

# Vorkurs in Informatik

Eine Einführung ins Informatikstudium an  
der Universität Zürich

Emanuel Giger, Giacomo Ghezzi, Michael Würsch, and  
Harald Gall

University of Zurich, Switzerland



University of Zurich  
Department of Informatics



# Zielsetzung

Sanfter Einstieg ins  
Studium der Informatik.

Kleinsten gemeinsamen  
Nenner schaffen.

*Socializing.*



# Ablauf

1. Tag: Grundlagen
2. Tag: Software Engineering by Example
3. Tag: Einführung in die Programmierung

# Ablauf: Tag 1

## 09:30 bis 12:00

- Was ist ein Computer?
- Wie ist ein Computer aufgebaut?
- Das Rechnen mit Wahrheitswerten
- Zahlensysteme
- Wie bringe ich den Computer dazu, für mich Probleme zu lösen?

## 13:00 bis 16:00

- Eine Einführung in die Programmierung mit Scratch

# Ablauf: Tag 2

## 09:30 bis 12:00

- Eine Einführung in das systematische Entwickeln von Software (aka. Software Engineering)
- Beginn Gruppenarbeiten: Ein kleines eigenes Projekt mit Scratch

## 13:00 bis 16:00

- Fortsetzung vom Morgen

# Ablauf: Tag 3

## 09:30 bis 12:00

- Kurzpräsentationen der Gruppenarbeiten vom Vortag
- Eine Einführung in die Programmierung mit Groovy

## 13:00 bis 16:00

- Fortsetzung vom Morgen

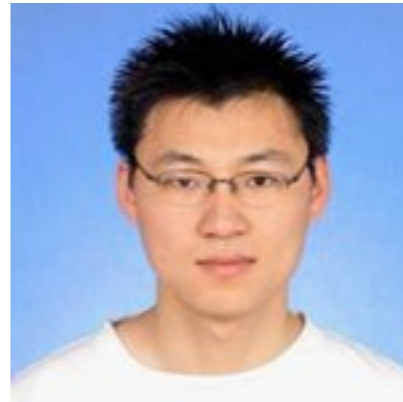
# Das Institut für Informatik (ifi)



Institut für Computerlinguistik



# Wer sind wir?

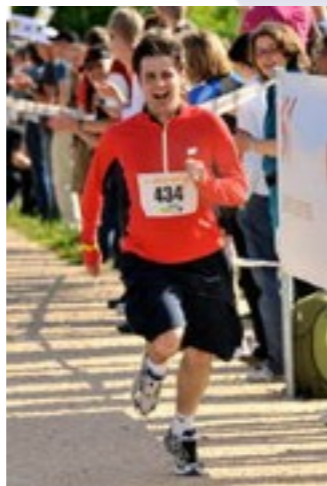


s.e.a.l.

software evolution & architecture lab



software evolution & architecture lab





# Was sind unsere Forschungsschwerpunkte?

- Software Evolution
- Software Wartung und Reengineering
- Software Architekturen
- Produkt-/Programmfamilien
- Verteilte Software Engineering Prozesse
- Methodologien und Paradigmen für Software Entwicklung
- Semantic Web Engineering & Recommender Systems

# Programmieren in der Assessment-/Bachelorstufe

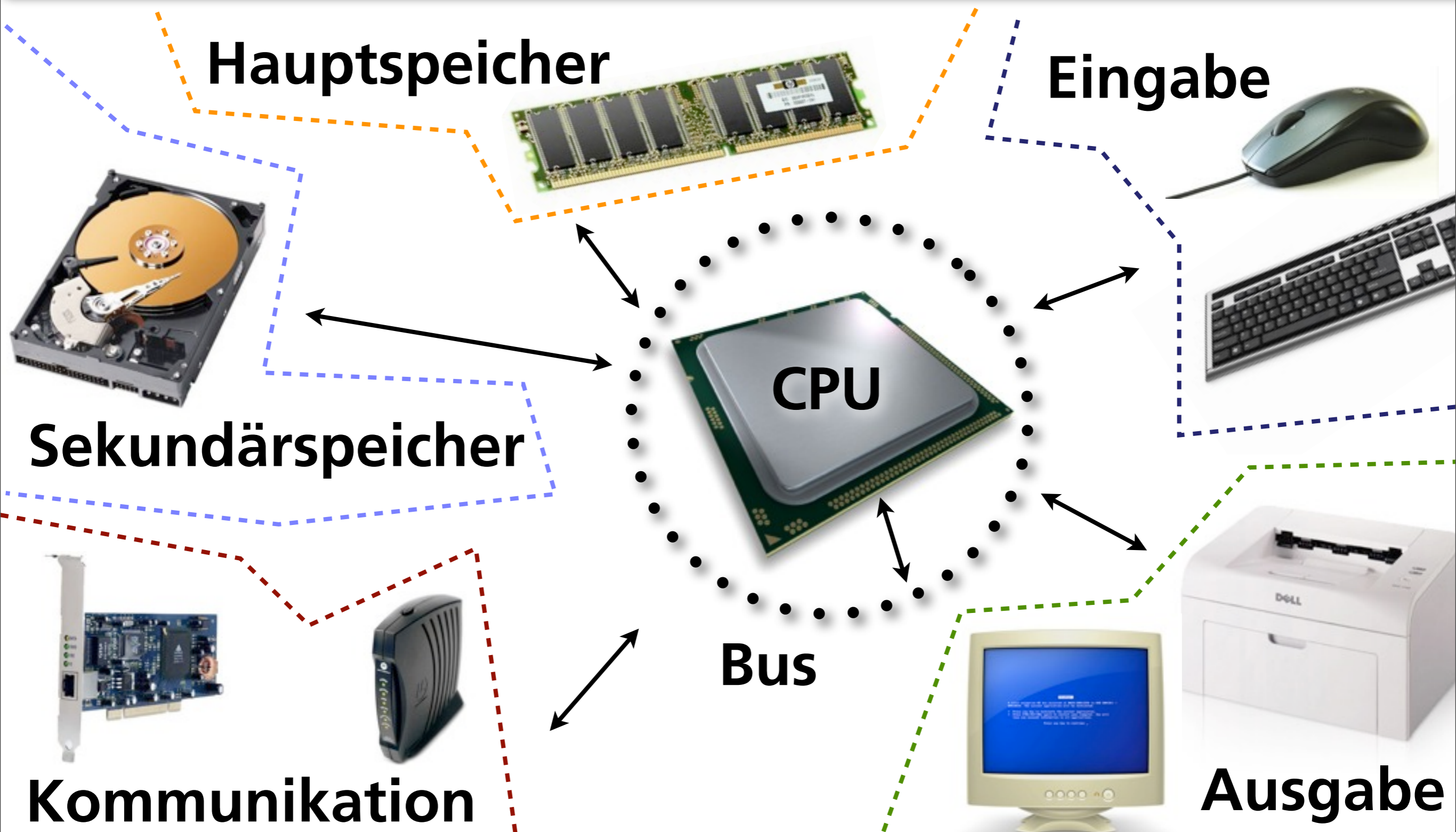
1. Semester: Einführung in die Programmierung  
(Informatik I, Java)
2. Semester: Algorithmen und Datenstrukturen  
(C++)
3. Semester: ??
4. Semester: Software Praktikum  
(aka. SoPra, Java)



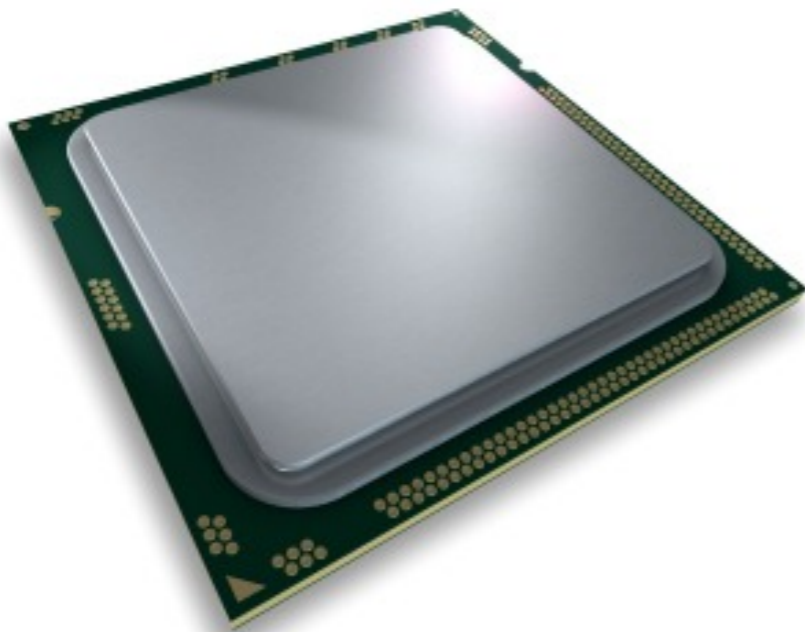
# Was ist ein Computer?



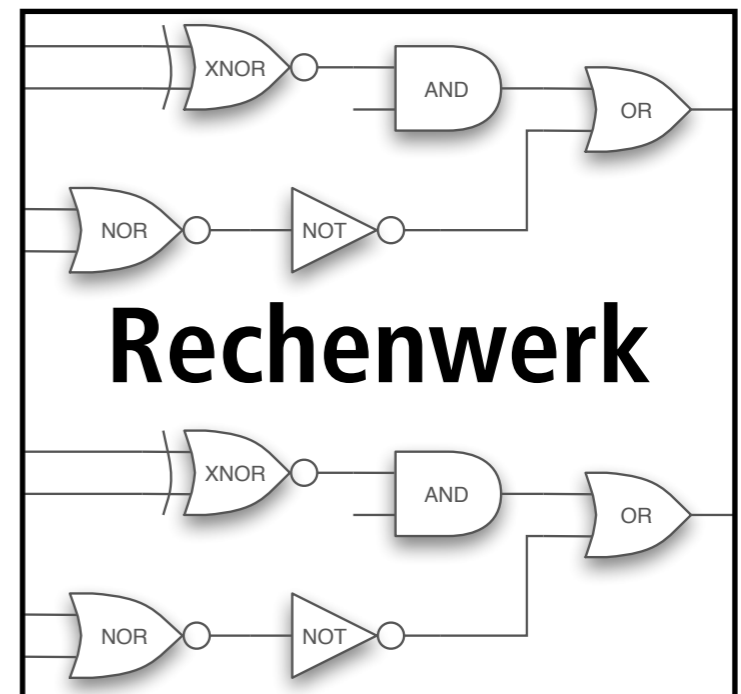
# Wie ist ein Computer aufgebaut?



# (C)entral (P)rocessing (U)nit



=





# Bool'sche Algebra

wahr oder falsch?

