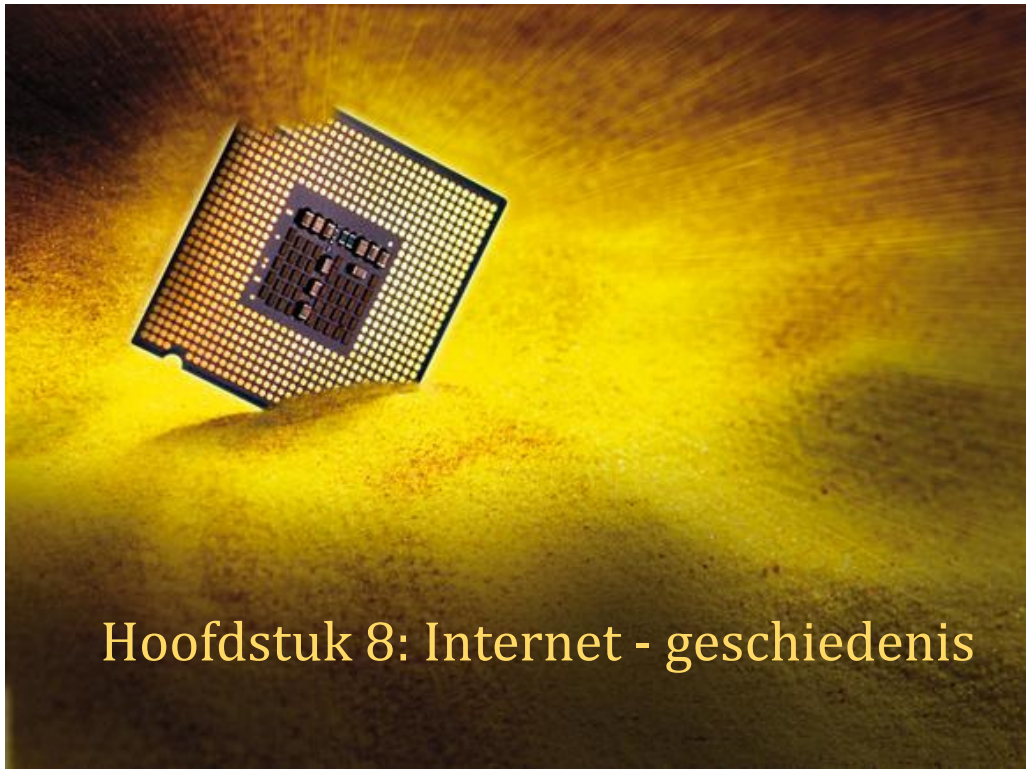
- **Microsoft's Windows**:
  - Open besturingssysteem
  - Iedereen mag er software voor ontwikkelen
    - Geeft andere bedrijven kansen
  - Microsoft concentreerde op besturingssysteem & software
    - niet op hardware en niet op alle software
- **Apple**:
  - Hield en houdt controle over het hele systeem, software & hardware
    - Werkte initieel tegen hun (eind '90 bijna failliet)
    - Via "apps" kan je software aanbieden
  - Pakt nu succesvol uit met 'totaalproducten'
  - Gebruiksgemak, stijl en design steeds prioritair

Analoog bij OS voor smartphones: Android versus Apple OS

Waarmaken van Leibniz's droom



Informatica deel III: technologie, historiek en economische aspecten



Geschiedenis: ARPANET

- TCP/IP ontstond uit Arpanet, ontwikkeld door Amerikaans leger
- Eisen aan digitaal communicatiesysteem:
  - **Flexibel**
  - **Gedecentraliseerd**, geen 'baas'
  - **Onafhankelijkheid**: delen kunnen op zich werken
  - **Zelf-regulerend**
  - ➡ **robustness and survivability**
    - including the capability to withstand losses of large portions of the underlying networks (due to a nuclear attack)
    - pakketjes kunnen verloren gaan: fouten of 'overlopen' van de queues bij bottlenecks

Het internet werkt compleet anders dan een traditioneel telefonienetwerk. Bij telefonie zijn er centrales die een exclusieve connectie reserveren tussen de 3 belpunten. Internet is gedecentraliseerd, gegevens worden in pakketjes opgedeeld die elk hun eigen route mogen volgen over gedeelde lijnen.

Microsoft vòòr 1995

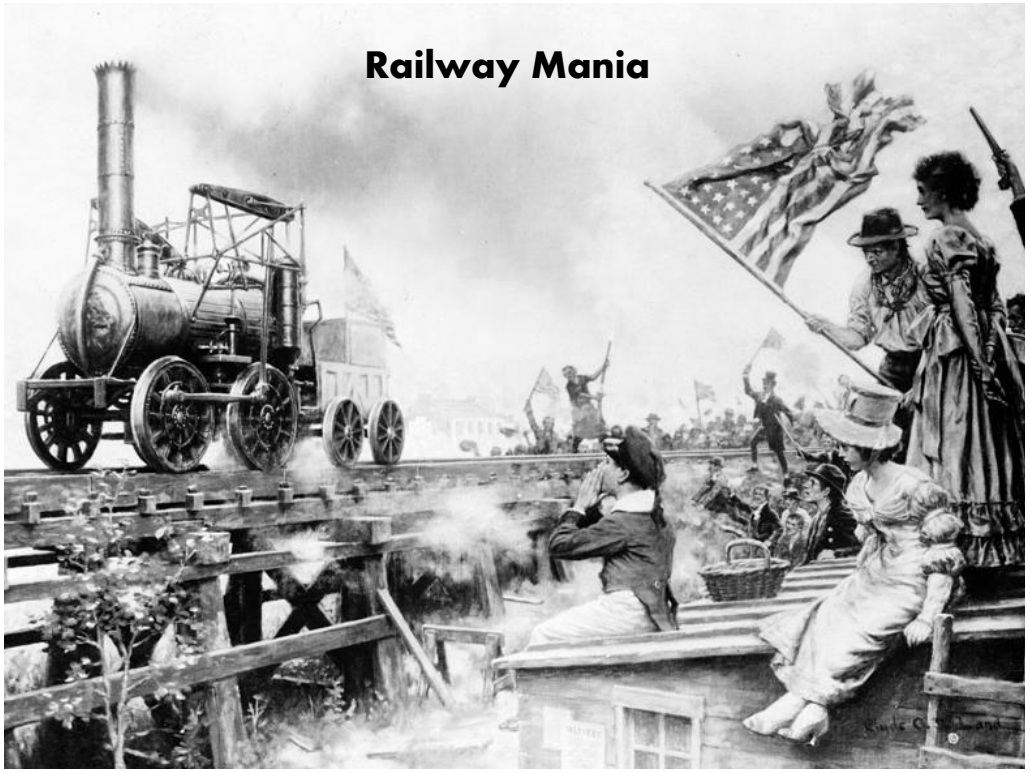
- Microsoft: door Windows, grootste speller in IT
- Maar is niet geïnteresseerd in internet (ik beroepsmatig ook niet)
- Op CERN wordt Netscape Navigator ontwikkeld, de *eerste browser*
  - Om informatie (file) van een remote computer te visualiseren
- Netscape wordt de grootste speler
  - Is nu Firefox/Seamonkey

