

# **706.088 INFORMATIK 1**

## **GESCHICHTE & BINÄRE OPERATIONEN**

# **WIEDERHOLUNG PROGRAMMIERUNG**

# DEKLARATION VON VARIABLEN

- › Boolesche Werte (True, False)
- › Natürliche Zahlen (integer: 1, 11, 100000)
- › Reelle Zahlen (float: 0.1, 3.7)
- › Text (string: "Kette von Zeichen")
- › Kein Wert (None)
- › Tupel
- › Listen
- › Dictionary

# OPERATOREN

Operatoren definieren die Interaktion  
von Variablen und (deren) Werten

Operator	Beispiel	Ausgabe
+x, -x	-2	-2
+, -	$3 + 4$	7
*, /	$3 * 4$	12

# OPERATOREN

Operator	Beispiel	Ausgabe
//, %	13//3, 13 % 3	4,1
**	3**2	9

# OPERATOREN

Operator	Beispiel	Ausgabe
not	not True	False
and, or	True or False, True and False	True, False
<, <=, >=, >, ==	4 > 5	False

# OPERATOREN

## Zuweisung

Operator	Beispiel	Ausgabe
=	a = 4, b = a	a = 4, b = 4
+=	a += b	a = 8, b = 4
-=	a -= b	a = 4, b = 4
*=	a *= 3	a = 12, b = 4
/=	a /= b	a = 3.0, b = 4
//=	b //= a	a = 3.0, b = 1.0

# KONTROLLSTRUKTUREN

```
if True:  
    print("always here")  
else:  
    print("never here")
```

```
always here
```

# SCHLEIFEN

```
print("forever")
while True:
    print("and ever")
```

```
forever
and ever
and ever
and ever
...
...
```

# SCHLEIFEN

```
list = ["test1", "test2", "test3"]
for element in list:
    print(element)
```

```
test1
test2
test3
```

# FUNKTIONEN

```
def add(x, y):
    return x + y

def print_result(result):
    print("Result: {}".format(result))

def main():
    a = 10
    b = 20
    c = add(a, b)
    print_result(c)

if __name__ == "__main__":
    main()
```

```
Result: 30
```

# **GESCHICHTE DER INFORMATIK**

# **WAS IST INFORMATIK**

**Wissenschaft der " systematischen " oder " automatischen  
Verarbeitung " von " Information "**

**Information + Automatik**

# GESCHICHTE DER INFORMATIK

- › Aus 3 Teilgebieten zusammengesetzt
  - › Mathematik
  - › Mechanik
  - › Elektronik
- › ... oder vom Abakus zum Quantencomputer

# GESCHICHTE DER INFORMATIK

'Logische Maschinen' bereits im 13. Jahrhundert

Wunsch der Automation von mathematischen  
Berechnungen

Erstes **mechanisches** Rechengerät wurde im Jahr 1623 von  
Wilhelm Schickard gebaut

# MATHEMATIK, MECHANIK UND INFORMATIK



Schickard'sche Rechenmaschine (1623 n. Chr)  
By Herbert Klaeren - Transferred from [1], CC BY-SA 3.0, Link

# SCHICKARD'SCHE RECHENMASCHINE

- › 1623 n. Chr
- › Basierend auf Zahnräderen
- › Addition und Subtraktion von bis zu **sechsstelligen** Zahlen
- › Speicherüberlauf
  - » Akustisches Signal
- › Pläne bis 1960 verloren

