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

s.e.a.l.
software evolution & architecture lab

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)

s.e.a.l.
software evolution & architecture lab

VMML
Surfing and snowboarding lab
university of zürich

CSG Communication Systems Group

DIS
Dynamic and Distributed
Information Systems

Information Management
Research Group

RE
RG
IFI
REQUIREMENTS ENGINEERING
RESEARCH GROUP

ai lab

Database Technology Research Group

IFI
educational engineering

Institut für Computerlinguistik

http://www.ifi.uzh.ch

Wer sind wir?

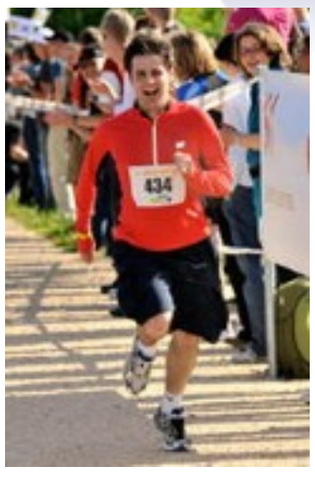




s.e.a.l.

software evolution & architecture lab





http://seal.ifi.uzh.ch/people

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)

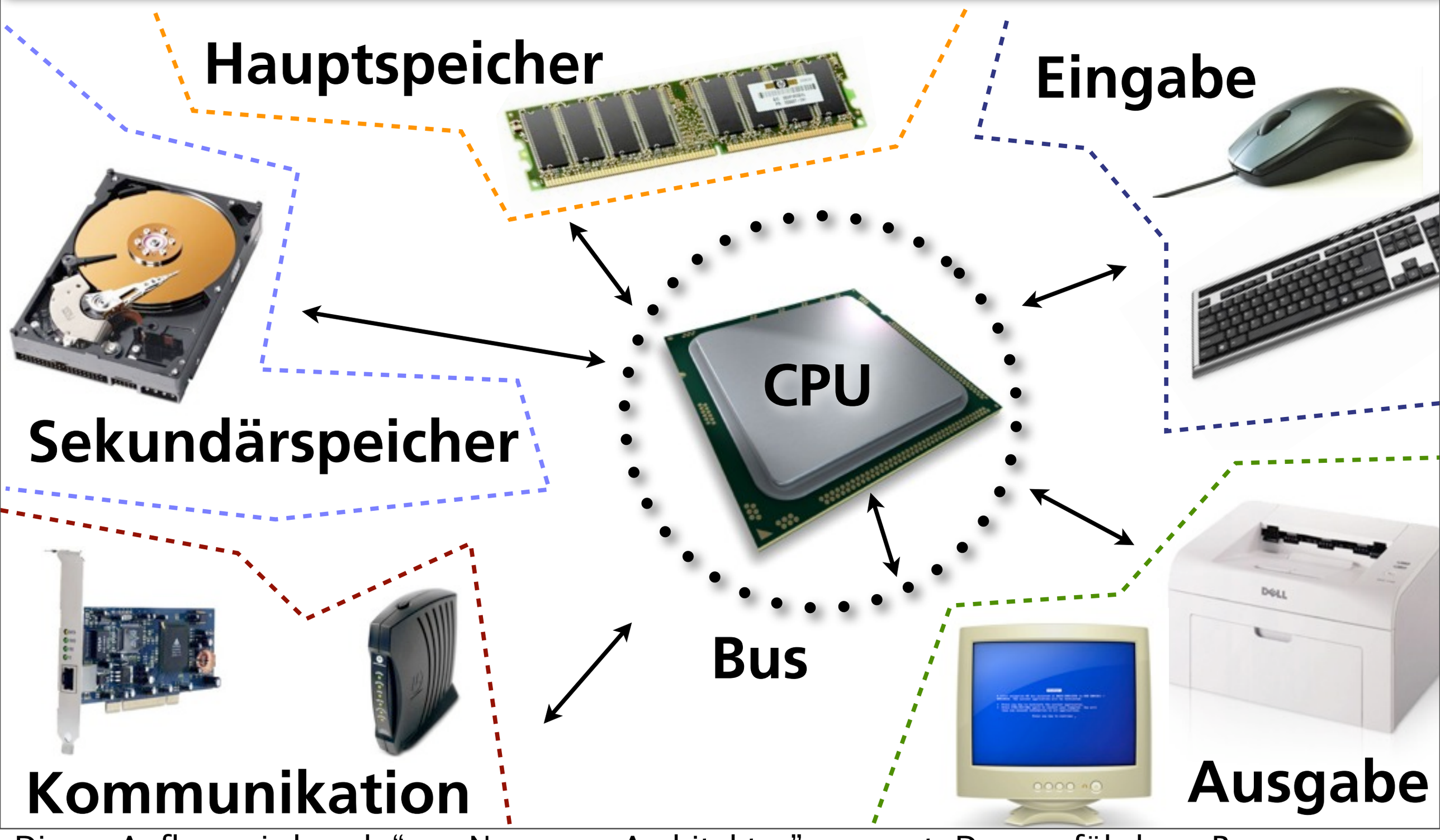
Sopra Beispiele.

Was ist ein Computer?

Ein Computer ist ein elektronisches Gerät, das Daten (Informationen) speichert und verarbeitet. Ein Computer beinhaltet sowohl Hardware, wie auch Software. Hardware ist der physische Teil, sprich der “sichtbare”, während Software die unsichtbaren Anweisungen zur Steuerung der Hardware umfasst.

Programmieren heisst, dass man Anweisungen niederschreibt, die der Computer ausführen soll.

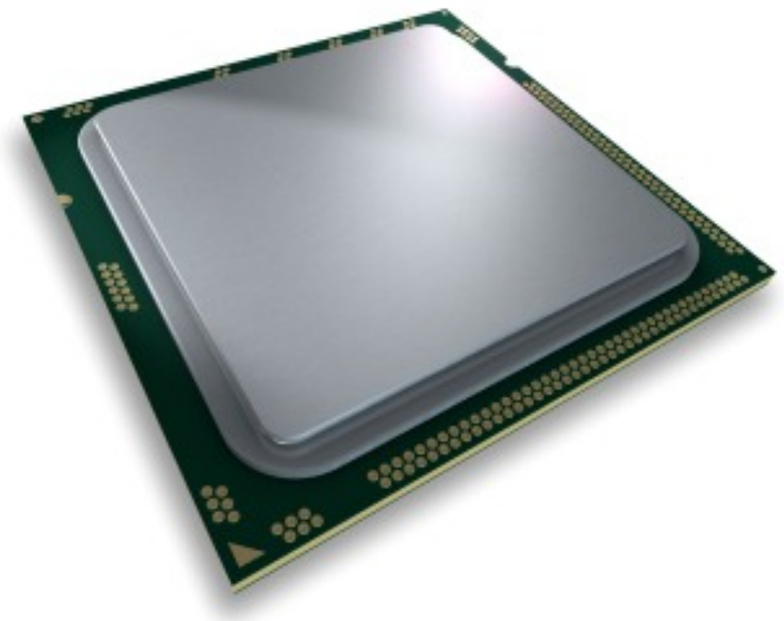
Wie ist ein Computer aufgebaut?



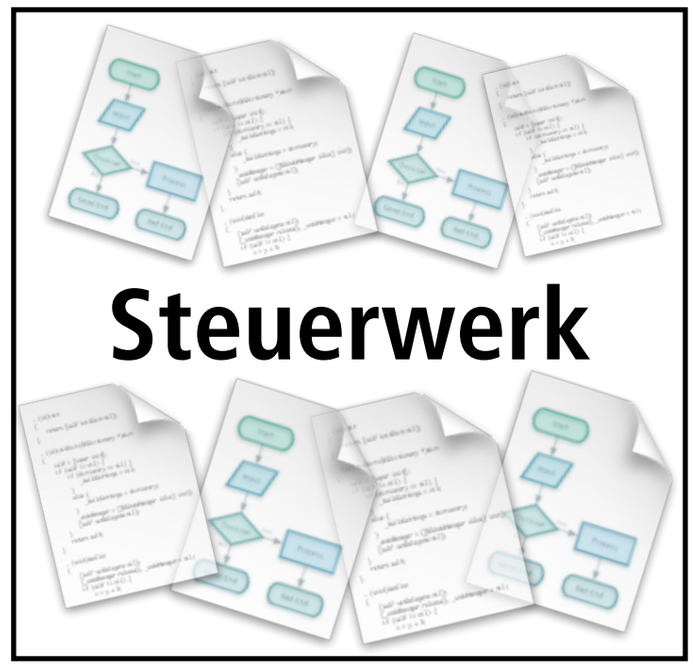
Dieser Aufbau wird auch “von Neumann Architektur” genannt. Das ausführbare Programm wird im Hauptspeicher gehalten. In früheren Computern war das Programm “hard-wired”, also fest in der Hardware verdrahtet.