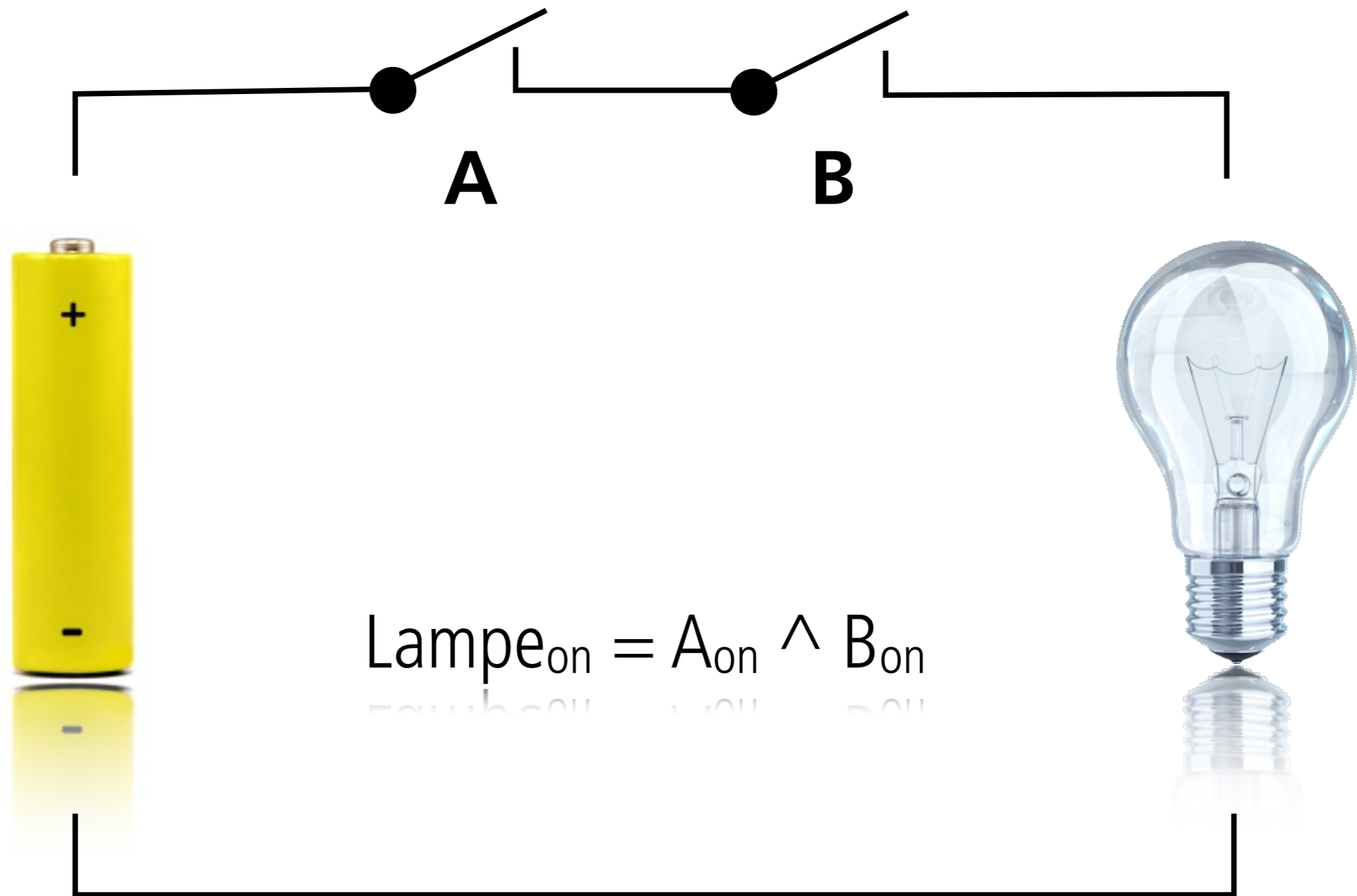
*Bsp: Eine Bestellung kann nur in Auftrag gegeben werden, wenn der Kunde bereits ein Kundenkonto besitzt und eine gültige Kreditkarte angegeben hat.*

$\text{Bestellung}_{\text{ok}} = \text{Konto}_{\text{Existiert}} \text{ und } \text{Kreditkarte}_{\text{gültig}}$





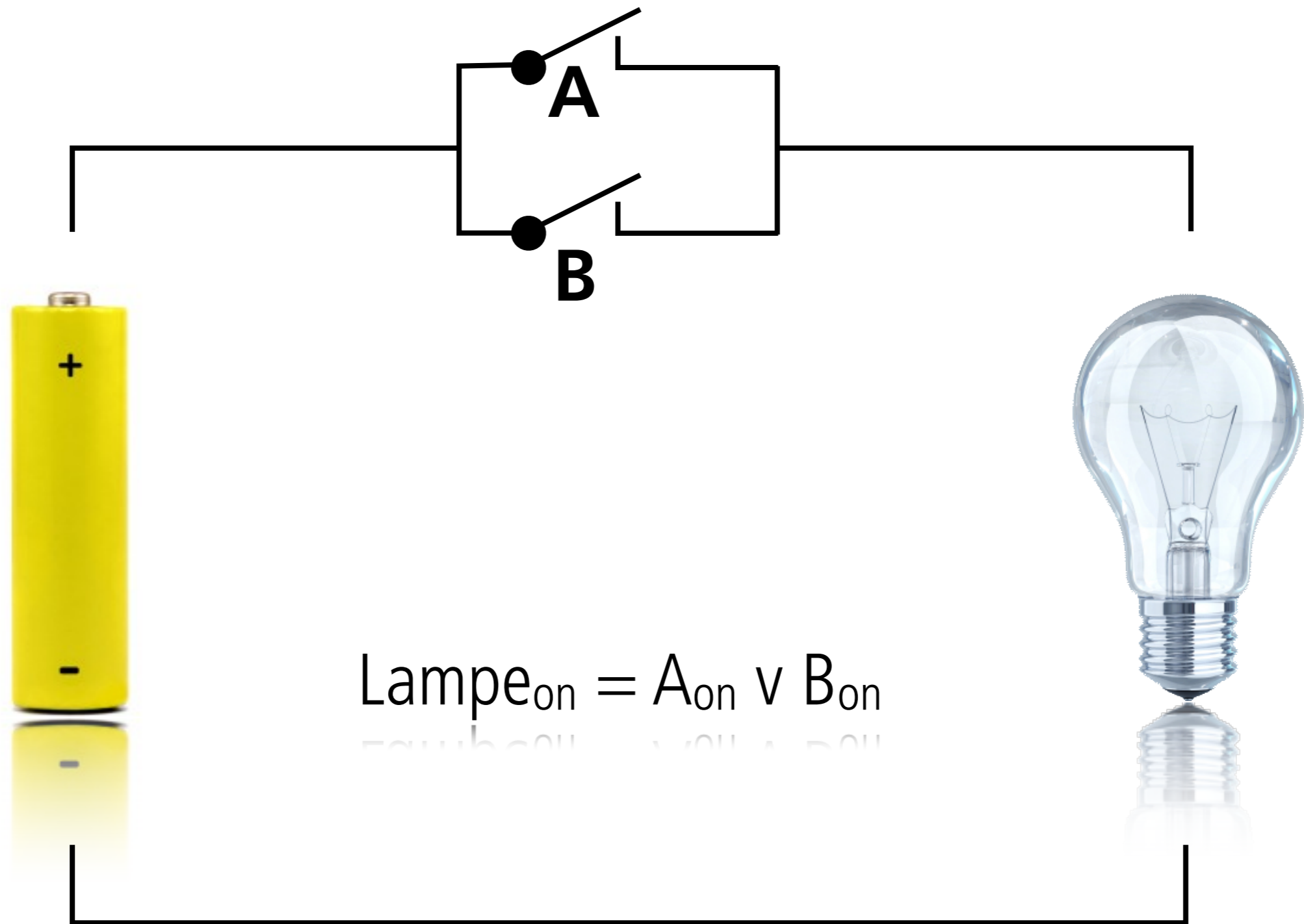
# Bool'sche Algebra: AND



# Wahrheitstabelle: AND

<b>AND</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>F</b>

# Bool'sche Algebra: OR



# Wahrheitstabelle: OR

<b>OR</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>F</b>

# Aufgabe: Bool'sche Ausdrücke

Wann sind die folgenden  
Ausdrücke **wahr**?