De volgende revolutie werd…

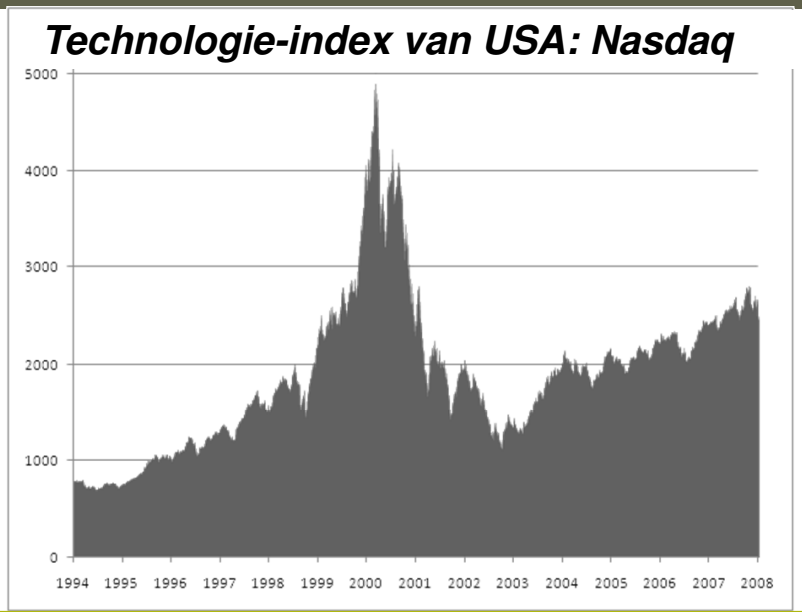
INTERNET!

Microsoft reageert

- Lanceert Internet Explorer in 1995
  - Gratis bij Windows waardoor iedereen het begint te gebruiken
- Koopt hotmail op
  - En onlangs nog Skype en Nokia (maar dominante positie lijkt nu definitief verloren)
- Verovert het internet
  - Door macht van Windows



De internetrevolutie



Begin 19e eeuw was er de spoorwegbubbel, eind 20e eeuw de internetbubbel.

2000 tot nu: + 55%

De internetbubbel

Of internetzeepbel of dotcom-crisis

- Nieuwe technologie opent nieuwe mogelijkheden, creëert (te) hoge verwachtingen

➡ **Investeren geblazen!**

- Professionals/leken steken hun geld in aandelenfondsen en aandelenclubs om de boot niet te missen
- Bubbel <= Hebzucht!

IT-sector: the winner takes it all

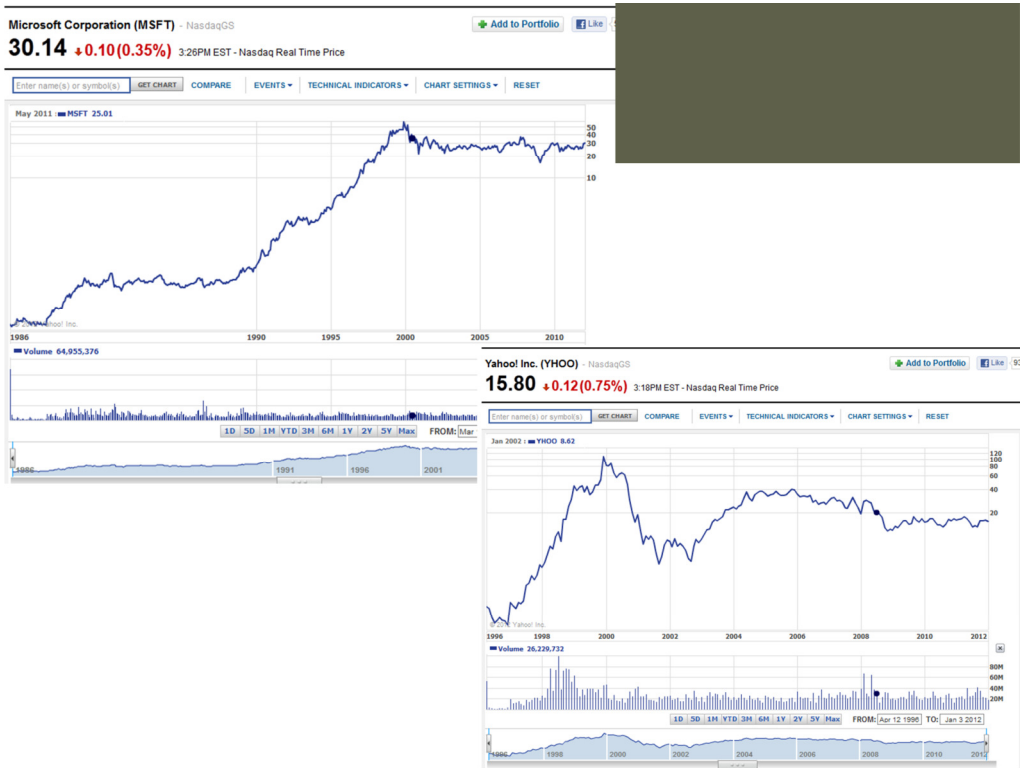
- Windows, office, TCP/IP-protocol, pdf, Google, facebook, twitter, …
- ⇔ in andere sectoren kunnen er meerdere spelers zijn (bvb meerdere automerken, wasmachines, …)
- Kan leiden tot monopolieposities en oneerlijke concurrentie (verstoorde marktwerking).
  - Microsoft kreeg verscheidene veroordelingen van de Europese Commissie
- *Alternatief*: standaards waar iedereen zich aan houdt

De IT-sector heeft zo zijn specifieke economische wetten. Eentje is dat het meestal 1 speler is die de markt verovert doordat de consument er alle baat bij heeft dat iedereen hetzelfde product gebruikt, wegens gemak van interoperabiliteit en uniformiteit. Dit wordt ook bereikt door standaards.

De economie zou totaal veranderen => 'dotcom'-economie

- Internet: ongekende commerciële mogelijkheden
  - *Anders communiceren*
  - *Anders kopen*
    - Ook kerstbomen kopen op het internet…
- Belangrijkste: =**aandacht** (hits/leden/…)
  - *"Get large or get lost"*
  - agressieve marktpenetratie door middel van het uitbouwen van netwerken.
- Extreme verliezen in het begin werden gezien als slechts investeringen. Winst/omzet maken zou later komen.