# BLAISE PASCAL - PASCALINE



Pascaline

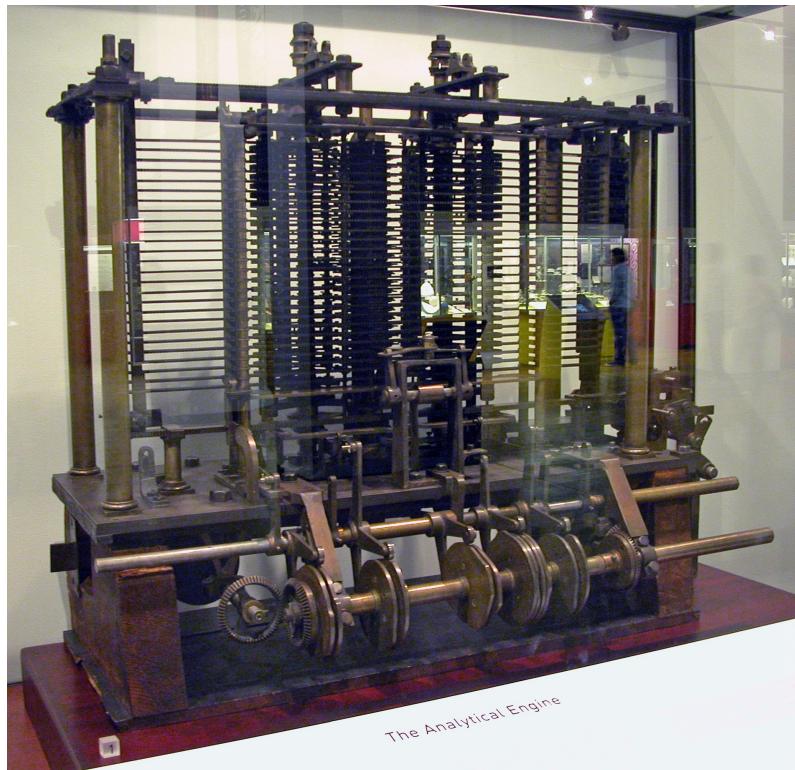
By © 2005 David Monniaux / , CC BY-SA 3.0, [Link](#)

8 . 1

- › Erster (mechanischer) Taschenrechner
- › 50 Prototypen
- › Öffentliche Präsentation 1645
- › Funktionsumfang
  - » Addition
  - » Subtraktion
  - » von 2 Zahlen

# MATHEMATIK, MECHANIK UND INFORMATIK

- › Charles Babbage (1791 - 1871)
  - » Mathematiker, Philosoph, Erfinder und Entwickler.
  - » “Einer der Väter des (mechanischen) Computers”
  - » Bekannt für die Babbage- oder Difference-Engine zur Lösung polinomialer Gleichungen.



Analytical Machine von Babbage  
Von Bruno Barral (ByB), CC BY-SA 2.5, <https://commons.wikimedia.org/w/index.php?curid=6839854>

9 . 2

# MATHEMATIK, MECHANIK UND INFORMATIK

- › Herman Hollerith (1860 - 1929)
  - » Erfand das Lochkartensystem zur Datenspeicherung (1884)
  - » Automatische Auswertung der gespeicherten Daten durch 'Lochkarten-Leser'
  - » Ebnete den Weg zur Speicherung von ersten Computer-Instruktionen.
  - » Verwendung der Hollerith Maschine (Tabulating machine ) zur Volkszählung 1890 in den USA

	1	1	3	0	2	4	10	On	S	A	C	E	a	c	e	g		EB	SB	Ch	Sy	U	Sh	Hk	Br	Rm
2	2	2	4	1	3	E	15	Off	IS	B	D	F	b	d	f	h		SY	X	Fp	Cn	R	X	Al	Cg	Kg
3	0	0	0	0	0	W	20		0	0	0	0	0	0	0	0	.	0	0	0	0	0	0	0	0	0
A	1	1	1	1	0	25		A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	2	2	2	2	5	30		B	2	2	.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C	3	3	3	3	0	3		C	3	3	3	.	3	3	3	3	3	3	3	3	3	3	3	3	3	3
D	4	4	4	4	1	4		D	4	4	4	4	.	4	4	4	4	4	4	4	4	4	4	4	4	4
E	5	5	5	5	2	C		E	5	5	5	5	5	.	5	5	5	5	5	5	5	5	5	5	5	5
F	6	6	6	6	A	D		F	6	6	6	6	6	.	6	6	6	6	6	6	6	6	6	6	6	6
G	7	7	7	7	B	E		G	7	7	7	7	7	.	7	7	7	7	7	7	7	7	7	7	7	7
H	8	8	8	8	a	F		H	8	8	8	8	8	.	8	8	8	8	8	8	8	8	8	8	8	8
I	9	9	9	9	b	c		I	9	9	9	9	9	.	9	9	9	9	9	9	9	9	9	9	9	9

Lochkarte

By Unknown - Library of Congress <http://memory.loc.gov/mss/mcc/023/0008.jpg>, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=30538485>