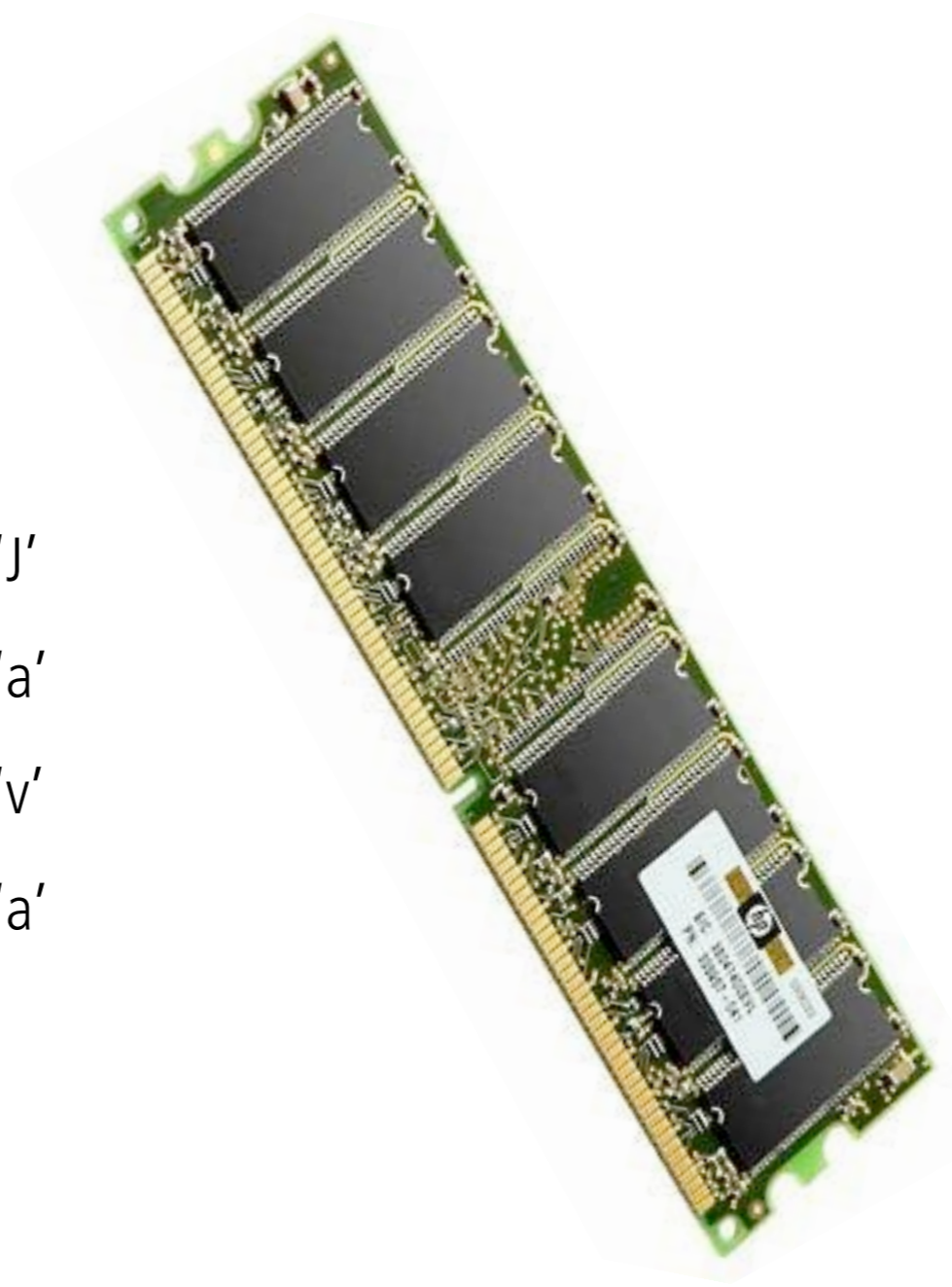
$$(A \vee B) \wedge B$$

$$(A \vee B) \wedge \neg B$$



# Hauptspeicher

•	•	
•	•	
•	•	
2000	01001010	Kodierung für Buchstabe 'J'
2001	01100001	Kodierung für Buchstabe 'a'
2002	01110110	Kodierung für Buchstabe 'v'
2003	01100001	Kodierung für Buchstabe 'a'
2004	00000011	Kodierung für Zahl 3





# Zahlensysteme

## **Dezimal**

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

## **Binär**

0, 1

## **Hexadezimal**

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

# Konversion: Dezimal nach Binär

Beispiel:  $123_{10}$

Quotient →	61	30	15	7	3	1	0
	$2 \overline{)123}$	$2 \overline{)61}$	$2 \overline{)30}$	$2 \overline{)15}$	$2 \overline{)7}$	$2 \overline{)3}$	$2 \overline{)1}$
	122	60	30	14	6	2	0
Rest →	1	1	0	1	1	1	1

← Leserichtung

Resultat:  $1111011_2$

# Aufgabe: Dezimal nach Binär

Aufgabe:  $17_{10}$  nach Binär

# Aufgabe: Dezimal nach Binär

## Aufgabe: $17_{10}$ nach Binär

Quotient →	8	4	2	1	0
	$\begin{array}{r} 2 \overline{) 17} \\ \underline{16} \end{array}$	$\begin{array}{r} 2 \overline{) 8} \\ \underline{8} \end{array}$	$\begin{array}{r} 2 \overline{) 4} \\ \underline{4} \end{array}$	$\begin{array}{r} 2 \overline{) 2} \\ \underline{2} \end{array}$	$\begin{array}{r} \overline{) 1} \\ \underline{0} \end{array}$
Rest →	1	0	0	0	1

# Aufgabe: Dezimal nach Binär

## Aufgabe: $17_{10}$ nach Binär

Quotient →	8	4	2	1	0
	$\begin{array}{r} 2 \overline{) 17} \\ \underline{16} \end{array}$	$\begin{array}{r} 2 \overline{) 8} \\ \underline{8} \end{array}$	$\begin{array}{r} 2 \overline{) 4} \\ \underline{4} \end{array}$	$\begin{array}{r} 2 \overline{) 2} \\ \underline{2} \end{array}$	$\begin{array}{r} \overline{) 1} \\ \underline{0} \end{array}$
Rest →	1	0	0	0	1

Resultat:  $10001_2$

# Konversion: Binär nach Dezimal

Beispiel:  $1111011_2$

	1	1	1	1	0	1	1	
Sum:	$1 \times 2^6$	$1 \times 2^5$	$1 \times 2^4$	$1 \times 2^3$	$0 \times 2^2$	$1 \times 2^1$	$1 \times 2^0$	
Sum:	64	32	16	8	0	2	1	→ 123

Resultat:  $123_{10}$



# Aufgabe: Binär nach Dezimal

Aufgabe:  $10101_2$  nach Dezimal

# Aufgabe: Binär nach Dezimal

Aufgabe:  $10101_2$  nach Dezimal

	1	0	1	0	1	
Sum:	$1 \times 2^4$	$0 \times 2^3$	$1 \times 2^2$	$0 \times 2^1$	$1 \times 2^0$	
Sum:	16	0	4	0	1	→ 21

# Aufgabe: Binär nach Dezimal

Aufgabe:  $10101_2$  nach Dezimal

	1	0	1	0	1	
Sum:	$1 \times 2^4$	$0 \times 2^3$	$1 \times 2^2$	$0 \times 2^1$	$1 \times 2^0$	
Sum:	16	0	4	0	1	→ 21

Resultat:  $21_{10}$

# Addieren von Binärzahlen

$$\begin{array}{r} + \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \\ \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \\ \hline | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \quad | \end{array}$$

The diagram illustrates the addition of two 10-bit binary numbers. The top row shows the first number: 1101100111. The second row shows the second number: 1101100111. A horizontal line separates the two numbers. The bottom row shows the result of the addition: 1101100111. The plus sign is positioned to the left of the first column. Vertical lines separate the columns, and small vertical tick marks are placed below the horizontal line in each column.

# Addieren von Binärzahlen

		1	1	0	1	1	0	1	1		
+		1	0	1	1	1	0	1	1		
<hr/>											
	1	1	0	1	0	1	1	0			

1

Kann zu einem Überlauf führen!



# Aufgabe: Addition

# Aufgabe: Addition

$$\begin{array}{r} + \\ \hline 1 \quad 0 \quad 1 \quad 1 \quad 0 \\ 1 \quad 0 \quad 0 \quad 1 \quad 1 \end{array}$$



# Sekundärspeicher



# Software



# Gängige Vorurteile/ Ausreden

*“Computer sind intelligent.”*

*“Der Computer ist abgestürzt.”*

*“Der Computer erlaubt das nicht.”*

*“Der Computer hat die Datei verloren/  
kaputt gemacht.”*

# Computer...

...sind universelle Maschinen. Sie führen das Programm aus, das man ihnen eingibt.

Die guten Neuigkeiten:

1. *Der Computer wird **genau** das tun, was das Programm ihm vorschreibt.*
2. *Er wird es **sehr schnell** tun.*

Die schlechten Neuigkeiten:

1. *Der Computer wird **genau** das tun, was das Programm ihm vorschreibt.*
2. *Er wird es **sehr schnell** tun.*

# Software schreiben ist schwierig

Programme stürzen ab

Programme, die nicht abstürzen,  
funktionieren aber auch nicht  
unbedingt richtig

Programmierer sind verantwortlich für  
das korrekte Funktionieren ihrer  
Programme

# Beispiel: Ariane 5

4. Juni 1996: Millionen Dollar Schaden wegen einfachem Programmierfehler:

64-bit float -> 32-bit int





# Weitere Beispiele

## **28. July, 1962 - Mariner I space probe.**

Eine mit Bleistift niedergeschriebene Formel wird falsch abgeschrieben.  
Resultat: Die Sonde muss gesprengt werden.

## **1982 - Soviet gas pipeline.**

Die CIA lies absichtlich einen Fehler in eine Steuerungssoftware einbauen.  
Resultat: Die grösste nicht-nukleare Explosion in der Geschichte der Menschheit.

## **1985-1987 - Therac-25 medical accelerator**

Gerät zur Strahlentherapie dessen Betriebssystem von einem unerfahrenen Programmierer zusammengebastelt worden war. Resultat: Mindestens 5 Patienten sterben, mehrere werden schwer verletzt.



# Software schreiben macht Spass

Ein Programmierer entwirft und baut eigene Maschinen

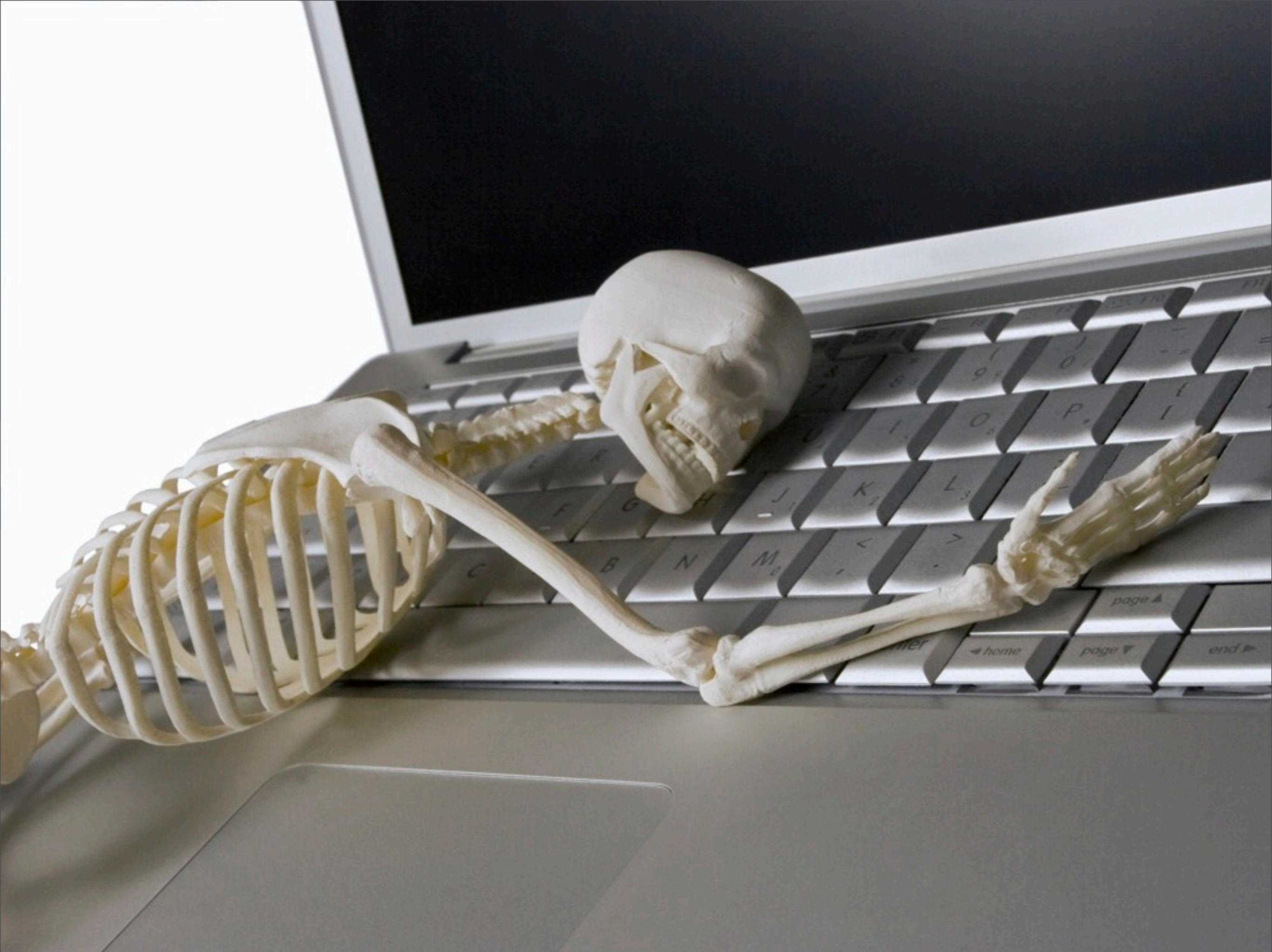
Er kann kreativ sein und seine Vorstellungskraft gebrauchen

Es ist faszinierend, wenn ein selbstgeschriebenes Programm läuft und vielleicht sogar seinen Benutzern den Alltag erleichtert

Wie bringe ich den Computer dazu,  
das zu tun, was ich will?