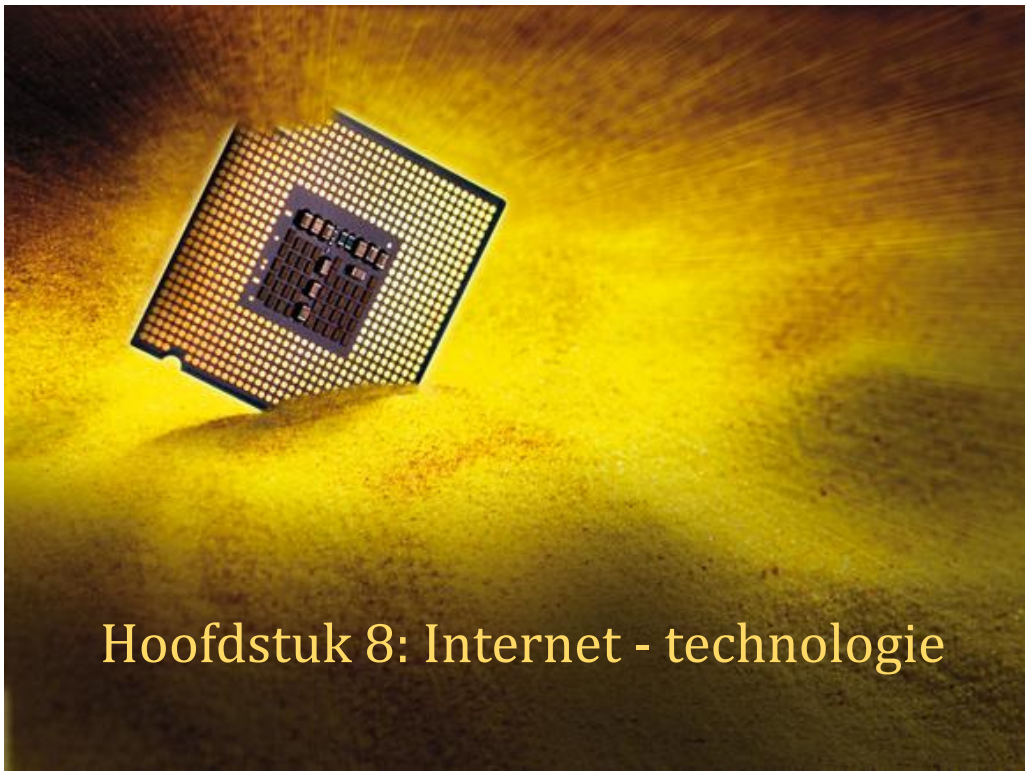
Velen geloofden dat we totaal anders zouden gaan consumeren. Je moest dus wel investeren in het internet, of je zou er binnen de kortste keren uit liggen.



Na het springen van de bubble (2000)

- Veel geld verloren
  - Investeerders en ook de 'gewone man', enkel de slimmeriken zullen er (net) op tijd uit gestapt zijn
- De droom spatte echter uit elkaar…
- ➡ Niemand wilt/durft meer te investeren in IT-technologie
- Tot… **Google** komt (2003-2004) en toont aan dat je wèl geld kunt verdienen op het Internet
  - Zie Nasdaq-index: begint weer te klimmen

Technologie heeft tijd nodig om te 'rijpen'.



Hoofdstuk 8: Internet - technologie

Technologie 1: netwerk

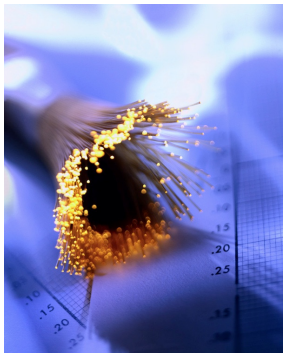
- **Lokaal network**:

electrische kabel

- **Glasvezel** verbindt

lokale netwerken

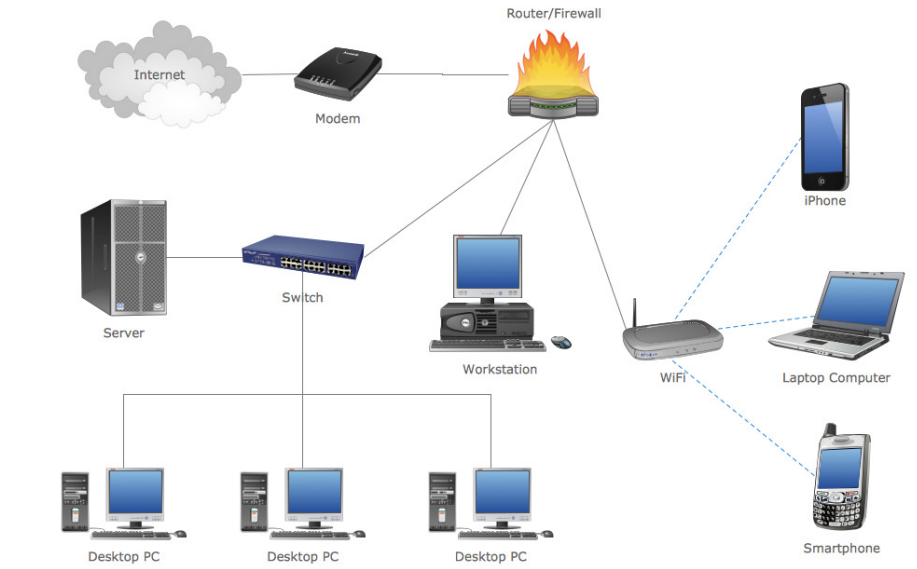
- ➡ informatie via licht
- The world's cable map:
  - **https://www.infrapedia.com/app (http://www.cablemap.info/)**



Technologie 2: componenten

- ***Netwerkkaart:*** toegang van je PC/laptop tot het internet, via een netwerkkabel of wireless
- ***Switch****:* netwerkknooppunt, gebruik je om meerdere connecties te verbinden
- ***Router****:* ook een netwerkknooppunt, maar bepaalt ook de route van het pakketje. Het is de toegang van een lokaal network tot het internet en het vormt de knooppunten op het internet
- ***Modem:*** maakt informatiesignalen geschikt om over een verbinding te worden getransporteerd, bvb wireless, telefoon- (Belgacom) of kabelnetwerk (Telenet)

Lokaal network (LAN)

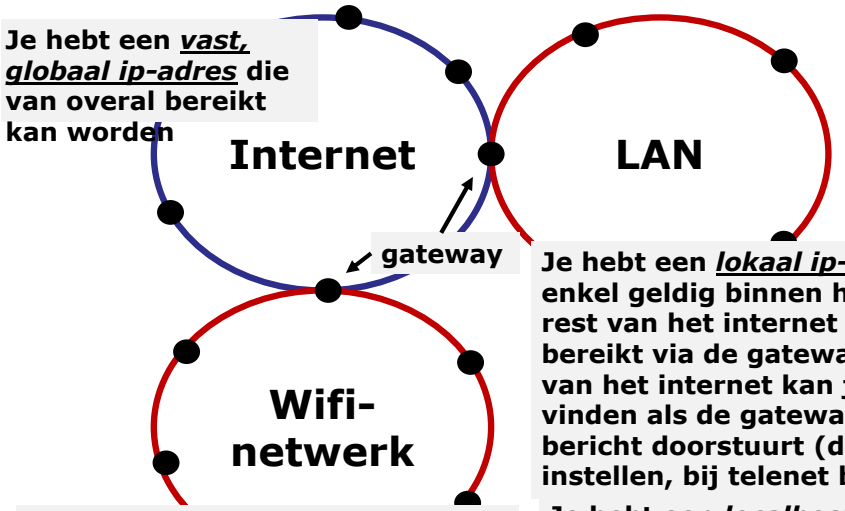


Technologie 3: protocol

- Protocol = afgesproken formeel communicatiewijze
- IP= Internet Protocol
  - Adres van elke computer: IP-adres
    - Bvb 192.168.0.233 in het nieuwe IPv6 is het een langer getal
    - Windows-commando *ipconfig* (start command prompt met *Windows-teken + cmd*) of onder *Performance -> Wifi* in de Task Manager
    - Domeinnaam is een *alias* voor het nummer ('naam') omzetting kan je doen via http://www.hcidata.info/host2ip.cgi
- Bericht wordt opgedeeld in IP-pakketjes die hun weg naar de bestemming zoeken over het net
  - Oud telefoonnetwerk: je had de hele lijn voor je gereserveerd
  - Het zoeken van de weg: TCP-protocol (Transmission Control Protocol)