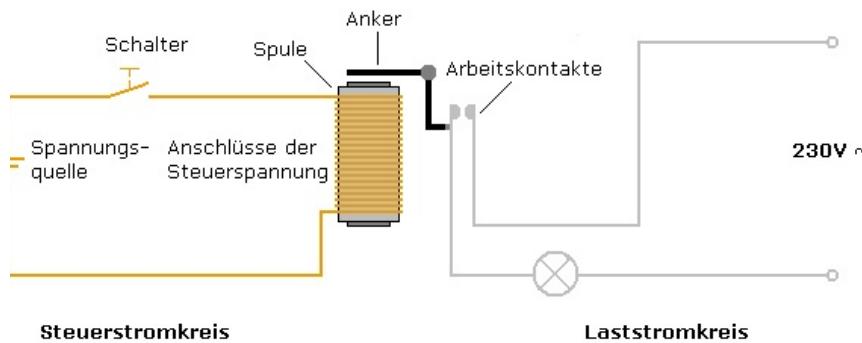


11.1

# RELAIS

- › Mechanischer Schalter
  - » Probleme:
    - › Umschalten dauert einige Sekundenbruchteile
    - › Benötigt viel Platz
    - › Taktfrequenz sehr beschränkt
    - › Mechanische Abnutzung

# RELAIS



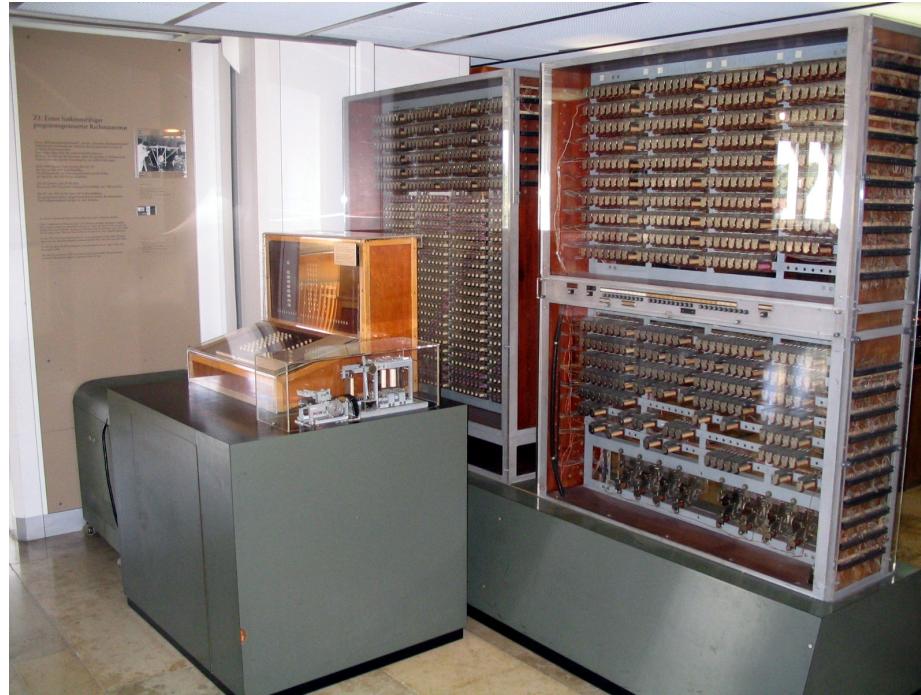
Spule wird unter Strom gesetzt, Magnetfeld zieht Anker, Arbeitskontakte werden geschlossen, Strom kann fließen

| Stefan Riepl in der Wikipedia auf Deutsch - Eigenes Werk, CC BY-SA 2.0 de,  
<https://commons.wikimedia.org/w/index.php?curid=10663175>

# KONRAD ZUSE (1910 - 1995)

Entwickelte den ersten **Digital**-rechner weltweit,  
die Z3 (1941)

- › 600 Relais für den Prozessor
- › 1600 Relais für den Speicher
- › Basierte auf **binärer Gleitkommaarithmetik**
- › So groß wie ein (sehr großer) Kleiderschrank
- › 15-20 Arithmetische Operationen / Sekunde Weitere Informationen zu Konrad Zuse: <http://zuse.zib.de>



Z3 im Deutschen Museum

Von Venusianer aus der deutschsprachigen Wikipedia, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=3632073>

# FIRST BUG

- › Bezeichnung 'Bug' für Fehler in Programmabläufen geht zurück auf Computer Pionierin **Grace Hopper**
- › Am 9. September 1947 dokumentierte sie eine Motte in einem Relais des Computers **Mark II Aiken Relay Calculator** im Log-Buch mit den Worten: "**First actual case of bug being found**"

11.6

9/9

0800 Autam started  
 1000 " stopped - autam ✓ { 1.2700 9.037 847 025  
 13' 0c (032) MP - MC ~~1.2700000000000000~~ 9.037 846 995 const  
 (033) PRO 2 2.130476415  
 const 2.130676415  
 Relays 6-2 m 033 failed special speed test  
 in relay " 10.0m test.  
 Relays changed  
 1100 Started Cosine Tape (Sine check)  
 1525 Started Multi Adder Test.  
 1545 Relay #70 Panel F  
 (Moth) in relay.  
  