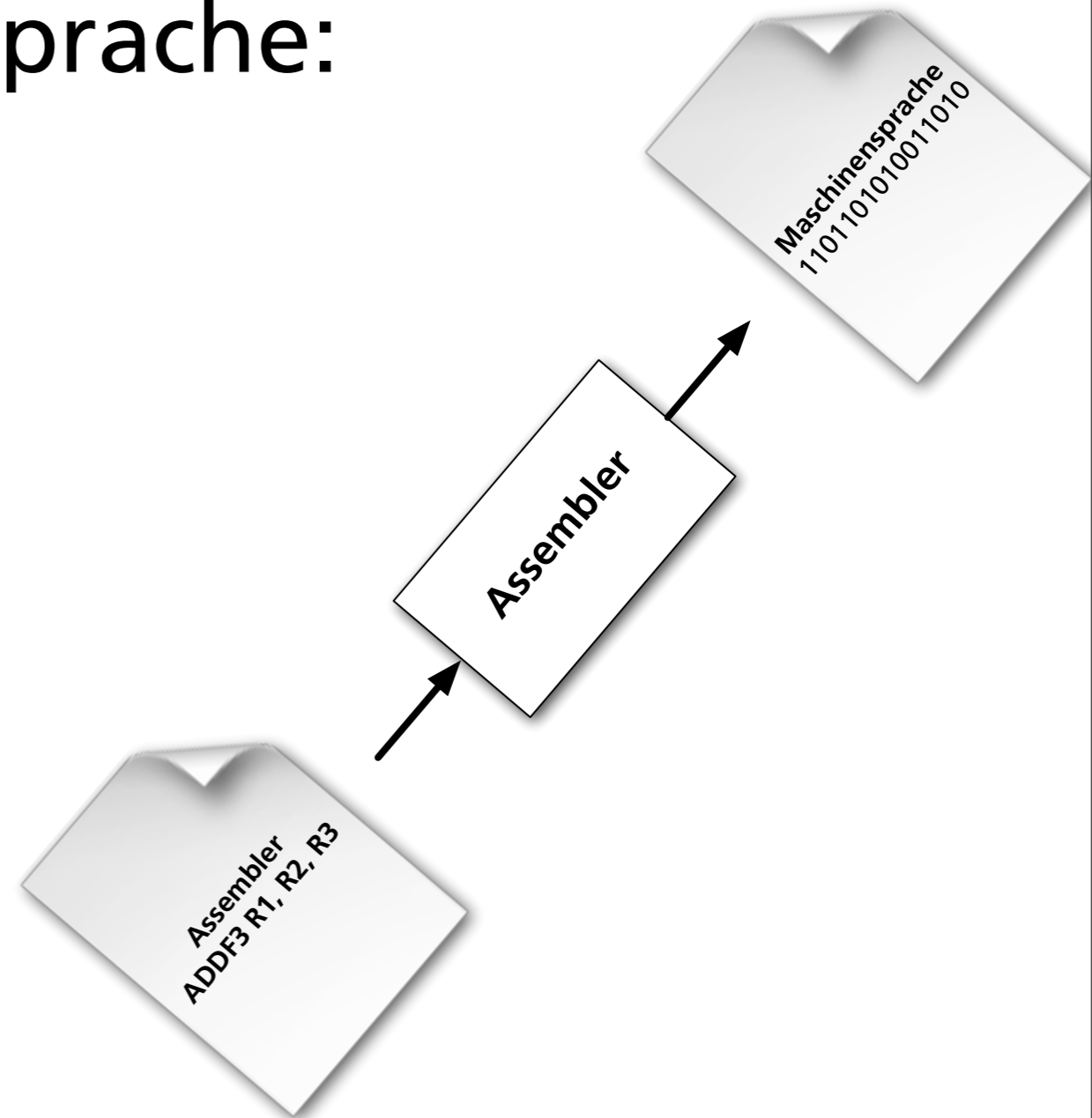
# Maschinensprache vs. Assembler

Addieren in Maschinensprache:

```
1101101010011010
```

Addieren in Assembler:

```
ADDF3 R1, R2, R3
```



# Höhere Programmiersprachen

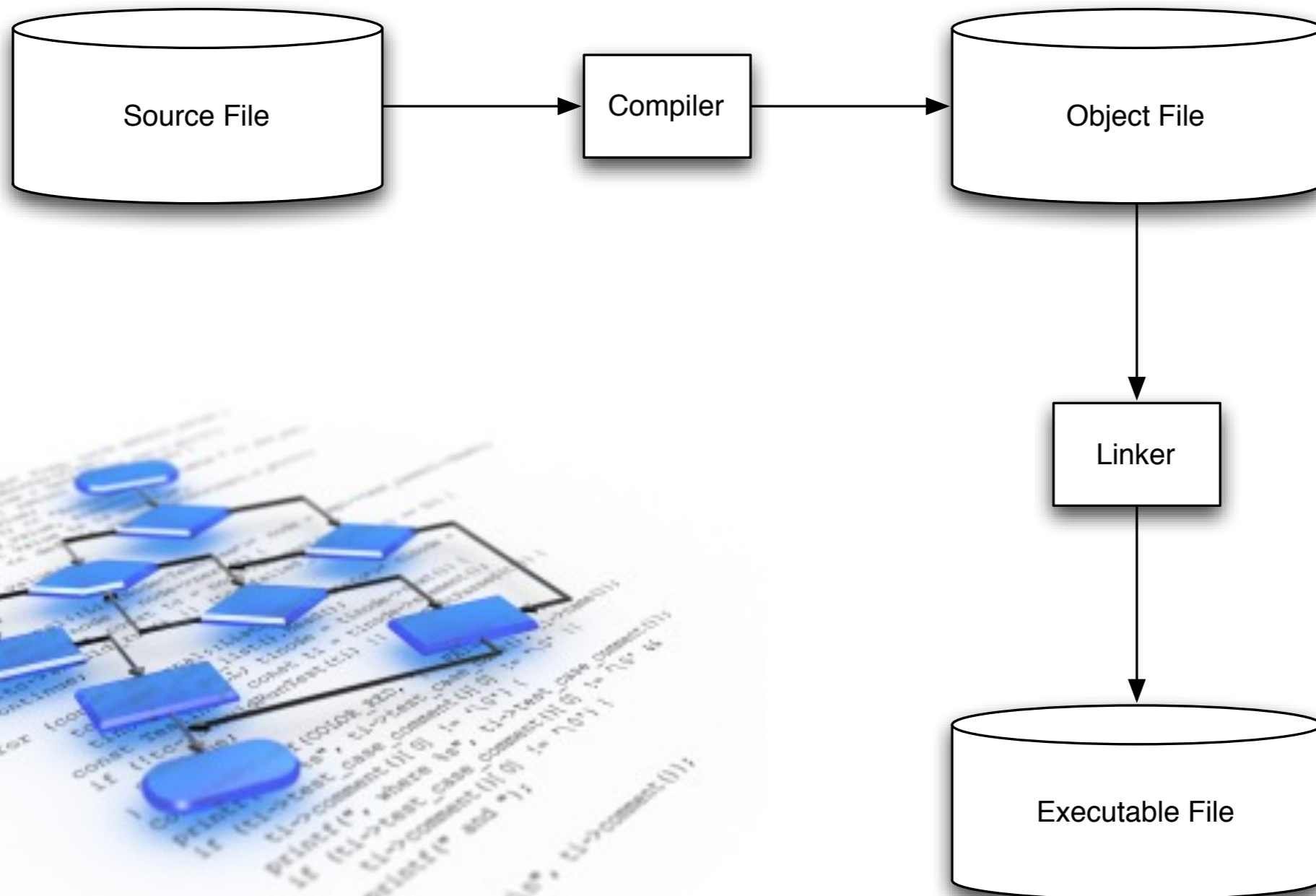
- Java
- COBOL (COmmon Business Oriented Language)
- FORTRAN (FORmula TRANslation)
- BASIC (Beginner All-purpose Symbolic Instructional Code)
- Pascal (benannt nach Blaise Pascal)
- Ada (benannt nach Ada Lovelace)
- C
- Visual Basic
- Delphi
- C++
- ...