➡ **TCP/IP-protocol**

Local Area Network (LAN) & Wifi



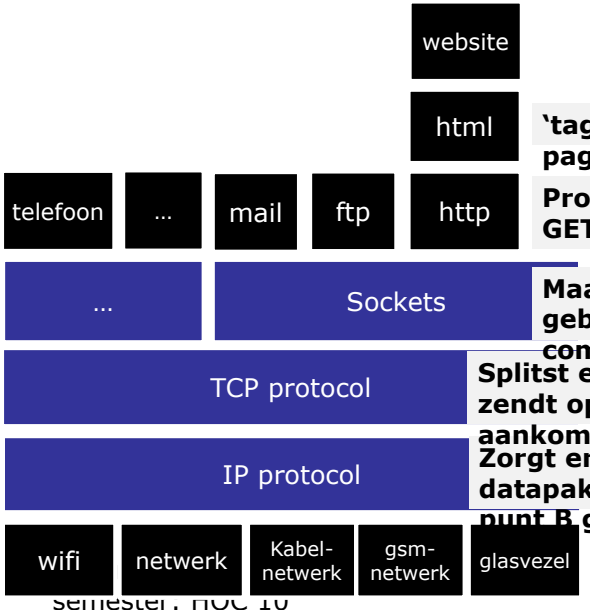
Je hebt een *vast, globaal ip-adres* die van overal bereikt kan worden

Je hebt een *lokaal ip-adres* enkel geldig binnen het LAN. De rest van het internet wordt bereikt via de gateway. De rest van het internet kan jou alleen vinden als de gateway het bericht doorstuurt (dit kan je instellen, bij telenet bvb)

Je krijgt een *tijdelijk ip-adres* elke keer als je de wifi activeert. Verder identiek als een LAN.

Je hebt een *localhost ip-adres* (meestal 127.0.0.1) om met jezelf te communiceren

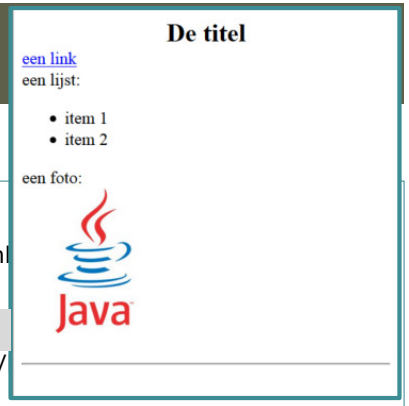
Technologie: softwarecomponenten



semester: HOC 10

Een html-webpagina

Op te vragen via rechtermuisknop -> View Page Source




```
<html>   ⇐ TAG
<head>
<meta http-equiv="content-type" content="text/html
<title>Webpaginavoorbeeld</title>
</head>
<body>   ⇐ BEGIN VAN WAT GETOOND WORDT
<div align="center"><big><big><b>De titel</b></
</div>
<a href="http://parallel.vub.ac.be/education/java/theorie.html">een
link</a><br>
een lijst:<br>
<ul>
<li>item 1</li>
<li>item 2</li>
</ul>
een foto:<br>
<img alt="" src="java.jpg" height="136" width="136"><br>
<br>
<hr size="2" width="100%"><br>
</body>
</html>   ⇐ EINDE VAN TAG
```

FIGUREN STAAN IN APARTE FILE

Browser: HTTP-protocol

Te bekijken met browser-add-on **Live HTTP Headers**

- Website opvragen gebeurt via een GET- of POST – request
  - Je stuurt www-adres
  - POST is voor het meezenden van input
- Antwoord: OK/found met inhoud die getoond wordt of foutmelding (vb page not found)
- De basis is een html-pagina die als file op server staat
  - Figuren staan apart als file op server en worden 1-voor-1 opgestuurd
  - 'Webserver.java' is voorbeeldcode van een eenvoudige webserver die html-files verstuurt naar de browser
- Eenvoudig websites maken: zie link op parallel

Internetconnecties via sockets

Client - server

- **Client** tracht een connectie te maken met een server via zijn ip-adres en 'poort'
- **Server** luistert op een 'poort' naar binnenkomende connecties
  - Een poort is een natuurlijk getal
  - Elke applicatie gebruikt een eigen poort
    - Websites (http): 80
    - File transfer (ftp): 20
    - http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
  - Zo kan de communicatie van de verschillende applicaties uit elkaar gehouden worden.