 First actual case of bug being found.  
 1625/00 Autam started.  
 1700 closed down.

Relay 2145  
 Relay 3370

By Courtesy of the Naval Surface Warfare Center, Dahlgren, VA., 1988. - U.S. Naval Historical Center Online Library  
 Photograph NH 96566-KN, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=165211>



Grace Hopper, By James S. Davis - Public Domain, <https://commons.wikimedia.org/w/index.php?curid=12421475>  
Grace Hopper "Queen of Software" on Letterman 

# ELEKTRONENRÖHRE

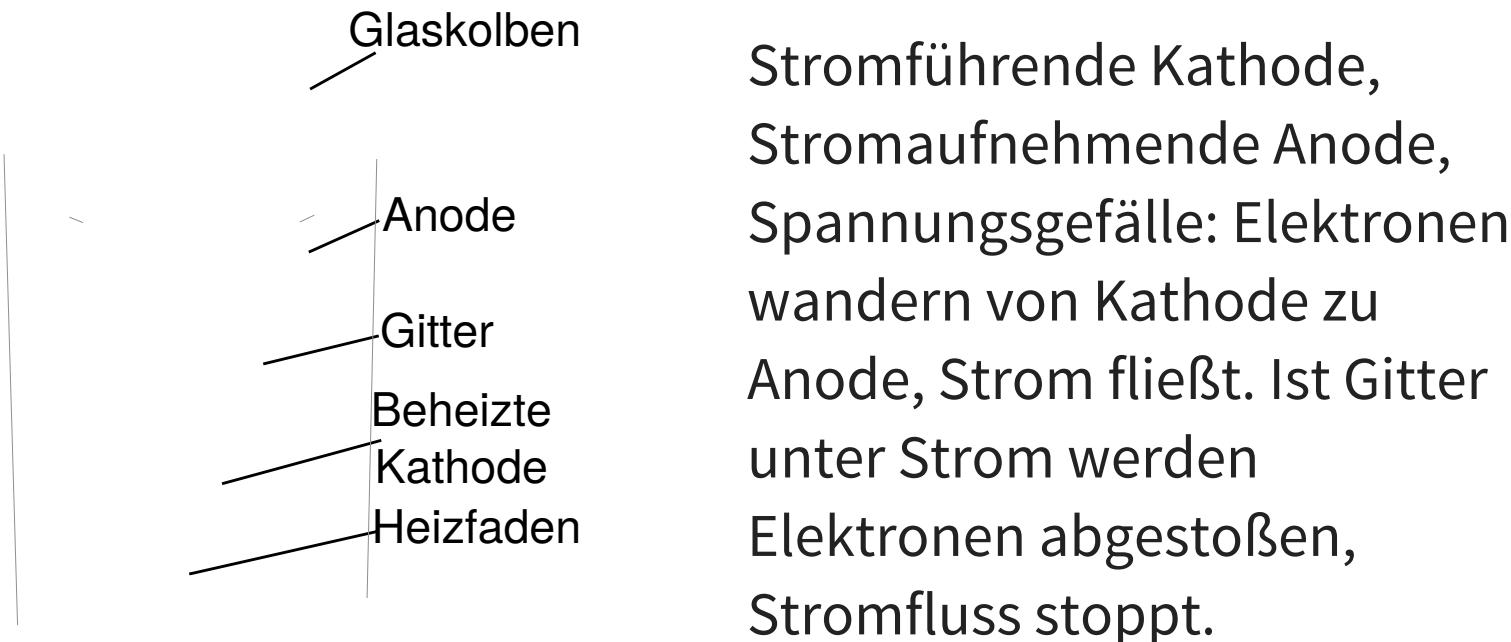


12 . 1

# ELEKTRONENRÖHRE

- › ist auch Schalter
- › 1000 mal schneller als Relais
- › Probleme:
  - » benötigt viel Strom
  - » Lebensdauer gering
  - » Programmierer für ENIAC waren eher Mechaniker