## Code-Beispiele:

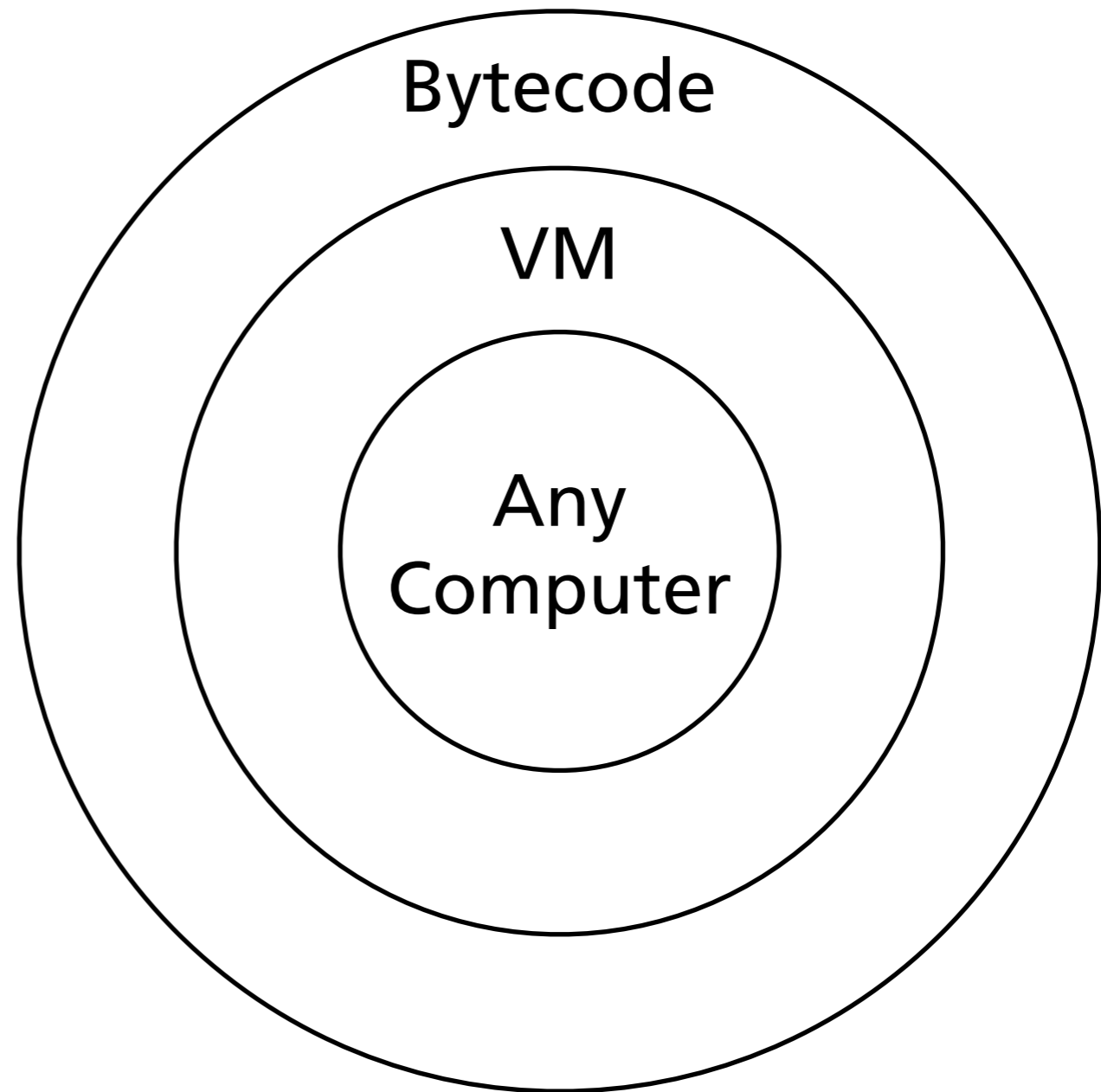
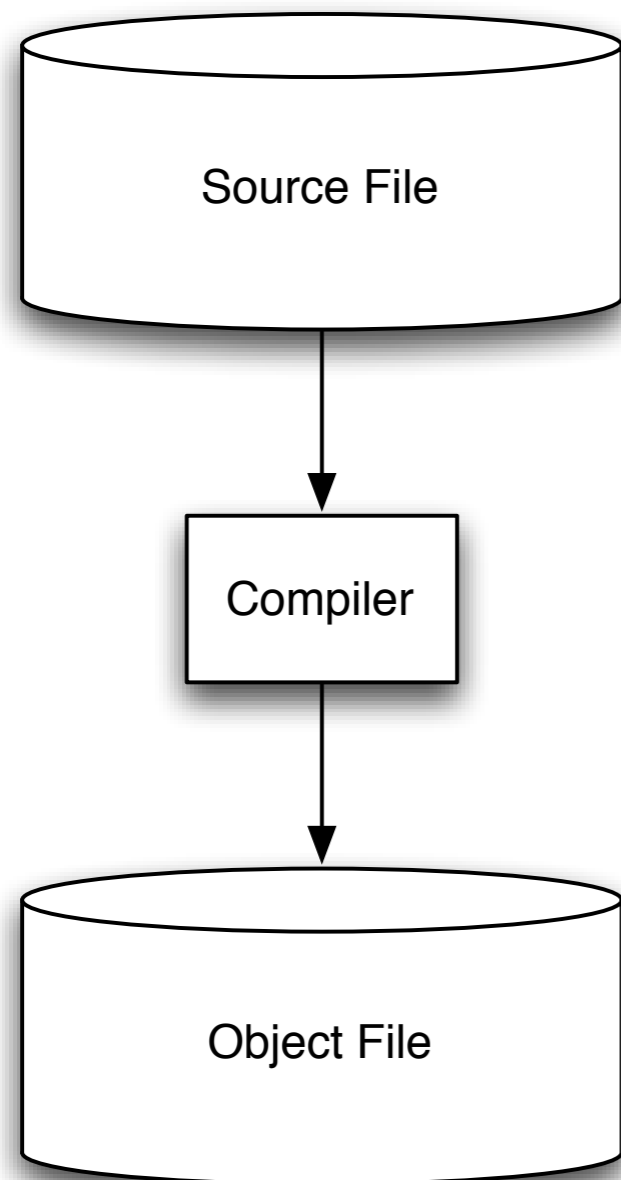
```
float area = 5 * 5 * 3.1415;  
new Window("This is the title").setVisible(true);
```