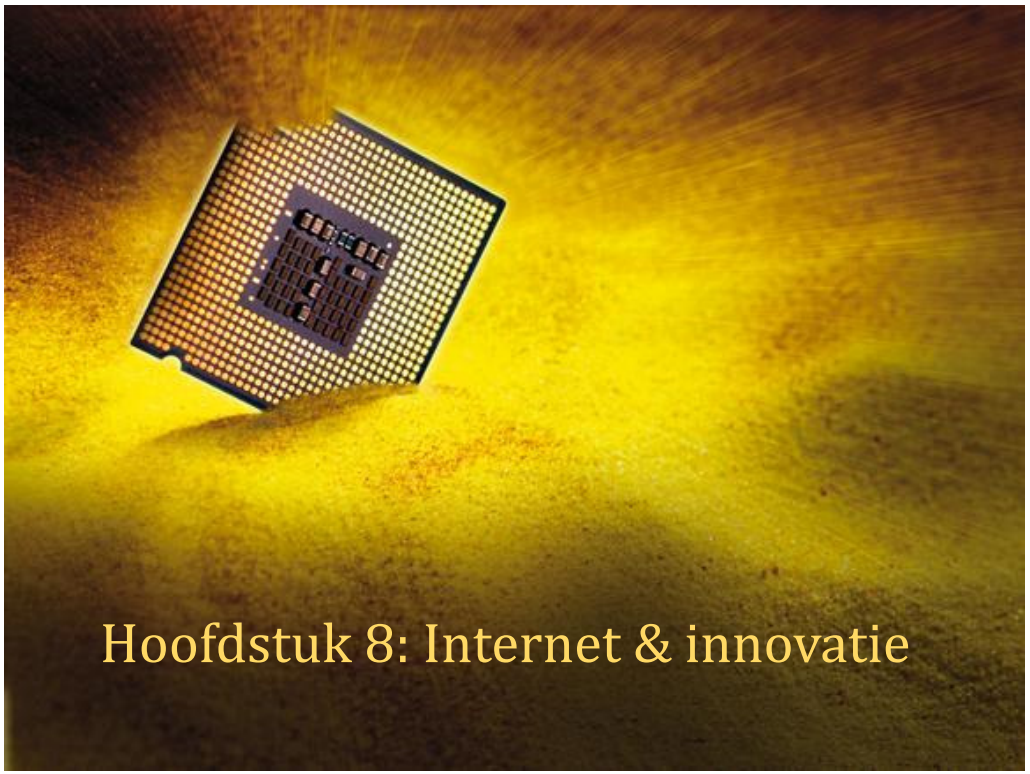
Test het uit

- Zoek het ip-adres op van enkele websites (domeinnaam)
  - http://www.hcidata.info/host2ip.cgi
- Zoek je eigen ip-adres op
  - Windows-commando *ipconfig* (Windows-teken en dan 'cmd' typen) of onder *Performance* -> *Wifi* in de *Task Manager* (via Windows-teken + x)
  - Gebruik het javaprogramma 'MyIpAddress'
- Test de connectie met een ander ip-adres
  - Commando (Windows-teken + cmd): **ping <ip-adres>**
- Maak een connectie via sockets
  - Start een server met Javaprogramma 'MyServer' (poort 6667) of MyPingPongServer (poort 6657 – verander NETWERK_TYPE)
    - Gebruik NETWERK_TYPE = LOCALHOST om op je eigen computer te testen
  - Maak een clientconnectie met de server met 'MyClient':
    - Specifieer ip-adres op lijn 20: `String IP_ADDRESS = "192.168.1.100";`
    - Als server op eigen computer staat: IP_ADDRESS = "127.0.0.1"
  - Maak een connectie met PingPongServer met 'MyClient': zet poort op 6657 en geef ip-adres

MyClient en **MyServer** staan in package **internet**

Extra Testjes

- Zoek het ip-adres op van enkele websites (domeinnaam)
  - http://www.hcidata.info/host2ip.cgi))
- Bekijk de HTTP-berichten in je browser
  - Firefox, Chrome: installeer add-on **Live HTTP Headers**
- Creëer en stuur je eigen http-bericht naar een webserver
  - Javaprogramma 'AccessWebServer': zet ip-adres en kopieer http-request



Film "The Social Network"

(2010) David Fincher

Over het ontstaan van facebook

Na de internetbubbel

Technologie-index van USA: Nasdaq

Na de dotcom-crisis

Investeringen vallen (even) stil

Maar: Internet wint meer en meer terrein

En plots… Google begint aan het internet te verdienen

De trein is vertrokken

Komt de droom toch uit?

webtechnologie

- Internet 1.0
  - Informatie te bekijken via browser
- Internet 2.0
  - Gebruiker interageert en voegt informatie toe
- Internet 3.0
  - Semantiek (betekenis)
- Webservices: informatie wordt ter beschikking gesteld, kan automatisch (door computerprogramma) opgehaald worden ipv. via browser
  - Vb: bustijden, google maps, …

De uitvinder van het internet

- 1989: maakte de eerste webserver (http) en browser (html) samen met de Belg **Robert Cailliau**
- 1999: dacht verder:

  "I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."

Door internet: verhoging efficiëntie

- Sneller en gemakkelijker communiceren
- Informatie overal aanwezig en toegankelijk
  - Vroeger: bibliotheek met beperkte info
- Digitale verwerking
  - Bvb tax-on-web: belastingen online invullen ipv via formulier die dan ingescand moet worden
- Webservices: *programma's kunnen info opvragen & gebruiken*
  - luchtvaartmaatschappijen
  - …

Dat internettechnologie heel wat mogelijk maakt is duidelijk, maar technologie is geen garantie op succes…

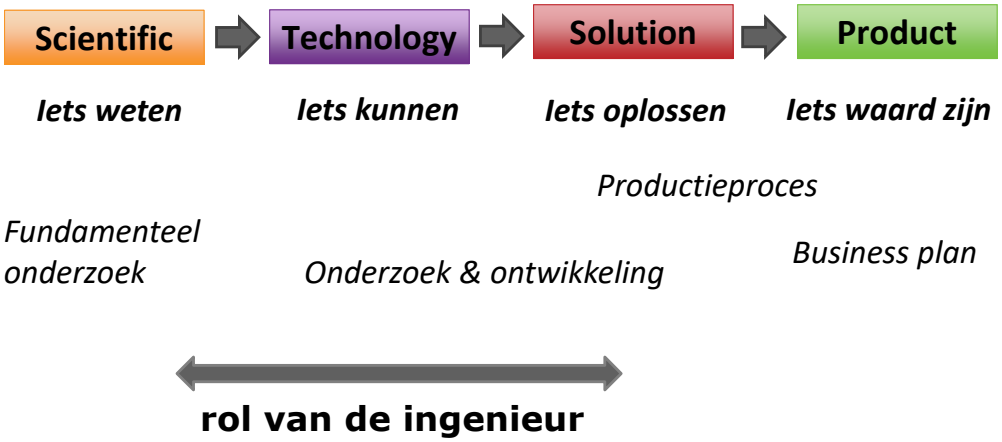
Technologie is niet alles

- Wat is de bijdrage van technologie tot het succes?
- Welke technologie is belangrijk?
- Onverwachte wendingen…
  - Sms werd onbedoeld een enorm succes, mms dan weer niet
- Standaardisatie! Vb: Gsm
  - Eenvormig systeem, van operator veranderen is gemakkelijk
  - USA: alle operatoren hebben een ander systeem
  - Gsm is nu dan ook wereldwijd de standaard (behalve in de USA…)

Wat is er nodig voor succes?

Ingenieurs/techneuten hebben het dikwijls moeilijk om dit te begrijpen omdat ze vooral bezig zijn met de technologie

De lange weg naar de markt



Wat is er, naast technologie, nodig voor succes?

(1) Vertrouwen

- *Vroeger*: securityproblemen
  - Geen vertrouwen
- Gebruik VISA, Online banking,
- Meer en meer online kopen: vertrouwen in kapaza, ebay, …
  - Je betaalt een onbekende en vetrouwt dat hij het opstuurt!
- Ons koopgedrag verandert (*maar niet eensklaps en massaal*)
- Oud voorbeeld: kernenergie
  - Ingenieur vindt dat de consument maar moest vertrouwen dat alles veilig is. Hij kan niet overweg met 'irrationele' argumenten

(2) Gebruiksvriendelijk

- Google maps <> Map24
  - Google via handige functionaliteiten veel gebruiksvriendelijker
  - Ik ben direct overgeschakeld…
- Google verzekerde dat het resultaat steeds snel ter beschikking was (binnen 1 seconde)
  - Voorheen moest je al eens lang wachten
- Apps & muziek
  - Wat is een 'app' anders dan een softwareprogramma?
    - Afgebakend programma door systeem beheerd
  - Itunes store: gemakkelijk muziek kopen

(3) Je moet de grootste zijn

- Of ten minste groot genoeg
- Hoe bereik je kritische massa?
- Vb1: website voor gepersonaliseerde concertaankondigen, afhankelijk van je muzieksmaak
  - Bestaat nog geen succesvolle versie
  - Nog geen 'winner'…
- Vb2: smart TV
  - Enkel succesvol met 1 standaard
  - Vele initiatieven (Apple, Samsung, Google, Microsoft, …), maar geen duidelijke winnaar. Geen enkele speler gunt de ander de machtspositie!