# ELEKTRONENRÖHRE

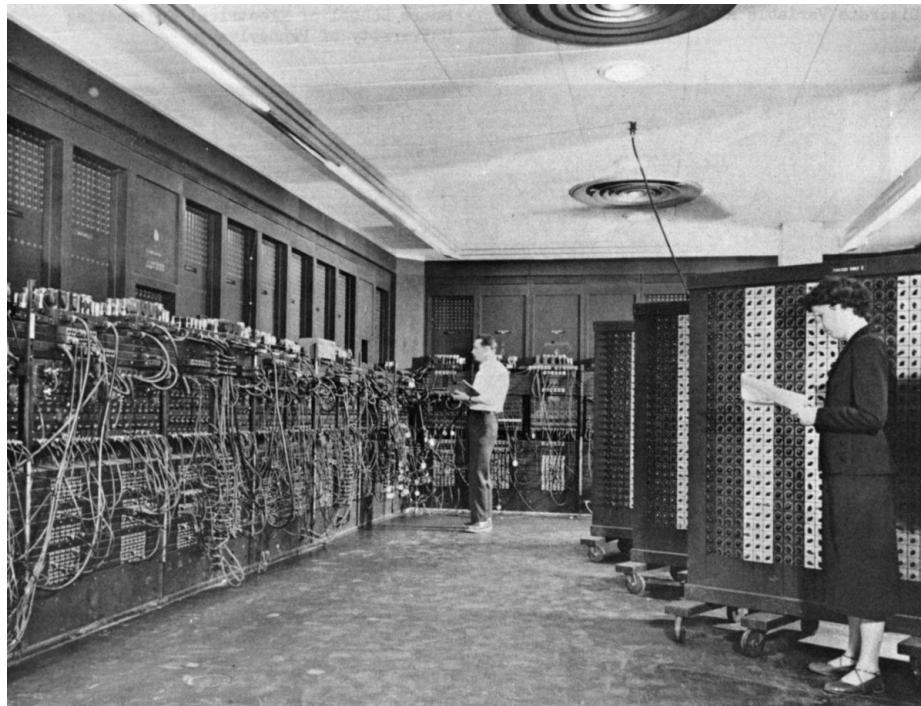


Von [Svjo](#); German translation: Wdwd - [File:Triode-english-text.svg](#), CC BY-SA 3.0,  
<https://commons.wikimedia.org/w/index.php?curid=50578519>

# ENIAC

Erster 'rein elektronischer' Rechner

- › 1946 vorgestellt
- › Elektronenröhren (Vacuum Tubes) zur elektrischen Speicherung von Zahlen
- › 17.468 Elektronenröhren
- › 167 m<sup>2</sup>
- › 27 Tonnen
- › 170 kW
- › USD 468.000 ~ heute USD 6.8 Mio



Eniac

Von Unbekannt - U.S. Army Photo, Gemeinfrei, <https://commons.wikimedia.org/w/index.php?curid=55124>

# ANGLEBIGKEIT VON ELEKTRONENRÖHREN



By Stefan Riepl (Quark48) - Self-photographed, CC BY-SA 2.0 de, [Link](#)

Ca. alle zwei Tage kam es zu einem Ausfall einer  
Elektronenröhre

15 Minuten bis fehlerhafte Elektronenröhre gefunden wurde

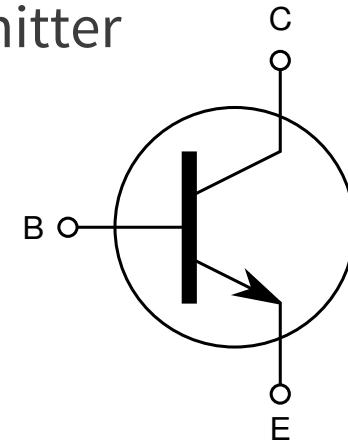
# TRANSISTOR

Shockley, Bardeen und Brattain (Bell Labs)  
entdecken 1947 den ersten Transistoreffekt  
und stellen den ersten **Transistor** vor.