# Der Kompiliervorgang



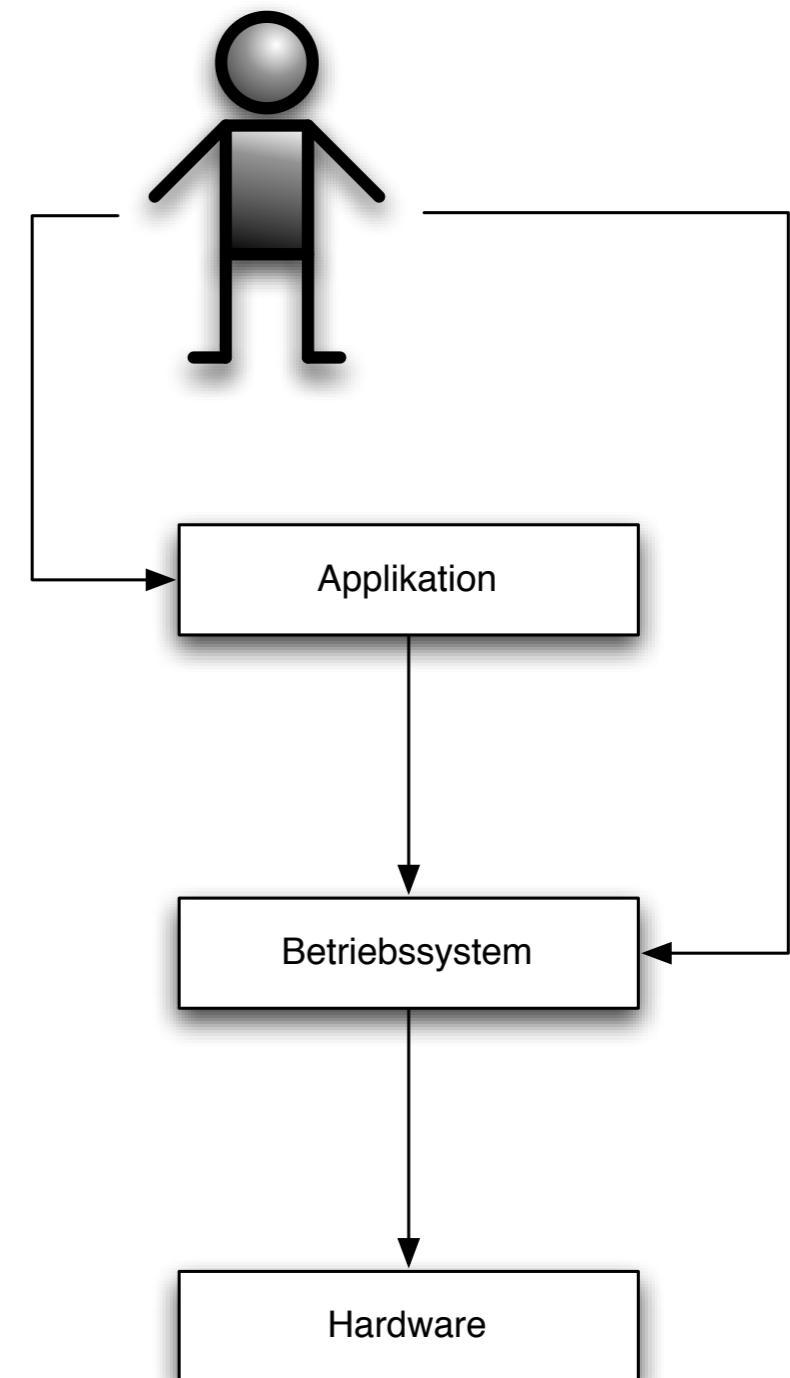
# Virtuelle Maschinen





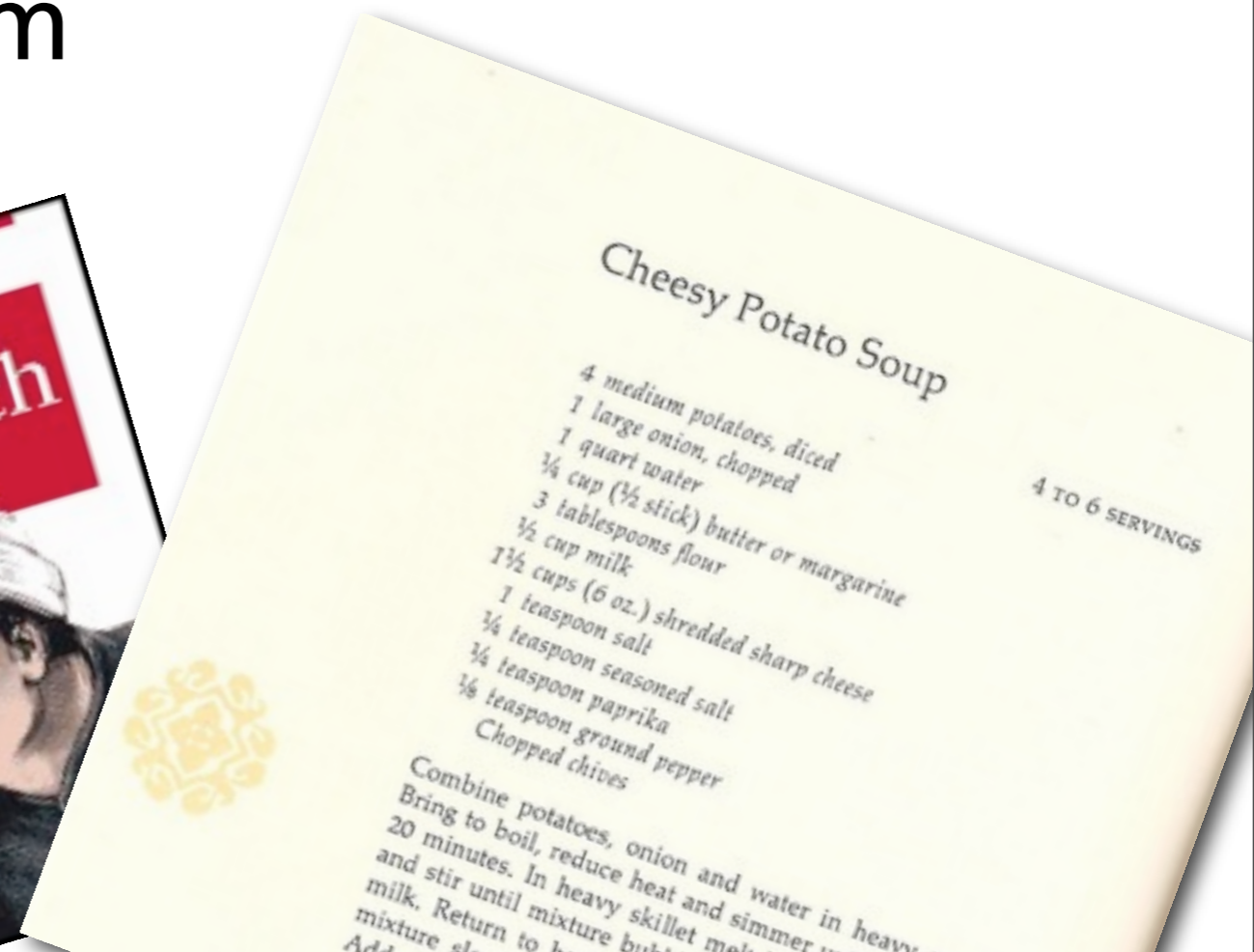
# Das Betriebssystem

- Controlling und Monitoring von Systemaktivitäten
- Allokation und Zuweisung von Systemressourcen
- Scheduling von Prozessen



# Algorithmisches Denken

## Vom Rezept zum ausführbaren Computerprogramm



# Der Risotto-Algorithmus


## Zutaten für 2 Personen:



1 dl	Wein
ca. 8 dl	Rinderbouillon
1x	kleine Zwiebel
1-2	Knoblauchzehe(n)
2 Esslöffel	Olivenöl
150 g	Rundkorn Reis
50 g	Parmesan
eine Messerspitze	Safranfäden

1. Das Olivenöl erhitzen, die Zwiebel und den Knoblauch fein würfeln und darin anschwitzen.
2. Den Reis begeben und 3-4 Minuten unter Rühren glasig werden lassen.
3. Mit dem Wein ablöschen und solange köcheln lassen, bis die Flüssigkeit verdunstet ist.
4. Den Safran begeben.
5. Die Rinderbouillon aufkochen und eine Kelle voll zum Reis hinzugeben und verdunsten lassen.
6. Diesen Vorgang solange wiederholen, bis sämtliche Bouillon aufgebraucht ist.
7. Den Parmesan unterrühren und heiss servieren.

# Weitere Algorithmen im Alltag

Prozess	Ausführender	Algorithmus	typische Anweisung
Kuchen backen	Bäcker	Rezept	nimm 1/2 kg Mehl...
Spielen einer Klaviersonate	Pianist	Partitur	
Bedienung eines Handys	Anrufer	Bedienungsanleitung	drücken sie die # Taste
Bau eines Radios	Bastler	Schaltplan und Montageanleitung	verbinde Transistor T1 mit T5

# Der Euklidische Algorithmus



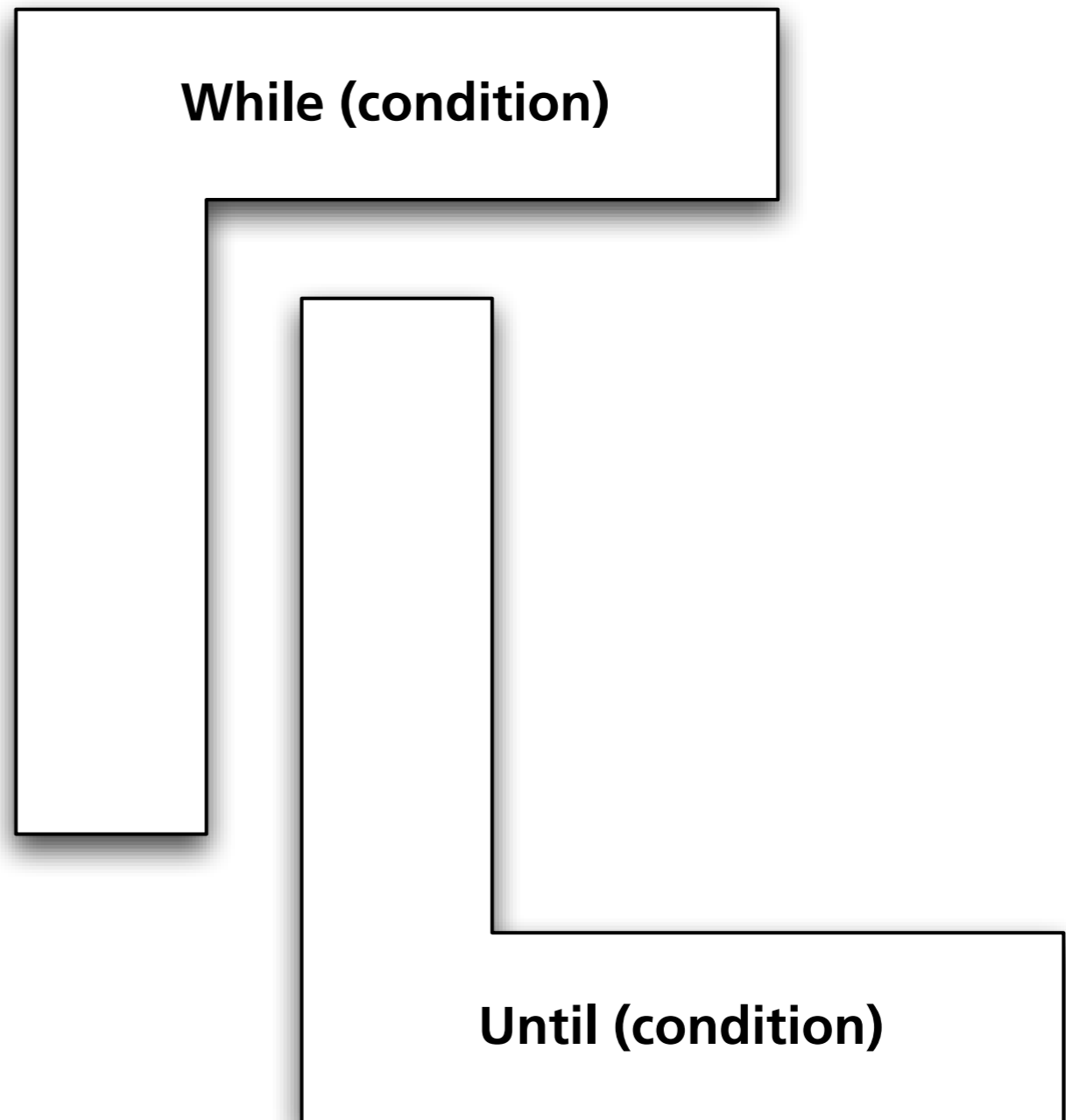
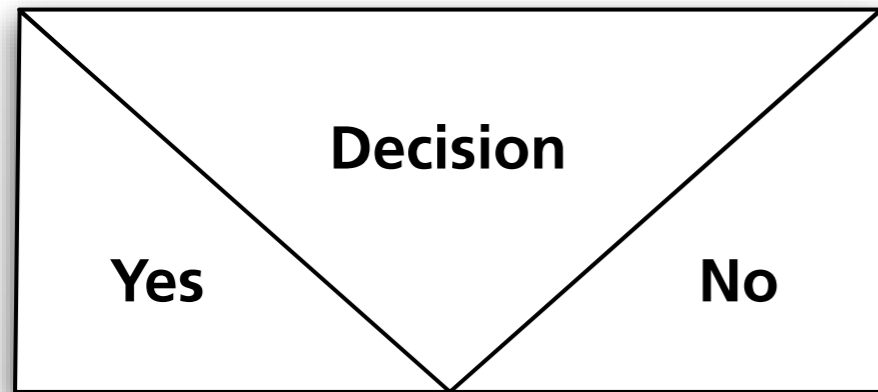
Der Euklidische Algorithmus (um 300 v. Chr. beschrieben) dient zur Ermittlung des **grössten gemeinsamen Teilers (ggT)** zweier natürlicher Zahlen  $A$  und  $B$ .

1. Sei  $A$  die grössere der beiden Zahlen  $A$  und  $B$  (entsprechend vertauschen, falls dem noch nicht so ist)
2. Setze  $A = A - B$
3. Wenn  $A$  und  $B$  ungleich sind, dann fahre fort mit Schritt 1. Wenn sie gleich sind, dann beende den Algorithmus: Diese Zahl ist der ggT