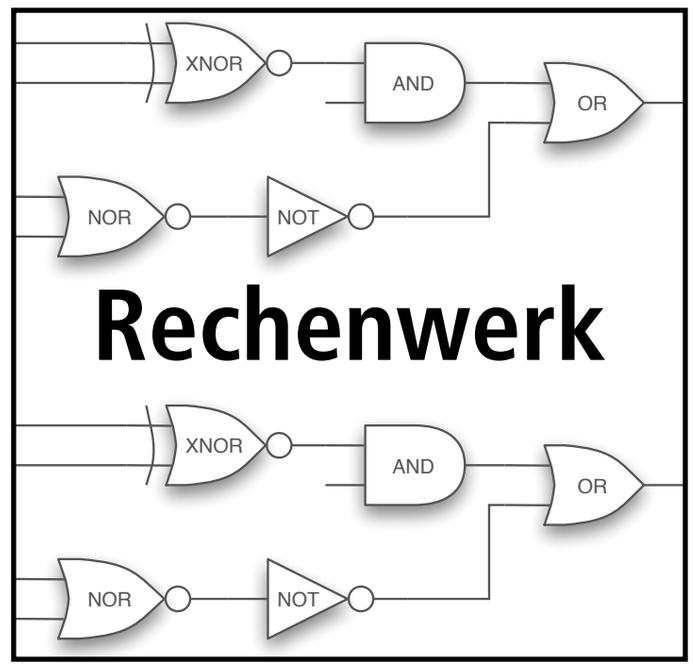
Der Prozessor kommuniziert über den Bus mit den peripheren Geräten, z.B.: Festplatten-Controller teilt dem Prozessor über einen sog. Interrupt mit “Hey, ich habe Daten, die bereit sind gelesen zu werden”. Der Prozessor liest dann den Speicherbereich aus, der der Festplatte entspricht.

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



=





Der Prozessor (CPU) ist das ‘Hirn’ eines Computers. Er lädt Anweisungen aus dem Speicher und führt diese aus. Ein Prozessor besteht normalerweise aus zwei Komponenten: Steuerwerk und Rechenwerk. Das Steuerwerk koordiniert die Aktionen der anderen Komponenten. Das Rechenwerk (oder die arithmetische und logische Einheit) führt numerische Operationen, wie Addition, Subtraktion, Multiplikation, Division), und logische Operationen (Vergleiche) durch.

Bool'sche Algebra

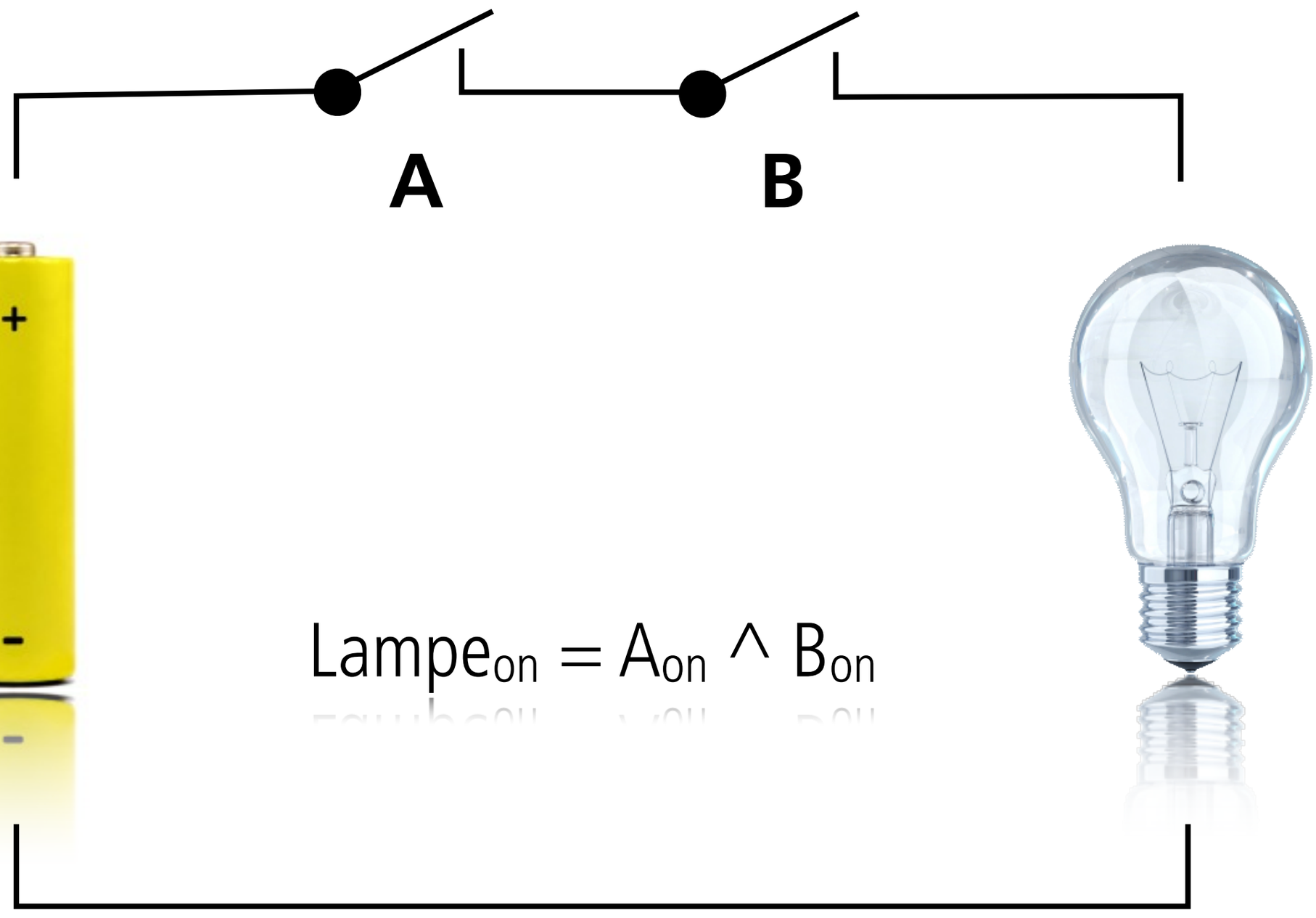
wahr oder falsch?

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

$Bestellung_{ok} = Konto_{Existiert}$ und $Kreditkarte_{gültig}$

In der Bool'schen Algebra – auch Bool'sche Logik genannt – geht es um Wahrheitswerte. Bool'sche Algebra definiert eine Menge von Operationen auf der Menge {0, 1} – oder auch {false, true}, {F, T}, etc.