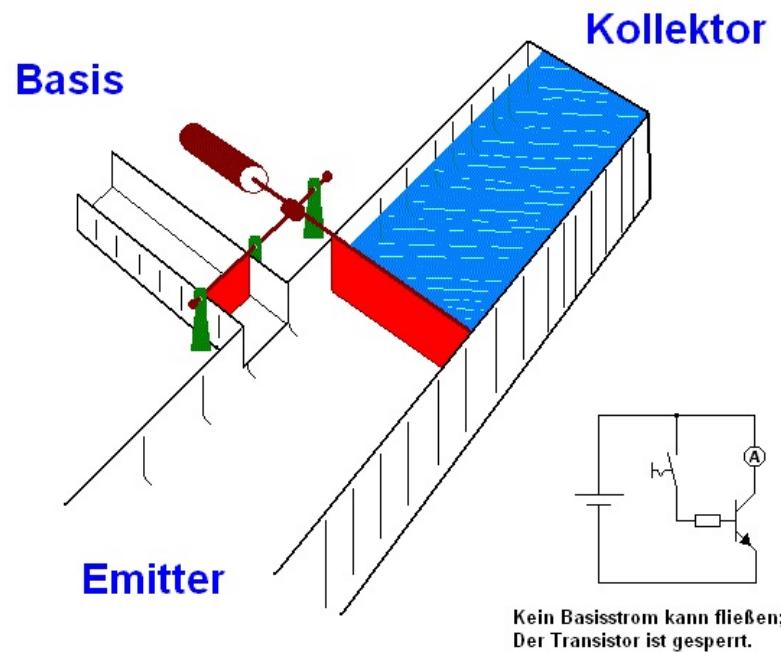
› 1956 Nobelpreis für Physik

# TRANSISTOR

- › Kleiner Strom zwischen Basis und Emitter schaltet großen Strom zwischen Kollektor und Emitter
- › Basis ist im Sperrbetrieb
  - » kein Strom fließt
  - » Kollektor wartet auf Strom
- › Wenn Spannung an Basis anlegt schaltet der Transistor
  - » Strom fließt zwischen Collector und Emitter
  - » Transistor leitet



# TRANSISTOR



Von Stefan Riepl (Quark48 21:02, 2. Dez. 2007 (CET)) - Eigenes Werk (Originaltext: selbst erstellt), CC BY-SA 2.0 de, Link

## Rechenleistung bis ca. 1960

- › 1-2 KByte Speicher
- › 0.02 MIPS

## Rechenleistung 1986 bis heute

- › 8 MByte - ~ 32GByte
- › > 100 MIPS

# MAILÜFTERL

- › 1955 an der TU Wien von Heinz Zemanek mit Studierenden gebaut.
- › erster Volltransistor Rechner am Europäischen Festland
- › 3.000 Transistoren, 5.000 Dioden, 20 km Draht, 132 kHz
- › Transistorspende von Philips
- › erste Berechnung 1958

# MAILÜFTERL



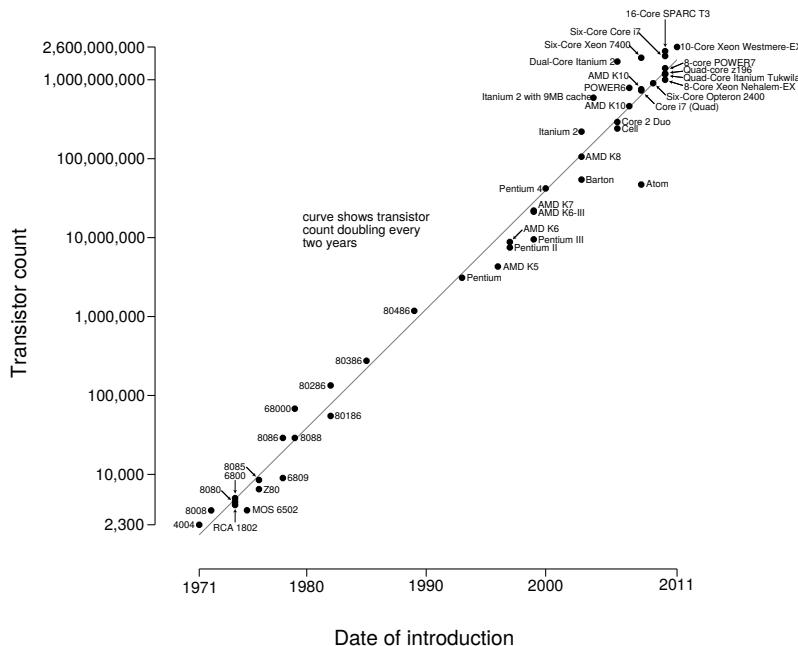
 go there

# MOORE'S LAW

- › Anzahl der Transistoren verdoppelt sich in regelmäßigen Abständen (alle 2 Jahre). → Leistung in **MIPS** verdoppelt sich auch.