# Beispiel: ggT von 14 und 8

Schritt	A	B	A - B
1.	14	8	6
2.	8	6	2
3.	6	2	4
4.	4	2	2
5.	2	2	gleich

# Nassi-Shneiderman Diagramme



# Aufgabe: Der Risotto-Algorithmus als NSD

## Zutaten für 2 Personen:

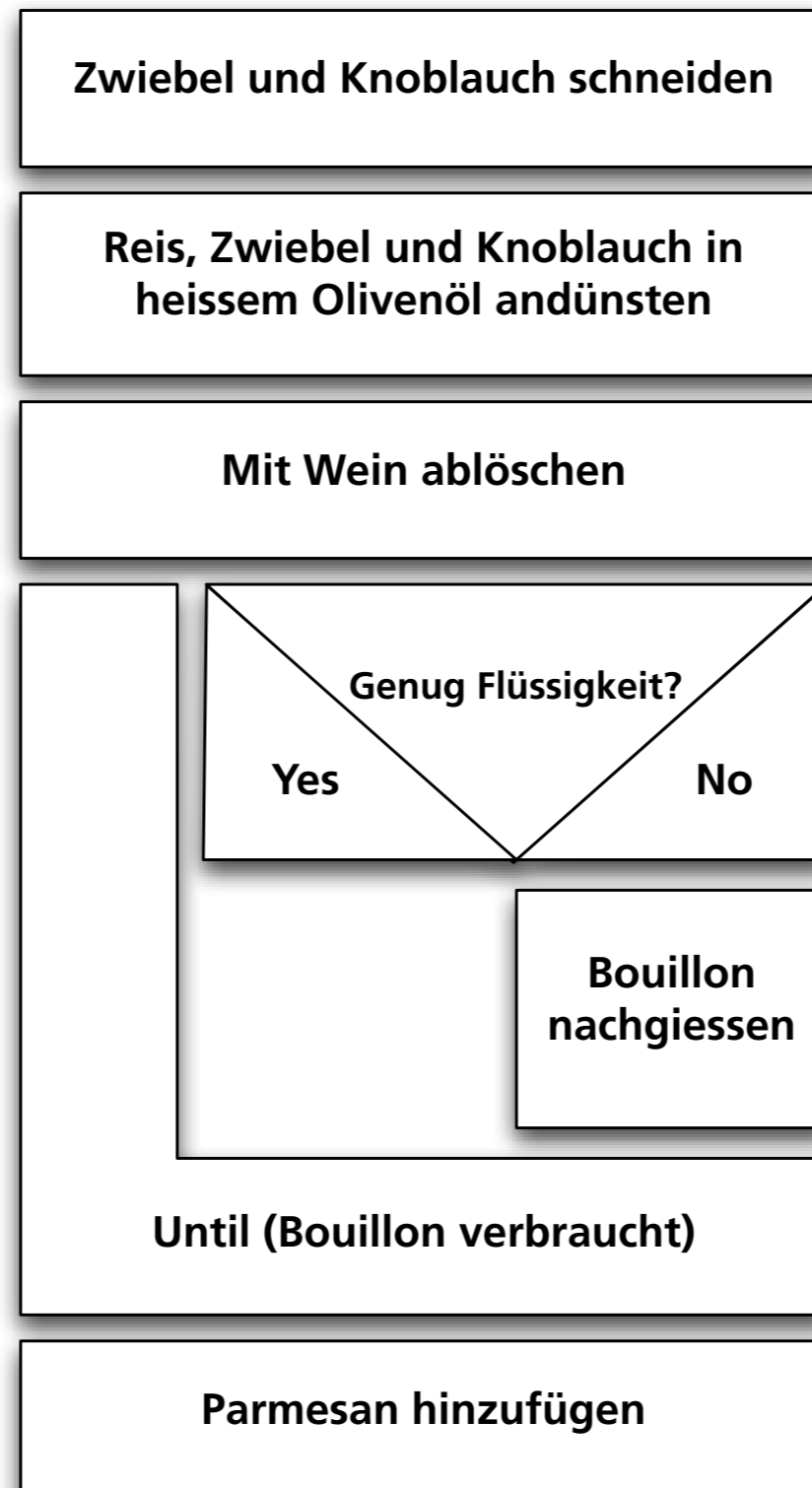


1 dl	Wein
ca. 8 dl	Rinderbouillon
1x	kleine Zwiebel
1-2	Knoblauchzehe(n)
2 Esslöffel	Olivenöl
150 g	Rundkorn Reis
50 g	Parmesan
eine Messerspitze	Safranfäden

1. Das Olivenöl erhitzen, die Zwiebel und den Knoblauch fein würfeln und darin anschwitzen.
2. Den Reis begeben und 3-4 Minuten unter Rühren glasig werden lassen.
3. Mit dem Wein ablöschen und solange köcheln lassen, bis die Flüssigkeit verdunstet ist.
4. Den Safran begeben.
5. Die Rinderbouillon aufkochen und eine Kelle voll zum Reis hinzugeben und verdunsten lassen.
6. Diesen Vorgang solange wiederholen, bis sämtliche Bouillon aufgebraucht ist.
7. Den Parmesan unterrühren und heiss servieren.



# Lösung: Der Risotto-Algorithmus als NSD



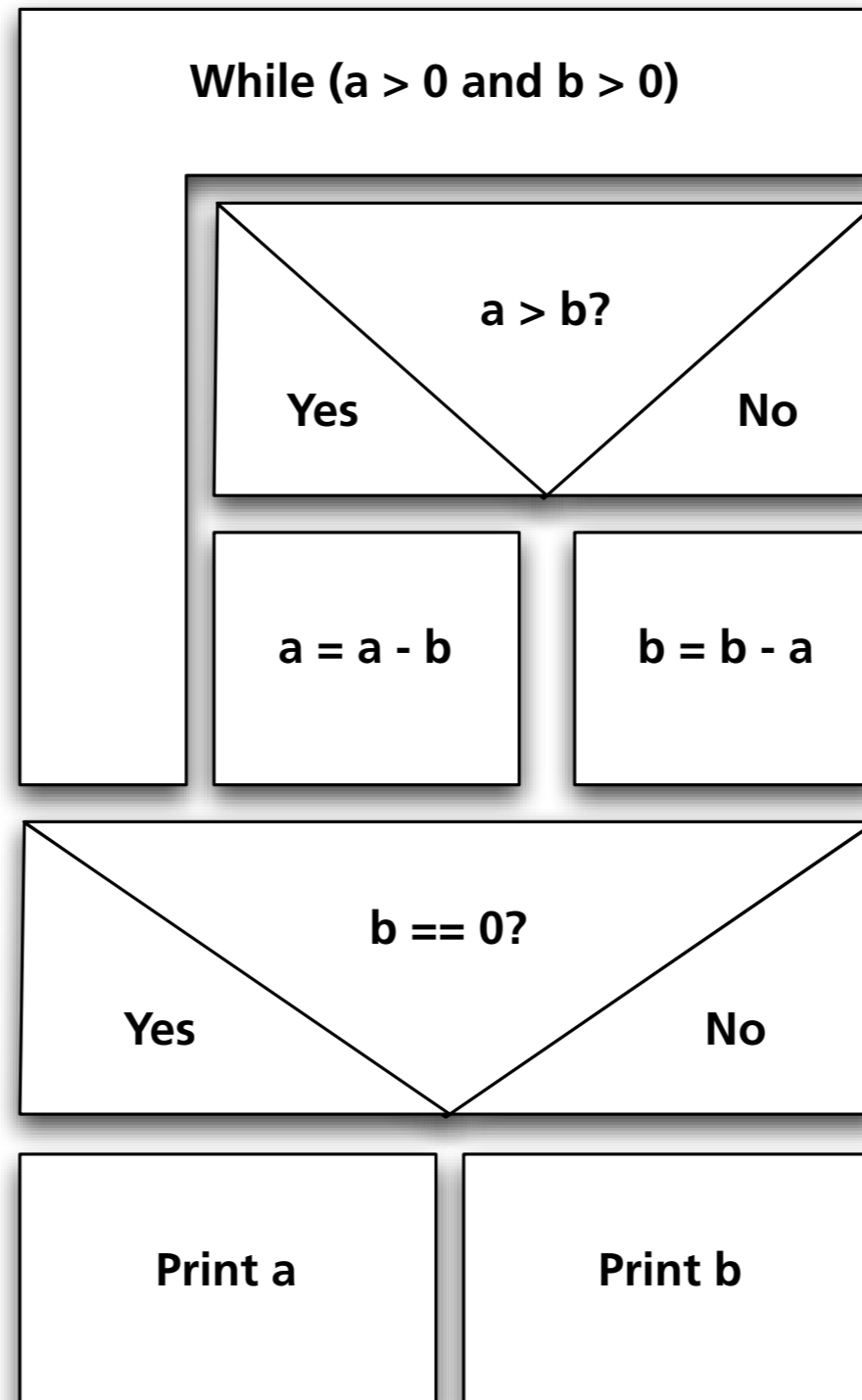
# Aufgabe: Der Euklidische Algorithmus als NSD



Der Euklidische Algorithmus (um 300 v. Chr. beschrieben) dient zur Ermittlung des **grössten gemeinsamen Teilers (ggT)** zweier natürlicher Zahlen  $A$  und  $B$ .

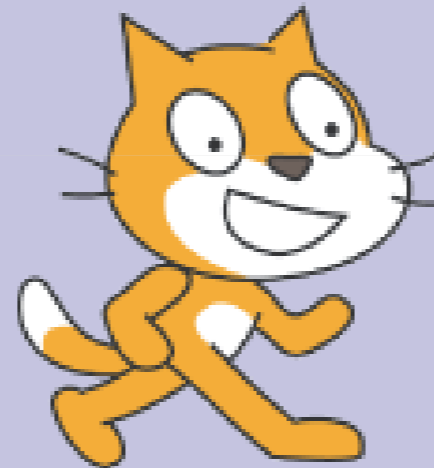
1. Sei  $A$  die grössere der beiden Zahlen  $A$  und  $B$  (entsprechend vertauschen, falls dem noch nicht so ist)
2. Setze  $A = A - B$
3. Wenn  $A$  und  $B$  ungleich sind, dann fahre fort mit Schritt 1. Wenn sie gleich sind, dann beende den Algorithmus: Diese Zahl ist der ggT

# Lösung: Der Euklidische Algorithmus als NSD





Getting Started with  
**SCRATCH**  
version 1.4



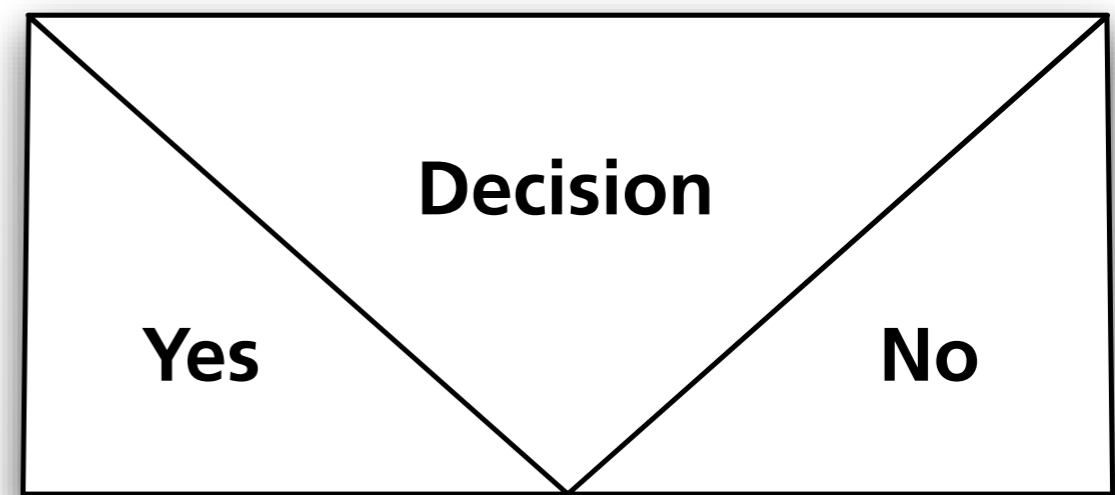
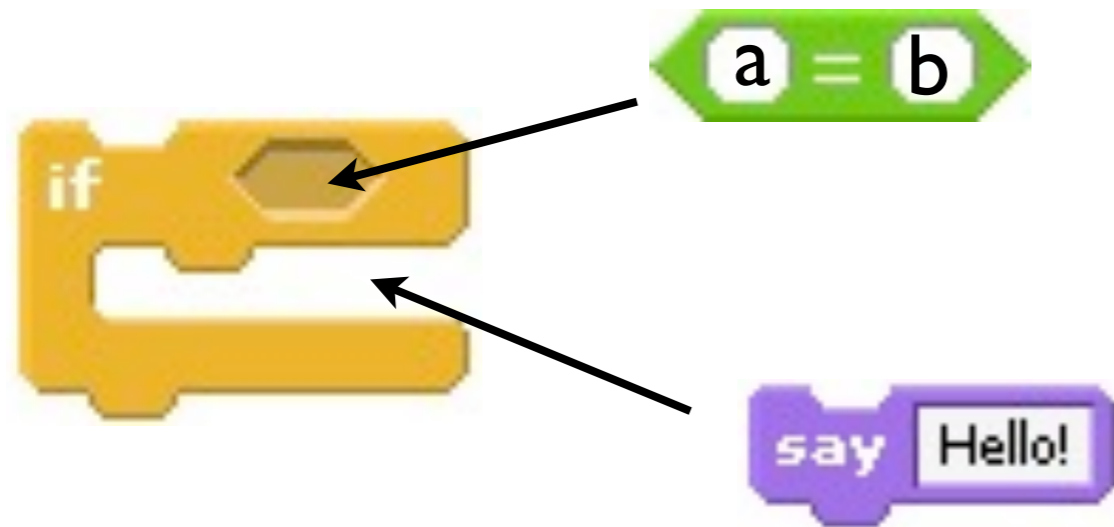
<http://scratch.mit.edu>