Bool'sche Algebra: AND

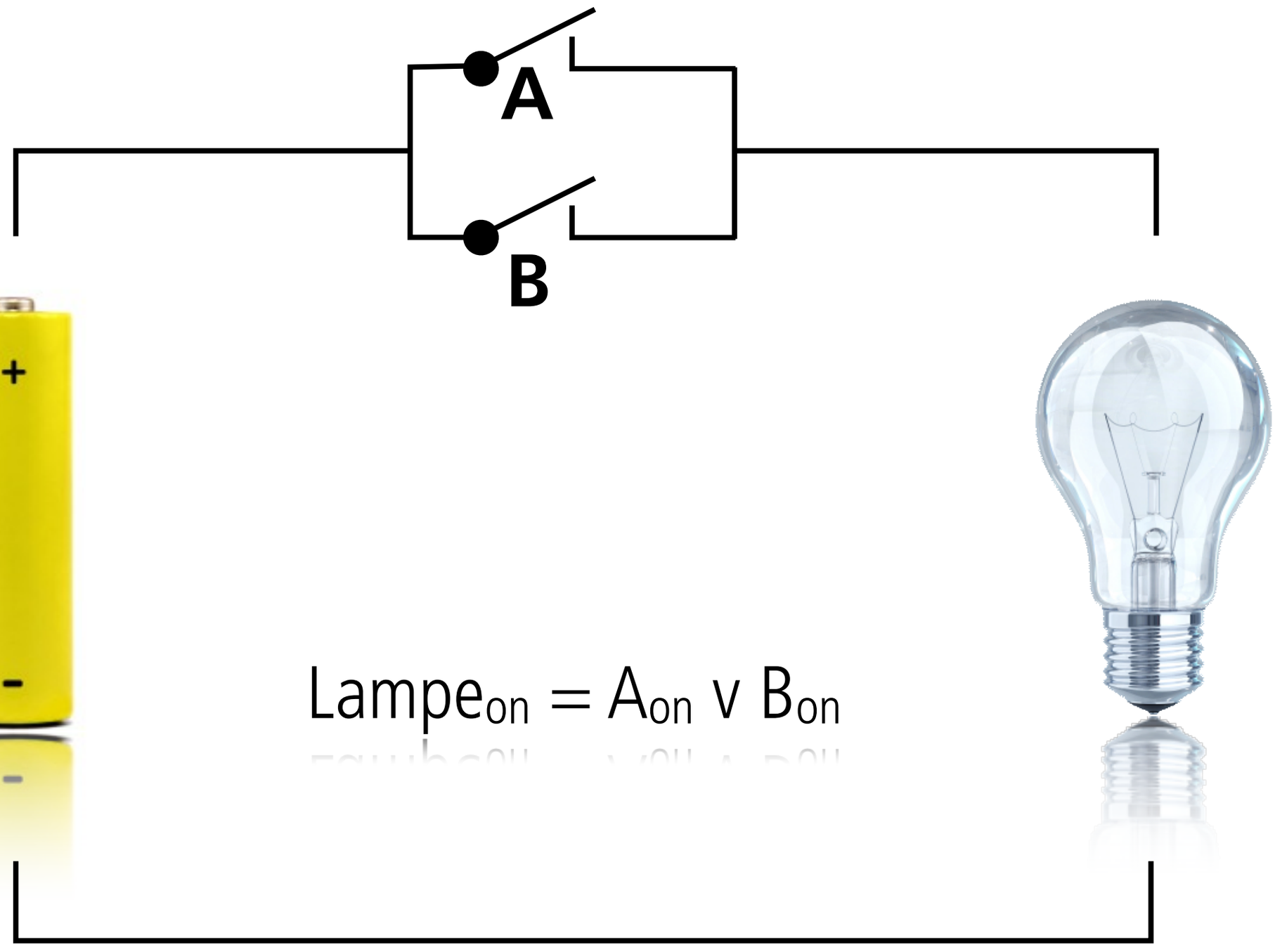


Der Ausdruck C = A ^ B ist als Ganzes genau dann wahr, wenn A wahr ist und B wahr ist.

Wahrheitstabelle: AND

AND	T	F
T	T	F
F	F	F

Bool'sche Algebra: OR



Der Ausdruck C = A v B ist als Ganzes genau dann wahr, wenn entweder A wahr ist oder B wahr ist, oder wenn A und B beide wahr sind. Achtung: Das bool'sche 'Or' entspricht nicht dem umgangsprachlichen 'Oder'!

Wahrheitstabelle: OR

OR	T	F
T	T	T
F	T	F

Aufgabe: Bool'sche Ausdrücke

Wann sind die folgenden Ausdrücke wahr?

(A v B) ^ B

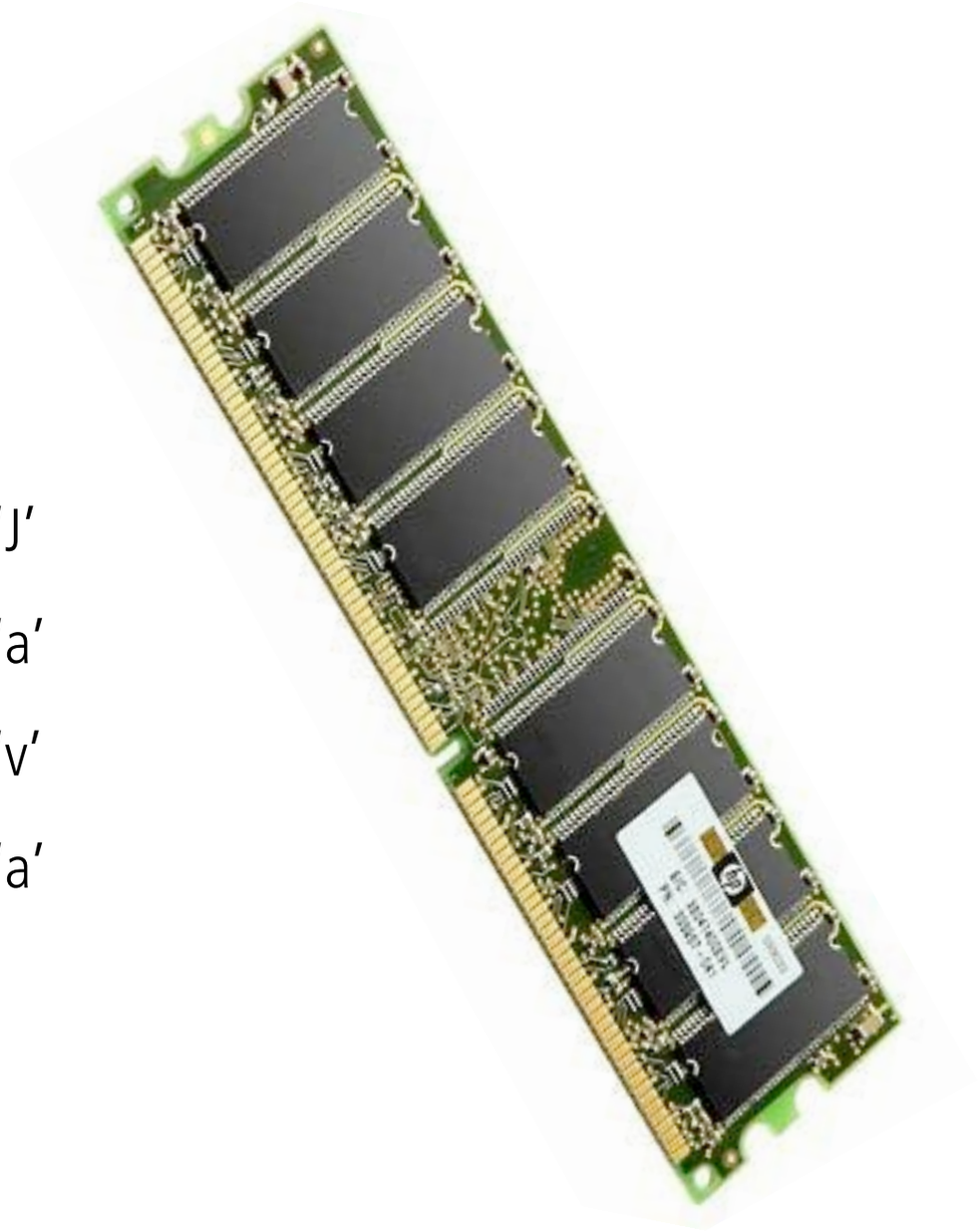
(A v B) ^ ¬B

Damit (A v B) ^ B wahr ist, muss B = T sein (bei and müssen beide Operanden wahr sein, damit der ganze Ausdruck wahr ist), A kann entweder wahr oder falsch sein.

Damit (A v B) ^ ¬B wahr ist, müssen B = F und A = T sein.

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



Hauptspeicher (oder Arbeitsspeicher/(R)andom (A)ccess (M)emory) speichert unterschiedliche Arten von Daten (Zeichenketten/Wörter, Buchstaben, Zahlen) in Form von Nullen und Einsen, sog. Bits ((B)inary Dig(its)). Der Speicher ist als eine Sequenz von Bytes organisiert, jedes Byte ist acht Bit lang und hat eine eindeutige Adresse.

Als Programmierer müssen wir uns glücklicherweise keine Gedanken über das Enkodieren und Dekodieren von Daten machen; Dies wird vom System automatisch auf Grund eines speziellen Kodierungsschemas vorgenommen. Der Buchstabe ‘J’ entspricht z.B. dem Byte 01001010 nach dem weit verbreiteten ASCII Schema.

Eine kleine Zahl wie ‘3’ passt gut in ein Byte. Wenn der Computer eine sehr grosse Zahl speichern muss, welche nicht in ein Byte passt, dann werden mehrere aufeinanderfolgende Bytes verwendet.