(4) Het menselijke & sociale aspect

- Wat zoeken mensen op internet…
  - 'Contact met andere mensen'
- Facebook's Zuckerberg: studeerde psychologie (naast computer science)
- Begrijpen van 'mens'
  - Apple's Steve Jobs: nadruk op *design*, op *gebruiker* ipv technologie

(5) Goed Business model

Winstgevend zijn op termijn

- Google, facebook:
  - via advertenties
  - Hebben informatie over gebruikers => gerichte reclame
- Betalende sites (bvb krant): moeilijk
  - We verwachten dat alles gratis is…

De grote IT-bedrijven

- De "Big 5", allen Amerikaans
  - Je moet de grootste zijn om de winner te zijn. USA is groot, België is klein.
- Zorgden voor vernieuwing: technologisch, bedrijfsfilosofie en wouden ook spelen op het ***maatschappelijke*** (cf Elon Musk)
- Investeerden veel, hoopten op doorbraken
  - Google ook in robotica, …
  - Maar: "De Standaard 2021 01 25 - Google geraakt maar niet op de maan.pdf"
- In het begin: veel sympathie
- Nu groeiend scepticisme (zeker in USA!)
  - Privacy
  - Te groot, te machtig, houden quasi-monopolies, dringen kleine bedrijven uit de markt, betalen nauwelijks belastingen, …
  - Zie De Standaard van augustus 2018: "Machtspositie van Google-facebook-apple gevaarlijk"

Belangrijkste IT-bedrijven

YAHOO! FINANCE

Vraag: welk aandeel brengt het meeste op?

Key Statistics: | Revenue | Net Income Avl to Common | = winst/omzet | Market Cap | = beurswaarde/winst | = winst/beurswaar

Mei 2020

	Omzet	Winst	Winstmarge	Beurswaarde (koers)	Koers-winst (Rendement)
Microsoft	138,7	46,2	33%	1390 (183)	30 (3,3%)
Apple	268	57,2	21%	1320 (303)	23,1 (4,3%)
Google (Alphabet)	166	34,5	21%	919 (1369)	26,6 (3,7%)
Facebook	73,3	20,9	28,5%	593 (211)	28,3 (3,5%)
Amazon	296	10,6	3,5%	1180 (2367)	111 (0,9%)

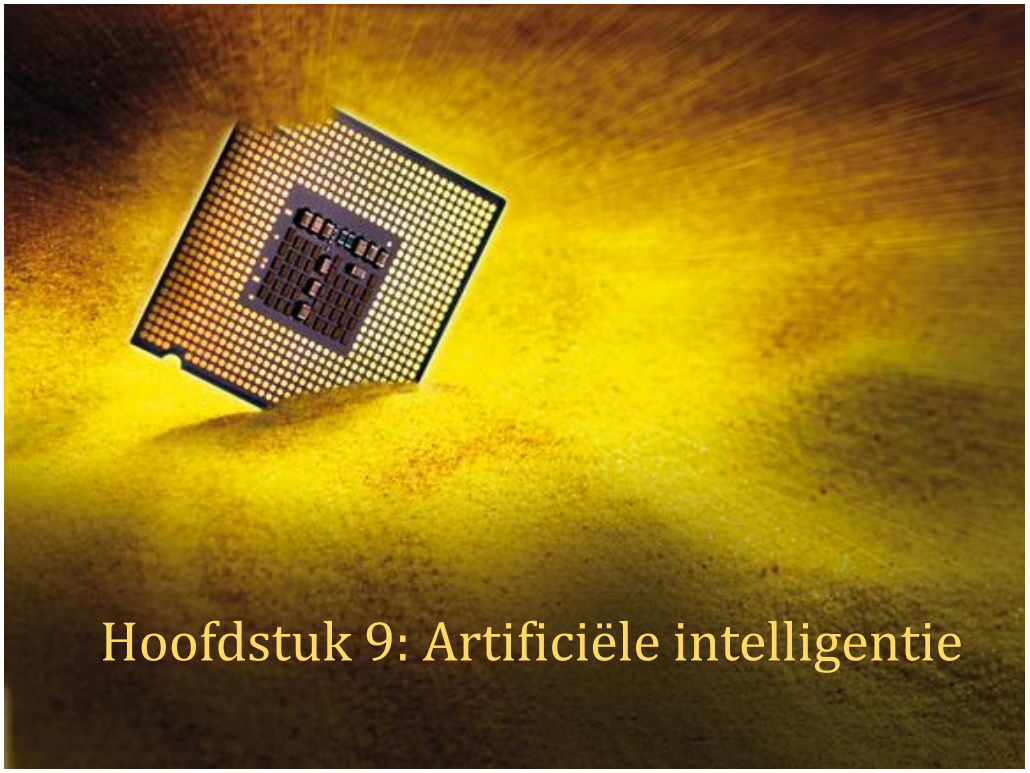
Mei 2022

	Omzet	Winst	Winstmarge	Beurswaarde	Koers-winst
Microsoft	192	72,5	37,6%	1980 (296)	27 (3,6%)
Apple	386	102	26%	2550 (154)	16,3 (6,0%)
Google (Alphabet)	270	74,5	27%	1510 (2287)	20,3 (4,9%)
Facebook	119,0	37,6	31%	563 (197)	14,9 (6,6%)
Amazon	447	21,4	4,7%	1170 (2175)	54,8 (1,8%)

Belangrijke bedrijfsstatistieken

Tussen haakjes de engelse term gebruikt op Yahoo-website onder (Key) Statistics

- **Omzet** (revenue): totaal bedrag van verkochte producten en diensten
- **Winst** (Net Income Avl to Common) = Omzet min de kosten
- **Winstmarge** (%) = Winst / Omzet
- **Beurskoers** = waarde van 1 aandeel
- **Beurswaarde** (Market Capitalization) = beurskoers x aantal aandelen
- **Koers-winstverhouding** = Beurswaarde / Winst
  - Geeft aan hoeveel jaar het duurt voor je je inzet terug hebt verdient
- **Rendement** (%) = 1 / Koers-winstverhouding
  - Winst die je maakt op je aandelen



Film 1968

Computer HAL

IBM

Artificiële Intelligentie, de grote uitdaging

“Is er een machine die even slim of zelfs slimmer is als de mens!?

Zal die er weldra zijn?”