# Algorithmen in Scratch: Statements

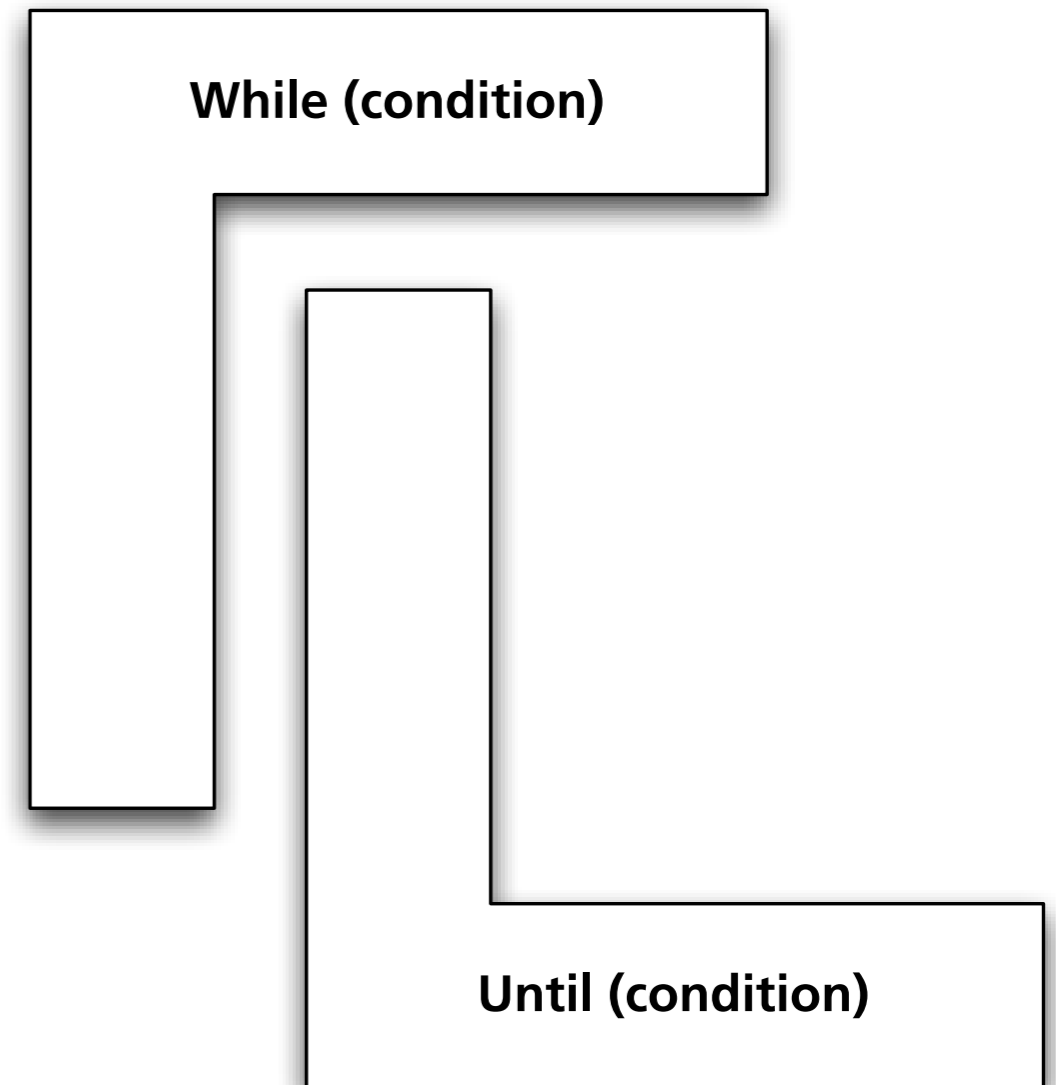
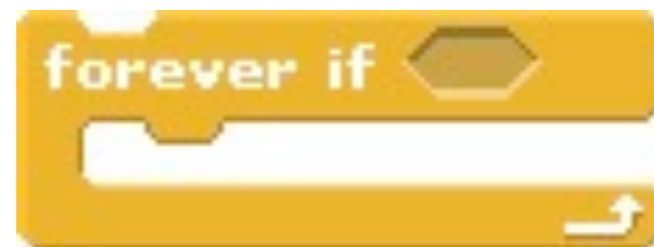


**Statement**

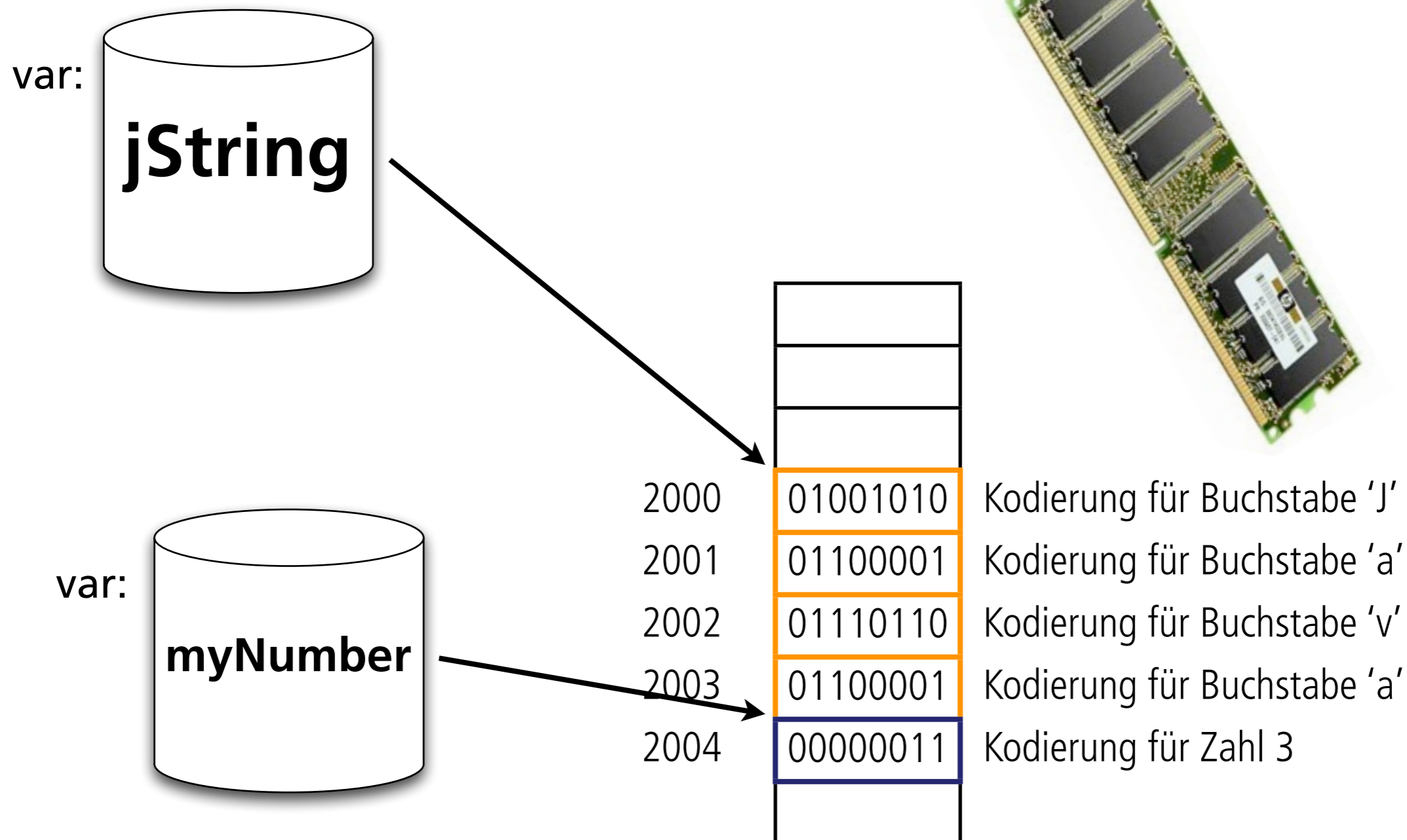
# Algorithmen in Scratch: Decisions



# Algorithmen in Scratch: Loops



# Variablen in einem Programm





# Variablen in Scratch

Make a variable

variable

Delete a variable

set score ▼ to 0

# SCRATCH 1.4 INTERFACE

The image shows the Scratch 1.4 interface with several components labeled and explained:

- SAVE LANGUAGE**: Located at the top left of the interface.
- SHARE**: Located at the top left of the interface.
- SPRITE ROTATION STYLE**: Located at the top left of the interface.
- CURRENT SPRITE INFO**: Located at the top left of the interface.
- TABS**: Located at the top left of the interface. Edit scripts, costumes, or sounds.
- TOOLBAR**: Located at the top left of the interface.
- VIEW MODE**: Located at the top left of the interface. Change to large or small stage view.
- PRESENTATION MODE**: Located at the top left of the interface. Present your project.
- BLOCKS PALETTE**: Located on the left side of the interface. Blocks for programming your sprites.
- SCRIPTS AREA**: Located in the center of the interface. Drag blocks in, snap them together into scripts.
- GREEN FLAG**: Located at the top right of the interface. A way to start scripts.
- STOP SIGN**: Located at the top right of the interface. Stops all scripts.
- STAGE**: Located in the center of the interface. Where your Scratch creations come to life.
- MOUSE X-Y DISPLAY**: Located at the bottom right of the interface. Shows location of cursor.
- NEW SPRITE BUTTONS**: Located at the bottom right of the interface. Create a new character or object for your project.
- SPRITE LIST**: Located at the bottom right of the interface. Thumbnails of all your sprites. Click to select and edit a sprite.