# MOORE'S LAW

Microprocessor Transistor Counts 1971-2011 & Moore's Law



By Wgsimon - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=15193542>



# ZAHLENSYSTEME

14 . 1

# DEZIMAL

- › Dezimal-system: Basis 10 (std. Integer)

Ziffern	Präfix
0-9	"" (keiner)

```
>>> i = 123
>>> i
123
>>> f = int("99")
>>> f
99
>>>
```

# HEX

› Hexadezimal-system: Basis 16

Ziffern Präfix

---

0-9, A-F "0x"

```
>>> h = 0xA  
>>> h  
10  
>>> h2 = 0xFF  
>>> h2  
255
```

# OKTAL

› Oktal-system: Basis 8

Ziffern      Präfix

---

0-7      "0o" (Null-O)

```
>>> o = 0o7
>>> o
7
>>> o2 = 0o144
>>> o2
100
```

# BINÄR

- › Binär-system oder Dual-system: Basis 2

Ziffern Präfix

---

0,1      "0b"

```
>>> b = 0b101
>>> b
5
>>> b2 = 0b111111
>>> b2
63
```

# BINÄRSYSTEM

- › Basis für Computer
- › Reduktion auf 2 Zustände
  - » 0: kein Strom, Spannung
  - » 1: Strom, Spannung

# BINÄRSYSTEM

- › Einzelne Stelle heisst: Bit (BInary digit)
- › rechteste Stelle: Least Significant Bit (LSB)
- › linkeste Stelle: Most Significant Bit (MSB)

# BINÄRSYSTEM

```
# Dezimal  
i = 1*10**3 + 8*10**2 + 1*10**1 + 1*10**0 # 1811  
  
# Binär  
b = 1*2**3 + 1*2**2 + 0*2**1 + 1*2**0 # 13
```

# BINÄRSYSTEM UMRECHNUNG (N=2)

- ›  $x(10) \rightarrow x(n)$ :
  - »  $x/n \Rightarrow y$ , Rest  $z$
  - »  $z$  an Stelle 0
  - »  $y \Rightarrow x$  wenn  $y \neq 0$
  - » von Vorne für Stelle 1, 2, ...
  - » wenn  $y = 0$  fertig.

# BINÄRSYSTEM UMRECHNUNG

$25(10) \rightarrow$  binär:

- ›  $25/2 \Rightarrow 12$ , Rest 1
- › 1 an Stelle 0 (LSB)
- ›  $12/2 \Rightarrow 6$ , Rest 0
- › 0 an Stelle 1
- ›  $6/2 \Rightarrow 3$ , Rest 0
- › 0 an Stelle 2
- › ...

- ›  $3/2 \Rightarrow 1$ , Rest 1
- › 1 an Stelle 3
- ›  $1/2 \Rightarrow 0$ , Rest 1
- › 1 an Stelle 4
- › fertig: 11001

# RECHNEN IM BINÄRSYSTEM

Grundregeln für Addition:

$$\begin{array}{r} 0 + 0 \\ \hline 0 + 1 \\ \hline 1 + 0 \\ \hline 1 + 1 \\ \hline 1 + 1 + 1 \end{array} \quad \begin{array}{l} = 0 \\ = 1 \\ = 1 \\ = 0 \text{ (1 Übertrag)} \\ = 1 \text{ (1 Übertrag)} \end{array}$$

# BINÄRE ADDITION

$$\begin{array}{r} + \\ \hline 0101101 & = 45 \\ \hline 0110110 & = 54 \\ \hline 1111 & = \text{Übertrag} \\ \hline 1100011 & = 99 \end{array}$$

# BINÄRE SUBTRAKTION

Computer führt Subtraktion auf Addition zurück: möglich durch Zweier-Komplementbildung

- › Darstellung von negativen Zahlen:
  - » MSB trägt Information über Vorzeichen
- › Limit an darstellbaren Zahlen:
  - » s: verfügbare Bits
  - » Kleinste darstellbare Zahl:  $-2^{** (s-1)}$
  - » Größte darstellbare Zahl:  $2^{** (s-1)} - 1$

# POSITIVE UND NEGATIVE BINÄRZAHLEN

bin	dec	bin	dec
0000	0	1000	-8
0001	1	1001	-7
0010	2	1010	-6
0011	3	1011	-5
0100	4	1100	-4
0101	5	1101	-3
0110	6	1110	-2
0111	7	1111	-1

# ZWEIER-KOMPLEMENTBILDUNG

- › ist MSB 1, ist die Zahl negativ
- › Schritt 1: alle Bits invertieren
- › Schritt 2: zum Schluss 1 addieren

$5(10) \rightarrow -5$	$0101(2)$	$-5(10) \rightarrow 5$	$1011(2)$
Schritt 1: invertieren	1010		0100
Schritt 2: +1	1011		0101

# SUBTRAKTION

Entspricht einer Addition mit dem Zweier-Komplement

› Was geschieht mit Überläufen?