Op examen volgende vragen kunnen beantwoorden, liefst met een eigen onderbouwde visie (gebruik voorbeelden!):

- **Wat is volgens u intelligentie?**
- **Is de huidige computer al intelligent?**
- **Zijn we al goed op weg naar een intelligente computer?**
- **Wat verwacht je in de nabije toekomst?**

Wees redelijk eigenzinnig – je krijgt pluspunten voor originele maar onderbouwde voorbeelden, standpunten, meningen, …

Onderzoeksdomeinen in de AI

- Twee belangrijke types AI-algoritmen
  - **Type A**: Patronen herkennen in gegevens
    - Gebaseerd op leren uit grote hoeveelheden data
    - Maakte recent een grote sprong voorwaarts met de ontdekking van *deep learning*
  - **Type B**: Symbolisch rekenen, gebaseerd op symbolen = tekens die een betekenis (semantiek) hebben
    - Was lang de voornaamste tak van de AI, maar zonder grote doorbraken
- Nog andere:
  - Reinforcement learning (zie hoofdstuk 5 van deel II)
  - Multi-agent systems
  - …

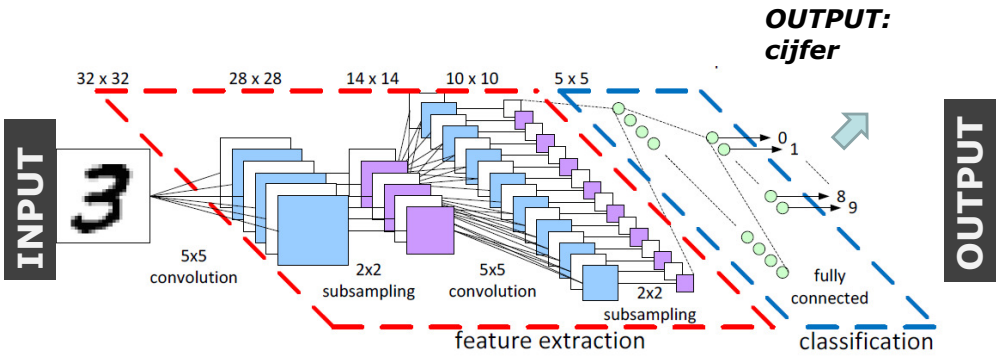
AI type (A) toepassingen

- Stemherkenning
  - Herkennen wat je zegt lukt redelijk momenteel
  - Maar herkennen ***wie*** spreekt, kan computer niet. Of de emoties in je stem herkennen
- Objectherkenning
  - **Voorbeelden**: security, gezichtsherkenning in facebook, verkeersbordherkenning, nummerplaatherkenning bij ingang VUB
  - Echter moeilijk bij slecht weer, donkerte, occlusie, …
- Gebaseerd op het trainen van modellen met voorbeelden
  - **Veel training data nodig**!! Data is nu het nieuwe ‘goud’
  - Door het taggen van vrienden in foto’s op Facebook hebben we Facebook aan die data geholpen!
  - Met je getrouwheidskaart van de supermarkt verzamelen ze data over je koopgedrag
  - Ook Facebook kent onze voorkeuren goed
    - Gebruikt tijdens verkiezingen om politieke keuze te beïnvloeden!

Automatische verkeersbordherkenning in de auto

Type (A): Verkeersbordherkenning

Met een **neuraal netwerk** (gebaseerd op hoe hersens werken)



Zo'n neuraal netwerk is ook de basis van '**deep learning**'. De gewichten van het hele network worden geleerd, ook die van de diepste lagen.

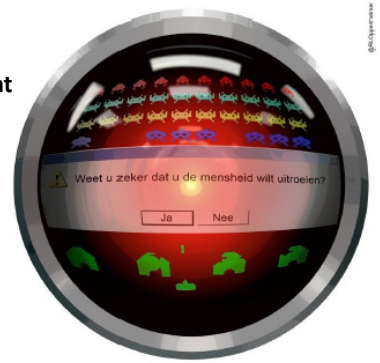
Lobe.ai: https://www.youtube.com/watch?v=IN69suHxS8w

DOORBRAAK IN 'MACHINE LEARNING' BIJ GOOGLE-FILIAAL DEEPMIND

Computer leerde zichzelf Atari 2600-spelletjes

"Een computer van Google DeepMind heeft zichzelf 49 spelletjes op de klassieke Atari 2600-console aangeleerd, en speelt nu even goed of beter dan een mens. Kinderspel? Allerminst. Misschien is het zelfs een doorbraak in artificiële intelligentie."

Gebaseerd op 'deep learning', de grote hype van het moment

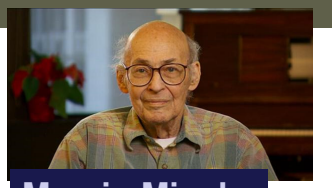


De Standaard – 26 februari 2015

Algemene versus specifieke intelligentie

- **Narrow/weak AI**
  - Kan 1 welbepaalde taak succesvol uitvoeren
  - Het leeralgoritme van Google Deepmind:
    - heeft veel tijd nodig om te leren aan de hand van veel voorbeelden (training data)
    - kan de resultaten niet extrapoleren van één spel naar een ander.
  - Goede resultaten behaald, maar noem ik eerder *slimme* dan *intelligente* algoritmen.
  - Modellen worden incrementeel verbeterd, maar volgens mij zullen ze nooit intelligentie bereiken. En zeker niet:
- **General/strong AI**
  - "= the ability to learn and apply its intelligence to solve *any* problem"
  - Hierin staan we nog nergens…

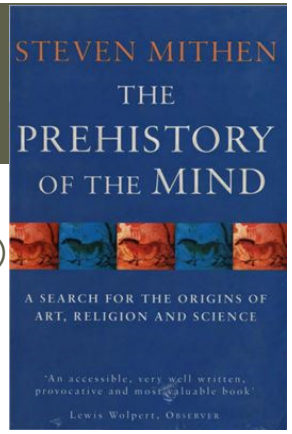
Hoe bekomen we intelligentie?



Marvin Minsky

- Founding father of AI
- Ontdekker van artificiële neurale netwerken
- The Society of Mind theory
  - Intelligentie ontstaat uit het samenvoegen van vele 'domme' building blocks die elk iets kunnen. Het geheel wordt intelligent.
  - Ga je zo van Narrow AI naar General AI?
    - Open vraag
    - *Ik denk het niet, iets algemeens nodig om intelligent met kennis om te gaan*

Steven Mithen



- Mens initieel: modulair denkend (hokjesdenken)
  - Zoals een Zwitsers zakmes: voor elke taak bestaat er een eigen geïsoleerd systeem
- Homo Sapiens wordt gekenmerkt door cognitieve vloeibaarheid ('cognitive fluidity')
  - De barrières tussen de modules zijn weg, hersens zijn minder gecompartimenteerd
- Om van 'Narrow' naar 'General AI' moeten we niet enkel alles samenvoegen, maar ook nood aan:
  - Creativiteit, gebaseerd op analogieën en metaforen

AI: type (B) toepassingen

- Redeneren, logische systemen
  - Gebaseerd op symbolen
  - Wat Leibniz voor ogen had (zie eerste hoofdstuk)
  - Gerelateerd aan taal
    - Begrijpen van onze, natuurlijke taal is ook nog een grote uitdaging. Voor ons geen probleem, voor computers onbegrijpbaar
  - Expertsystemen: neerschrijven kennis met de logica
- Semantische web
  - Ipv louter informatie, moet de webserver ook de *betekenis (= semantiek)* van de informatie begrijpen
    - Voorbeeld: je moet telkens je gegevens invullen omdat op een website je adres zo gespecifieerd is en op een andere website weer anders. De *betekenis* van je gegevens ontgaat de server.
  - Informatie omzetten in kennis
  - Zodat webservers met elkaar kunnen communiceren

Semantische Web

Tim Berners-Lee

- De uitvinder van het internet
- 1989: maakte de eerste webserver (http)

en browser (html) samen met de Belg **Robert Cailliau**

- 1999: dacht verder:

"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."