Zahlensysteme

Dezimal

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

Binär

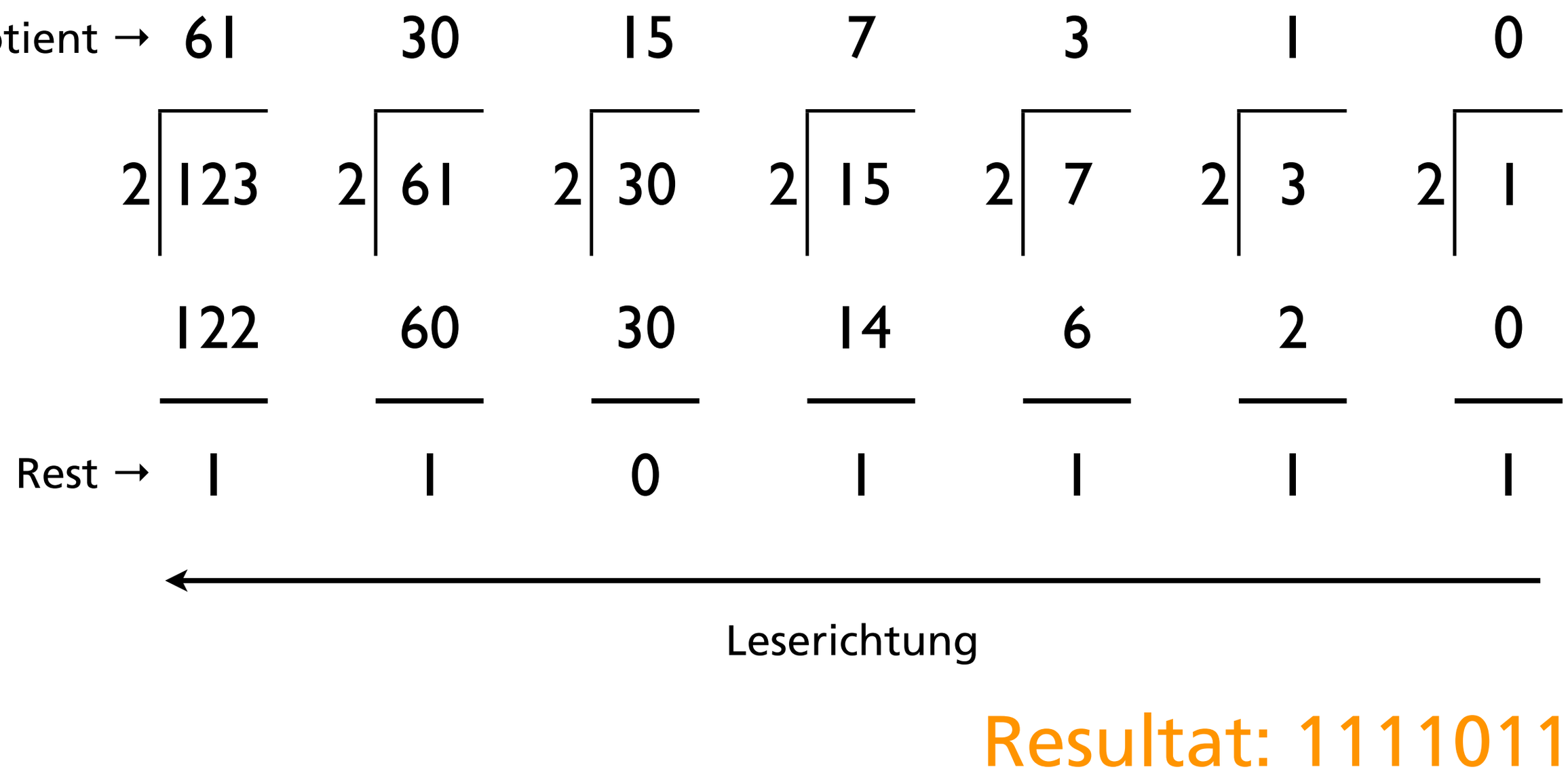
0, 1

Hexadezimal

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

Konversion: Dezimal nach Binär

Beispiel: 123_{10}



Resultat: 1111011_2

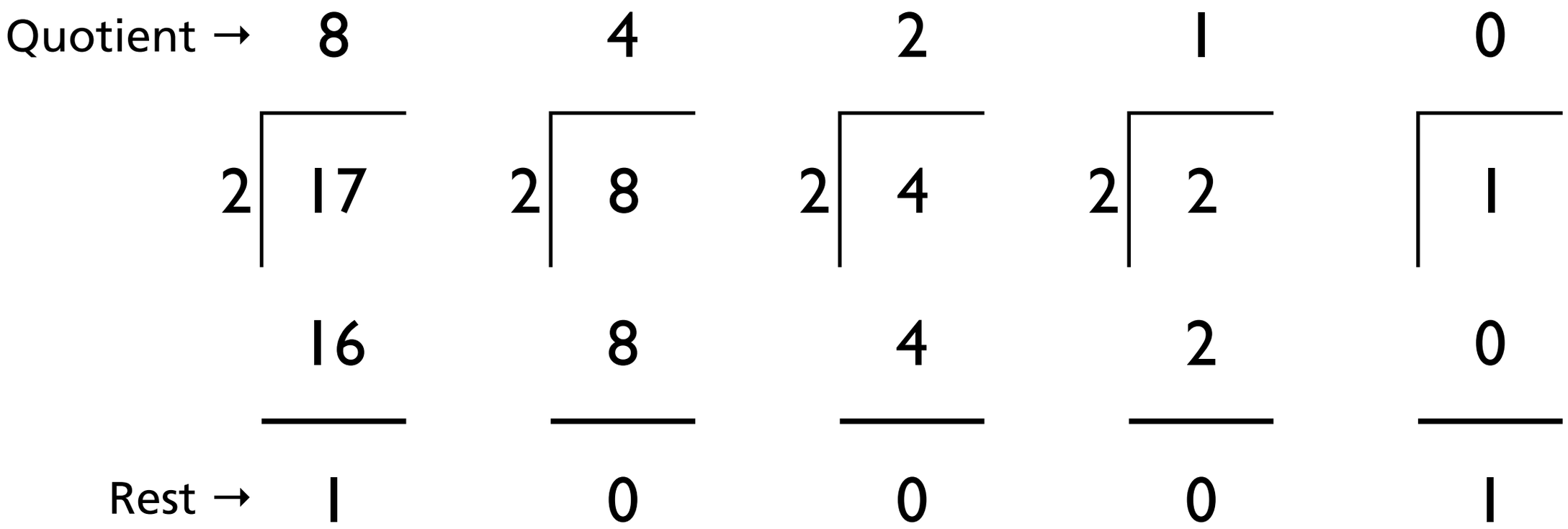
123 solange durch 2 teilen, bis der Quotient 0 ist. Jeweils den Rest, also 0 oder 1, notieren. Leserichtung beachten!

Aufgabe: Dezimal nach Binär

Aufgabe: 17_{10} nach Binär

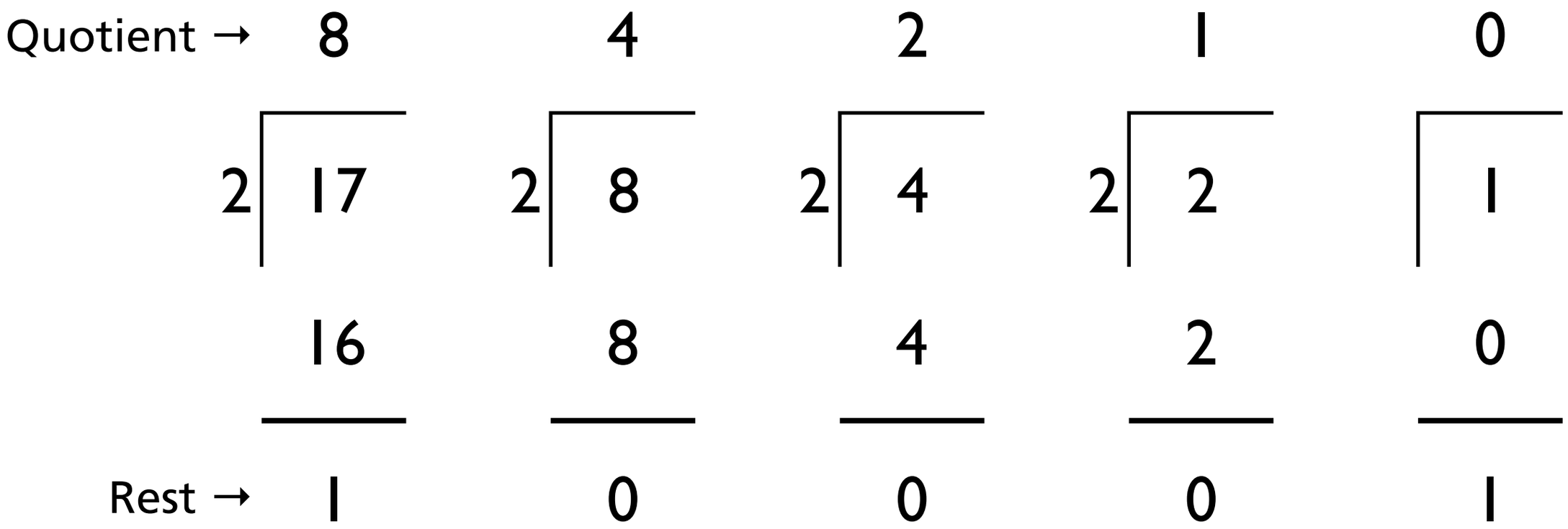
Aufgabe: Dezimal nach Binär

Aufgabe: 17_{10} nach Binär



Aufgabe: Dezimal nach Binär

Aufgabe: 17_{10} nach Binär



Resultat: 10001_2

Konversion: Binär nach Dezimal

Beispiel: 1111011_2

	1	1	1	1	0	1	1
Sum:	1×2^6	1×2^5	1×2^4	1×2^3	0×2^2	1×2^1	1×2^0
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:	1×2^4	0×2^3	1×2^2	0×2^1	1×2^0	
Sum:	16	0	4	0	1	→ 21