» werden verworfen

$$\begin{array}{r} 6(10) - 2(10) \\ \hline 6(10) & 0110 \\ \hline -2(10) & 1110 \\ \hline 1\ 0100 & = 4 \end{array}$$

# MULTIPLIKATION

kann manchmal durch Verschieben der Bits nach links  
durchgeführt werden

- › Multiplikation mit  $2^{**n}$ : (Shiften um n Bits)

$3 * 2$	$20 * 8$		
3	011	20	010100
2 ( $2^{**1}$ )	10	8 ( $2^{**3}$ )	001000
6	110	160	010100000

# AUSNAHMEN MULTIPLIKATION

Bei Multiplikation mit Zahlen ungleich  $2^{**n}$

```
13(10) * 5(10)
1101   * 101

 1101
+ 0000
+ 1101
1000001 = 65
```

# DIVISION

kann manchmal durch Verschieben der Bits nach rechts  
durchgeführt werden

- › Division mit  $2^{**n}$ : (Shiften um n Bits)

$$\begin{array}{r} 4//2 = 2 \\ \hline 4 & 100 & 24 & 11000 \\ \hline 2 (2^{**1}) & 10 & 8 (2^{**3}) & 01000 \\ \hline 2 & 10 & 3 & 11 \end{array}$$

# AUSNAHMEN DIVISION

Bei Division mit Zahlen ungleich  $2^{**}n$

```
25(10) // 5(10)
 11001 // 101 = 101
 110
  -101 >-----/ ||
 -----
 110
 +1010 Komplement |
 + 1
 -----
 10001
 10 >-----/ |
 101
 -101 >-----/ |
 -----
 101
 +1010 Komplement |
 + 1
 -----
 10000 ---> kein Rest
```

# LIMITIERUNG DER DARSTELLBAREN ZAHLEN

- › Gängige Computer haben 32-bit oder 64-bit Architektur:
  - » 32-bit: max positive Zahl  $2^{32} - 1$
  - » 64-bit: max positive Zahl  $2^{64} - 1$
  - » Python kann in Version 3 gut mit langen Zahlen umgehen.
  - » Grundproblem bleibt generell für Computer bestehen.

```
>>> i = 0b1111111111111111111111111111111111111111111111111111111111111111  
>>> i  
4294967295  
>>> i.bit_length()  
32  
>>> j = i+1  
>>> j.bit_length()  
33  
>>> bin(j)  
'0b10000000000000000000000000000000'  
>>> i64 = 0b1111111111111111111111111111111111111111111111111111111111111111  
>>> i64.bit_length()  
64  
>>> i64  
18446744073709551615
```

# UMWANDLUNG VON 64-BIT IN 16-BIT

› Ariane 5 Explosion  

# **BITOPERATOREN**

# BITOPERATOREN

Operatoren für Binärdarstellung

Operator	Zuweisung	Ergebnis
$\sim x$		bitweises Komplement (Einerkomplement) ( $= -x-1$ )
$\sim x+1$		Zweierkomplement ( $= -x$ )
$x \& y$	$x \&= y$	bitweises UND (AND)
$x   y$	$x  = y$	bitweises ODER (OR)
$x ^ y$	$x ^= y$	bitweises ausschließendes ODER (XOR)
$x << n$	$x <= n$	shiften von x um n Bit nach links
$x >> n$	$x >= n$	shiften von x um n Bit nach rechts

# BITOPERATOREN

```
>>> a = 0b1001
>>> b = 0b0110
>>> bin(a | b)
'0b1111'
>>> bin(a & b)
'0b0'
>>> bin(a ^ b)
'0b1111'
>>> bin(~a)
'-0b1010'
>>> bin(~a & b)
'0b110'
```

# BITOPERATOREN

```
>>> a = 0b1001
>>> b = 0b0110
>>> b >= 1
>>> b
3
>>> bin(b)
'0b11'
>>> bin(a ^ b)
'0b1010'
>>> a << 2
36
>>> bin(a << 2)
'0b100100'
```

# **BITOPERATOREN IN DER PRAXIS**

## **DEMO**

# FRAGEN?

16 . 6