Type (B) Google translate

- Vertalingen: hoe goed/slecht is Google Translate?
  - Kent hij verschil tussen "The electrician is working" en "The telephone is working"?
  - Intussen wel, maar complexere ambiguïteiten kan hij niet onderscheiden
- Google translate is gebaseerd op type (A) oplossing
  - Leren uit grote hoeveelheden data, namelijk documenten die door mensen vertaald zijn.
  - Pure statistiek en recent ook neurale netwerken.
  - Heeft geen model van grammatica, syntax of semantiek (betekenis)
- Resultaat: het idee van de zin (de essentie/pointe) ontgaat hem

Type (B) IBM's Watson

- IBM's supercomputer Watson
  - Kent de hele Wikipedia van buiten (en andere bronnen) en zocht naar verbanden
  - => won de Amerikaanse TV-kwis *Jeopardy!* tegen de beste Amerikaanse (menselijke) kwissers!
  - Watson was getraind met duizenden antwoorden van *Jeopardy!* Waarbij ook de waarheid was aangegeven
- Wordt nu ingezet in de medische wereld (als diagnosesysteem)
  - Leerde de hele corpus van medische literatuur
  - IBM CEO Rometty zei: "AI is real, it's mainstream, it's here, and it can change almost everything about health care."
- Maar: beperkingen kwamen naar boven
  - Heeft het moeilijk met ambiguïteiten
  - Kan subtiele aanwijzingen niet oppikken, terwijl een arts dat wel doet
  - Begrijpt extra inzichten niet uit nieuwe wetenschappelijke bevindingen
- Watson wel goed als veredeld opzoekinstrument (search engine zoals Google dus)…

Type (B) Algemene, intuïtieve kennis / gezond verstand

- Computers hebben vooral moeite met algemene, intuïtieve kennis (common sense) die voor ons evident is
  - "Mijn moeder is ouder dan mij"
- CYC (begon in 1984): alle algemene kennis opschrijven met logische regels
  - Intussen al 25 miljoen beweringen (lemma's)
  - Nog steeds onvoldoende om al onze intuïtieve kennis te beschrijven!
  - Terwijl wij, mensen, die kennis als 'vanzelf' weten/aanleren
- Old-school expert systemen falen ook…
- "The absence of common sense prevents intelligent systems from understanding their world, behaving reasonably in unforeseen situations, communicating naturally with people, and learning from new experiences. Its absence is considered the most significant barrier between the **narrowly focused AI applications** of today and the more **general, human-like AI systems** hoped for in the future."

Wat is intelligentie?

- **Wikipedia:** abstract thought, understanding, self-awareness, communication, reasoning, learning, having emotional knowledge, retaining, planning, and problem solving.
- Is taal noodzakelijk voor intelligentie?
- Begrijpen lijkt noodzakelijk! Maar wat is *begrijpen*?
  - Zie mijn analyse van menselijk schaken [Deel II hoofdstuk 5]
  - *Maar:* computers tonen aan dat *competentie kan zonder begrip*. Een volmaakte en mooie rekenmachine hoeft niet te weten wat rekenen is!

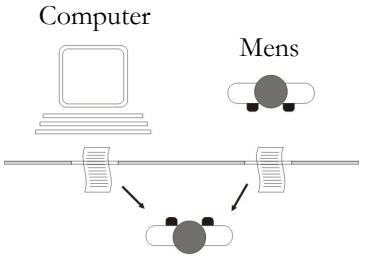
Er is nog geen eenduidige definitie van intelligentie!

- En wat is bewustzijn?
  - Gerenommeerd filosoof Daniel Dennet schreef hierover 'Van Bacterie naar Bach en terug' in 2017. Staat in VUB-Bibliotheek.

Turing Test (1950)

Alan Turing
1912-1954

- 'Can machines think?'
  - Artificiële intelligentie…
- Kunnen we verschil maken tussen mens en computer?



Omdat er geen duidelijke definitie bestaat over wat intelligentie inhoudt en wij als mens prototype zijn van intelligentie, bedacht Turing een test. Door vragen te stellen tracht je te ontdekken of er achter het scherm een computer of mens bevindt. Als we geen verschil meer kunnen maken, kunnen we de computer intelligent noemen.

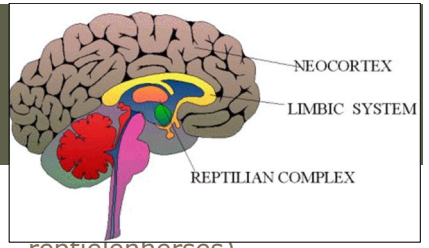
Turing test

- http://www.turinghub.com/
- http://www.loebner.net/Prizef/loebner-prize.html
  - Je kan je intelligent programma opladen
  - 100.000 dollar als je programma slaagt voor de Turing test
- http://cleverbot.com/
  - Communiceer je met een computer of met een mens?

Turing test for bots

- "The idea is to evaluate how we can make game bots, which are Non-Player Characters (NPCs) controlled by AI algorithms, appear as human as possible."
- Unreal Tournament 2004 (results)
  - tournament against one another and about an equal number of humans
  - tag opponents as human or bot.

Piet Vroon
(professor psychologie uGent)



- Drie soorten hersens
  - **Instinct** (Hersenstam/hypothalamus - reptielenherses)
  - **Conditionering** (Limbisch systeem - zoogdierenbrein)
  - **Intelligentie** (Neocortex – meest recente hersens)
- Evolutie van de mens
  - 1 miljoen jaar geleden: evolutionaire sprong (vuur – werktuigen)
  - 50.000 jaar geleden: finale evolutionaire sprong (taal?)
- Geest is federatie van delen
  - Onze daden zijn gevolg van combinatie van 3 hersens

Conclusies voor ingenieur

- Er is gespecialiseerd onderzoek naar A.I. dat geleid heeft tot goede resultaten voor zeer specifieke taken (**Narrow AI!**).
- Maar meestal is engineering nog noodzakelijk
- Meestal zijn simpele, slimme oplossingen even goed
- Blijf kritisch ("redelijk eigenzinnig")
- ➡ Gebruik je eigen intelligentie en ingenioziteit om adequate oplossing te vinden

Helen Greiner (2015): "in de AI worden we om de 5 jaar heen en weer geslingerd tussen het idee dat de technologie op het punt staat de wereld te veroveren, en de totale teleurstelling."