Aufgabe: Binär nach Dezimal

Aufgabe: 10101_2 nach Dezimal

	1	0	1	0	1	
Sum:	1×2^4	0×2^3	1×2^2	0×2^1	1×2^0	
Sum:	16	0	4	0	1	→ 21

Resultat: 21_{10}

Addieren von Binärzahlen

	I	I	I	I	0	I	I
+	I	0	I	I	0	I	I
	I		I		I		I	
	I	I	0	I	0	I	I	0

Jeweils gleiche Stellen (hinten beginnend) addieren. Für gerade Zahlen 1, für ungerade Zahlen 1, notieren und allfälligen Übertrag schreiben.

Obenstehendes Beispiel in Dezimalzahlen: 123 + 91 = 214

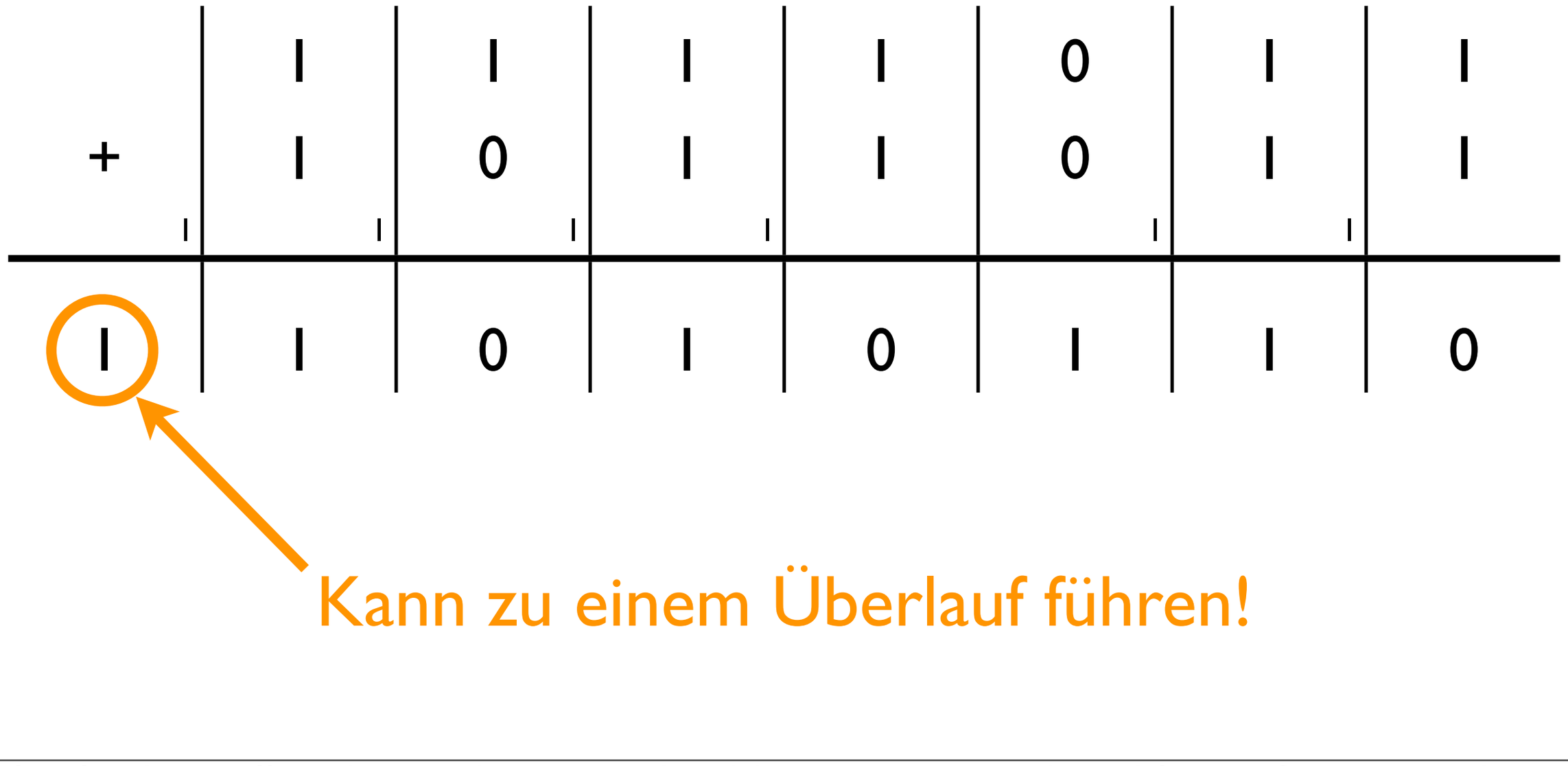
Zusatzfrage: Wie kann ich negative Zahlen im Binärsystem repräsentieren? Eine mögliche Antwort: Mit einem Vorzeichenbit. Vorderste Ziffer gibt an, ob die Zahl positiv (1) oder negativ (0) ist. Problem: Es kann ein Überlauf bei z.B. der Addition entstehen. Kann man ganz einfach z.B. in Groovy ausprobieren:

```
def a = Integer.MAX_VALUE
def b = a + 1

println a
println b
```

Ausgabe:
2147483647
-2147483648

Addieren von Binärzahlen



Jeweils gleiche Stellen (hinten beginnend) addieren. Für gerade Zahlen 1, für ungerade Zahlen 1, notieren und allfälligen Übertrag schreiben.

Obenstehendes Beispiel in Dezimalzahlen: 123 + 91 = 214

Zusatzfrage: Wie kann ich negative Zahlen im Binärsystem repräsentieren? Eine mögliche Antwort: Mit einem Vorzeichenbit. Vorderste Ziffer gibt an, ob die Zahl positiv (1) oder negativ (0) ist. Problem: Es kann ein Überlauf bei z.B. der Addition entstehen. Kann man ganz einfach z.B. in Groovy ausprobieren:

```
def a = Integer.MAX_VALUE
def b = a + 1

println a
println b
```

Ausgabe:
2147483647
-2147483648

Aufgabe: Addition

	1	0	1	1
+	1	0	1	1

Aufgabe: Addition

Aufgabe: Addition

		I	0	I	I
+		I	0	I	I
	I				
	I	0	I	I	0

Sekundärspeicher

Der Inhalt des Hauptspeichers geht verloren, wenn die Stromzufuhr des Computers unterbrochen wird. Er ist sog. volatil, dafür aber deutlich schneller als Sekundärspeicher. Sekundärspeichermedien umfassen Festplatten, USB-Sticks, CDs, Disketten, Bänder, etc.. Programme und Daten werden somit auf Sekundärspeichermedien dauerhaft gespeichert, aber jedes Mal in den schnellen Primärspeicher geladen, wenn sie verarbeiten werden sollen.

Software

Programme (oder Software) sind Instruktionen an den Computer, oder anders formuliert: Ihr teilt dem Computer mittels Programmen mit, was er für Euch erledigen soll.

Gängige Vorurteile/ Ausreden

“Computer sind intelligent.”

“Der Computer ist abgestürzt.”

“Der Computer erlaubt das nicht.”

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

Computer machen keine Fehler (in der Tat kann Hardware natürlich fehlerhaft sein, aber das ist sehr viel seltener als Programm-Fehler). Programme machen auch keine Fehler. Programmierer machen Fehler!

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

64-bit bedeutet: 64 Stellen im Binärsystem, um eine Gleitkommazahl darzustellen. Wenn man eine 64-bit Zahl in eine 32-bit Zahl konvertiert, dann werden einfach Stellen abgeschnitten (sprich eine ganz andere Zahl kommt heraus).

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.

1982 erwarben die Sowjets, im Rahmen ihrer Bestrebungen sensitive U.S. Technology zu stehlen oder verdeckt zu kaufen, ein Kanadisches Computersystem als Steuerung für eine trans-sibirische Naturgaspipeline. In dieses hatten Mitarbeiter der CIA vorher einen Fehler eingebaut, welcher – nach Ablauf einer bestimmten Zeit – zu einen viel zu hohen Druck in der Pipeline führen sollte. Die Explosion war sogar vom Weltraum aus zu beobachten.

1982 wurde Therac-25, ein Nachfolger der erfolgreichen Modelle Therac-6 und -20, gebaut. Der Beschleuniger konnte entweder durch eine zwischengeschaltete Wolfram-Platte Röntgenstrahlen oder direkt künstliche Beta-Strahlung zu Therapiezwecken einsetzen. Im direkten Modus wurde eine deutlich geringere Strahlenintensität verwendet. Im Gegensatz zu den Vorgängermodellen wurden die mechanischen Sicherheitssperren durch eine Softwarelösung ersetzt. Software schien viel verlässlicher, da sie keinen Abnützungserscheinungen unterliegt. Was die Ingenieure nicht wussten, war dass ein Programmierer ohne formale Ausbildung das zu Grunde liegende Betriebssystem zusammengeschustert hatte. Ein subtiler Fehler – eine sog. Race Condition – führte dazu, dass jemand, der schnell tippen konnte, den hoch-energetischen Modus aktivieren konnte, ohne dass die Wolfram-Platte bereits in Position war.

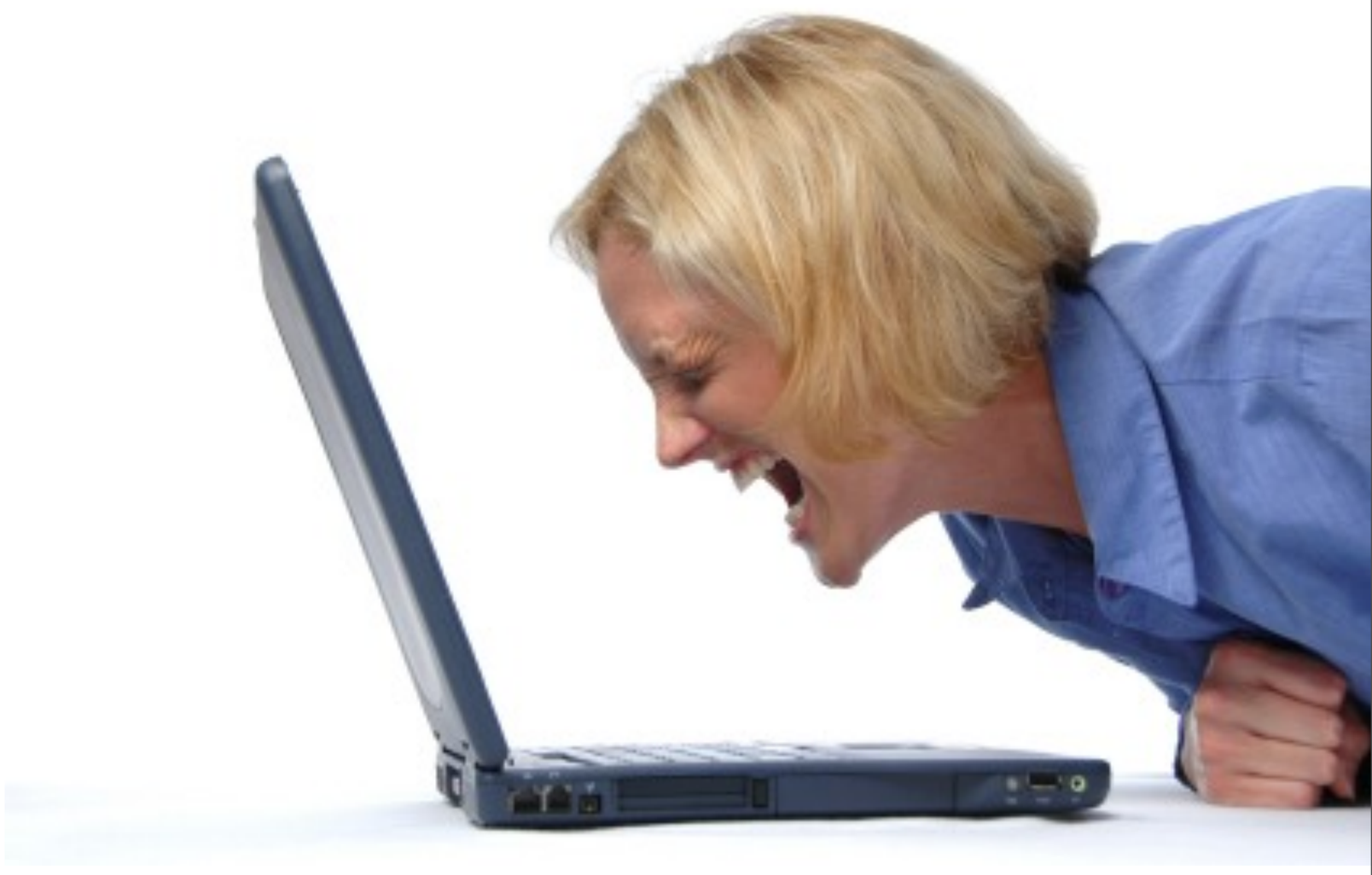
Software schreiben macht Spass

Ein Programmierer entwirft und baut eigene Maschinen

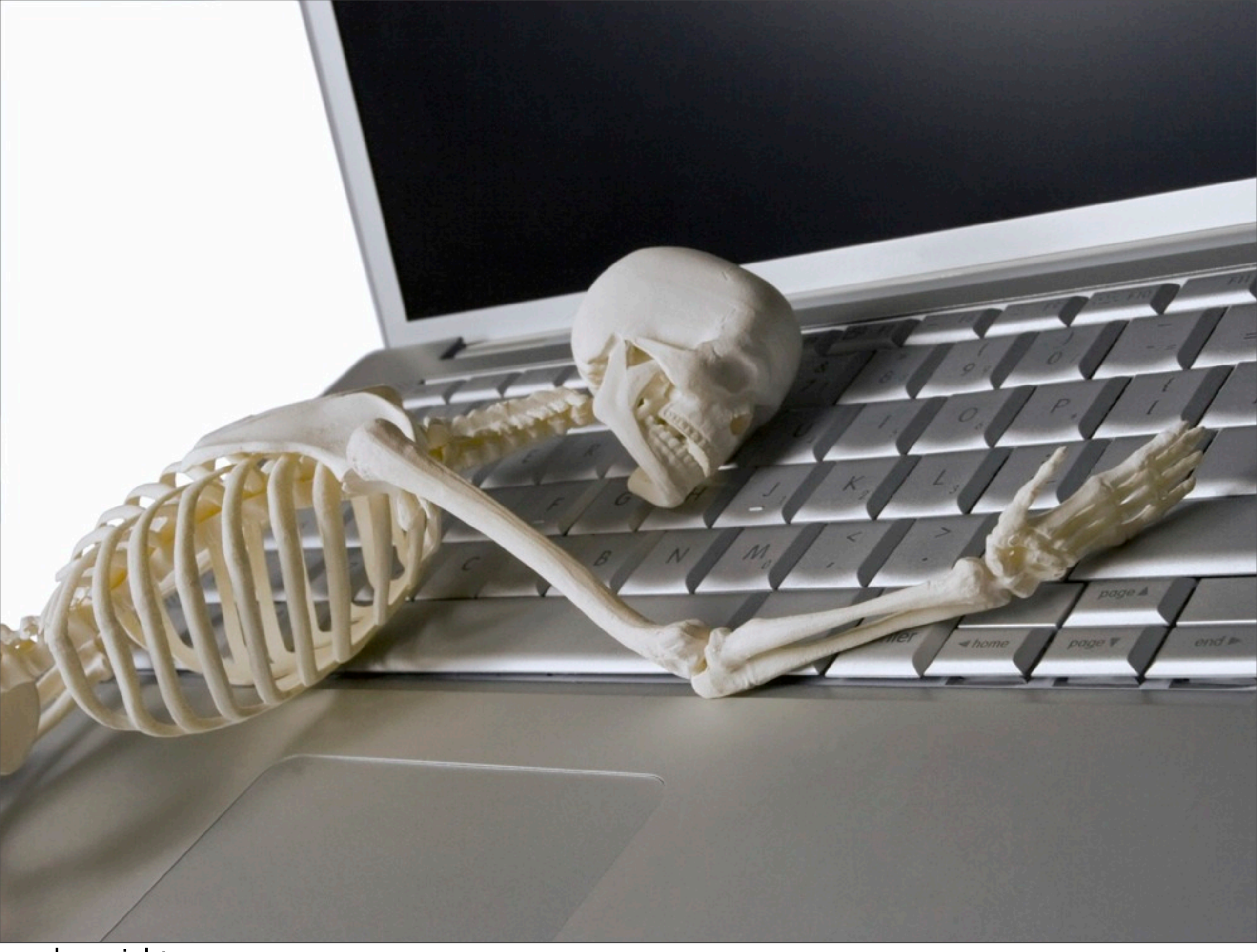
Er kann kreativ sein und seine Vorstellungskraft gebrauchen

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?



So...?



...eher nicht.

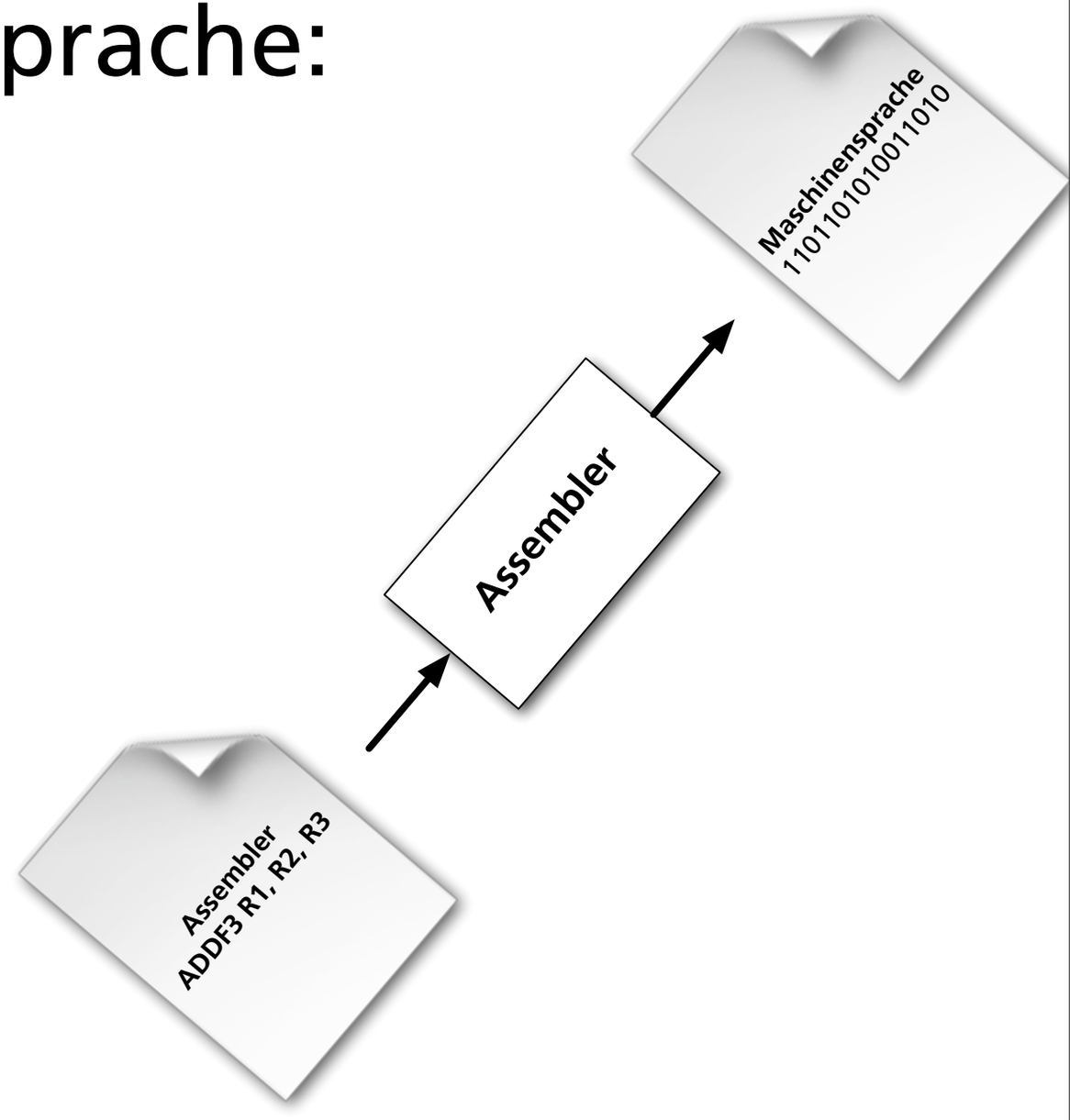
Maschinensprache vs. Assembler

Addieren in Maschinensprache:

```
1101101010011010
```

Addieren in Assembler:

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



Die Sprache in der Computer Instruktionen entgegennimmt, ist die sog. Maschinensprache – eine Anzahl primitiver Anweisungen, die in jeden Computer eingebaut sind. Die Anweisungen sind in Binärkode gehalten. Assembler ist eine sog. low-level Programmiersprache, die ins Leben gerufen wurde, um das Programmieren zu vereinfachen. Da Computer Assembler nicht direkt verstehen können, müssen die Assembler-Befehle vor der Ausführung noch in Maschinencode übersetzt werden.

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:

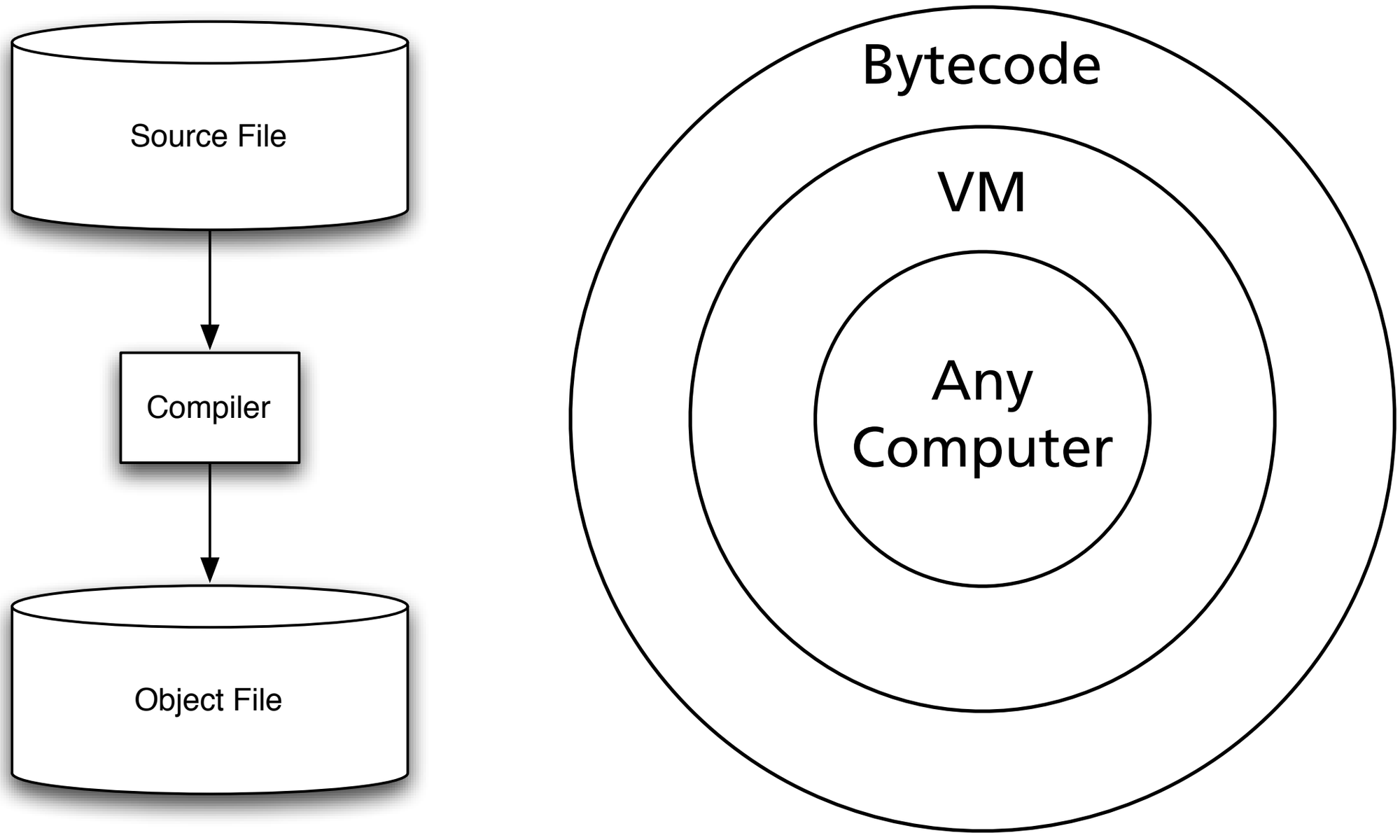
```
float area = 5 * 5 * 3.1415;
new Window(“This is the title”).setVisible(true);
```

Von Assembler wurde weiter abstrahiert, damit man Englischen Wörtern und Phrasen ähnliche Befehle verwenden kann, um ein Programm zu gestalten. Das Resultat waren Sprachen, wie Cobol, C++, etc.

Der Kompiliervorgang

Programmcode wird vom Compiler in Objektcode umgewandelt. Der Linker fügt dem Objektcode weitere Funktionalität z.B. aus sogenannten Klassenbibliotheken hinzu (z.B. DLLs in C++/Windows: Dynamic-link library).

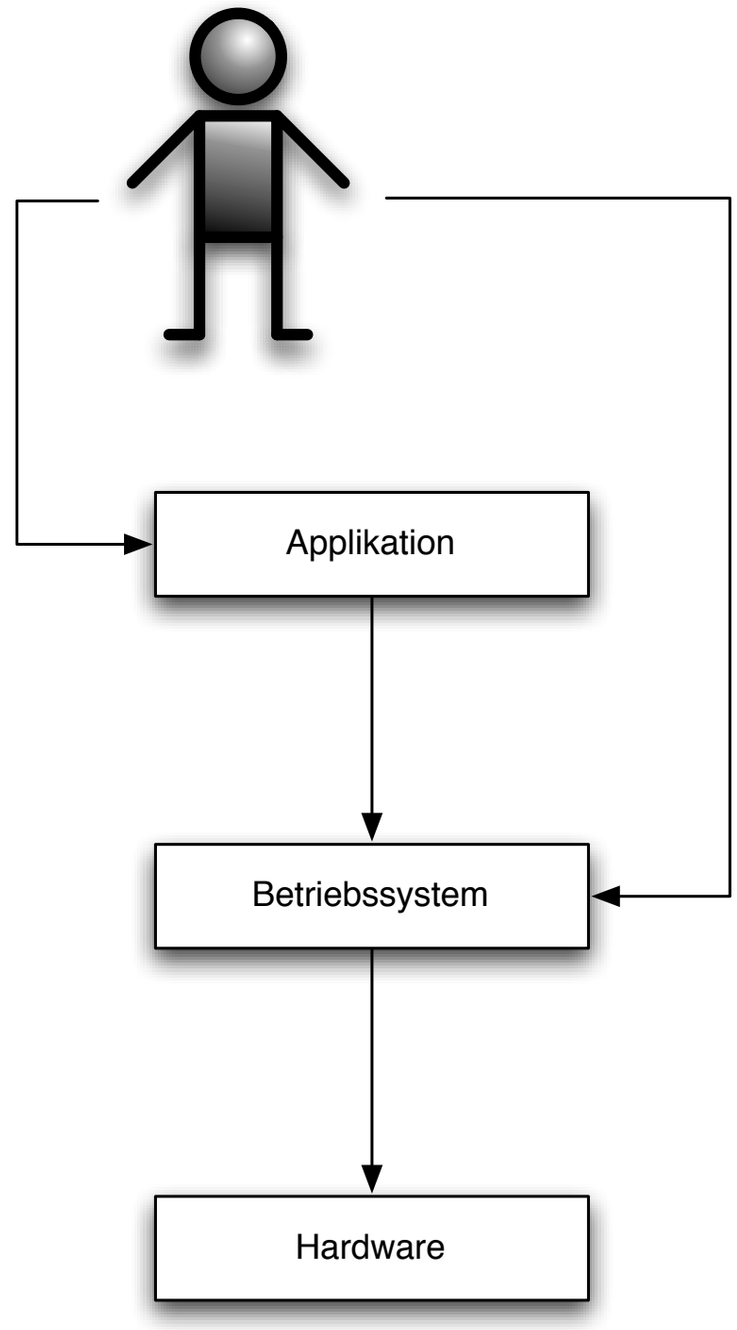
Virtuelle Maschinen



‘write once - run everywhere‘-Idee. Die Funktionalität der Klassenbibliothek wird durch die VM bereitgestellt. Der Linker-Schritt entfällt somit. Der VM Hersteller “versteckt” die Spezialitäten eines jeden Computers (Linux, OSX, Win, Nokia OS,...) hinter der VM. Die Bytecode-Programme “sehen” in der VM immer den gleichen Typ Computer.

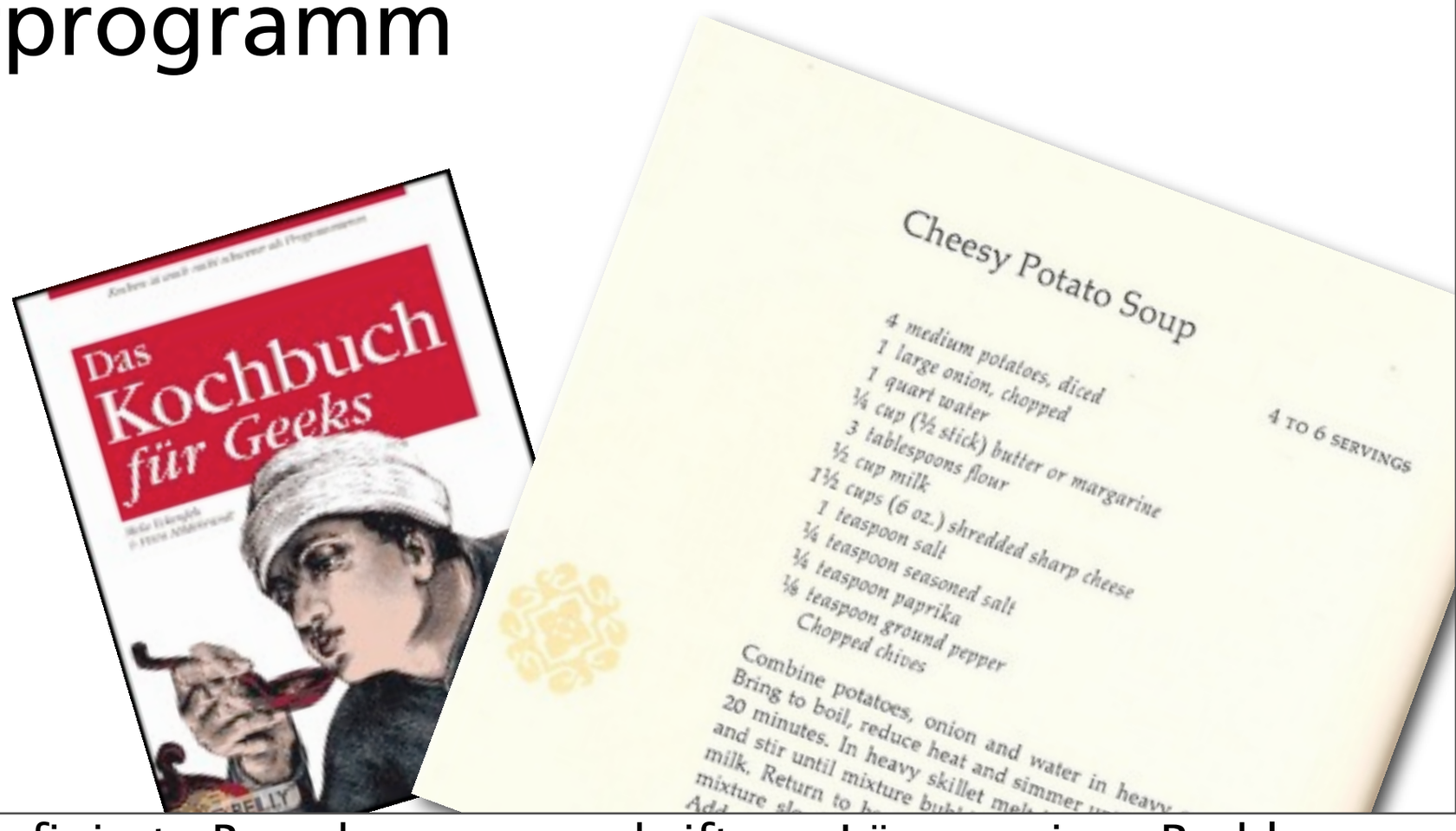
Das Betriebssystem

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



Algorithmisches Denken

Vom Rezept zum ausführbaren Computerprogramm



Ein Algorithmus ist eine genau definierte Berechnungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen und wird durch eine endliche Menge von Regeln definiert, die nacheinander angewendet und oft nach bestimmten Bedingungen wiederholt werden.

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 beigeben 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 beigeben.

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

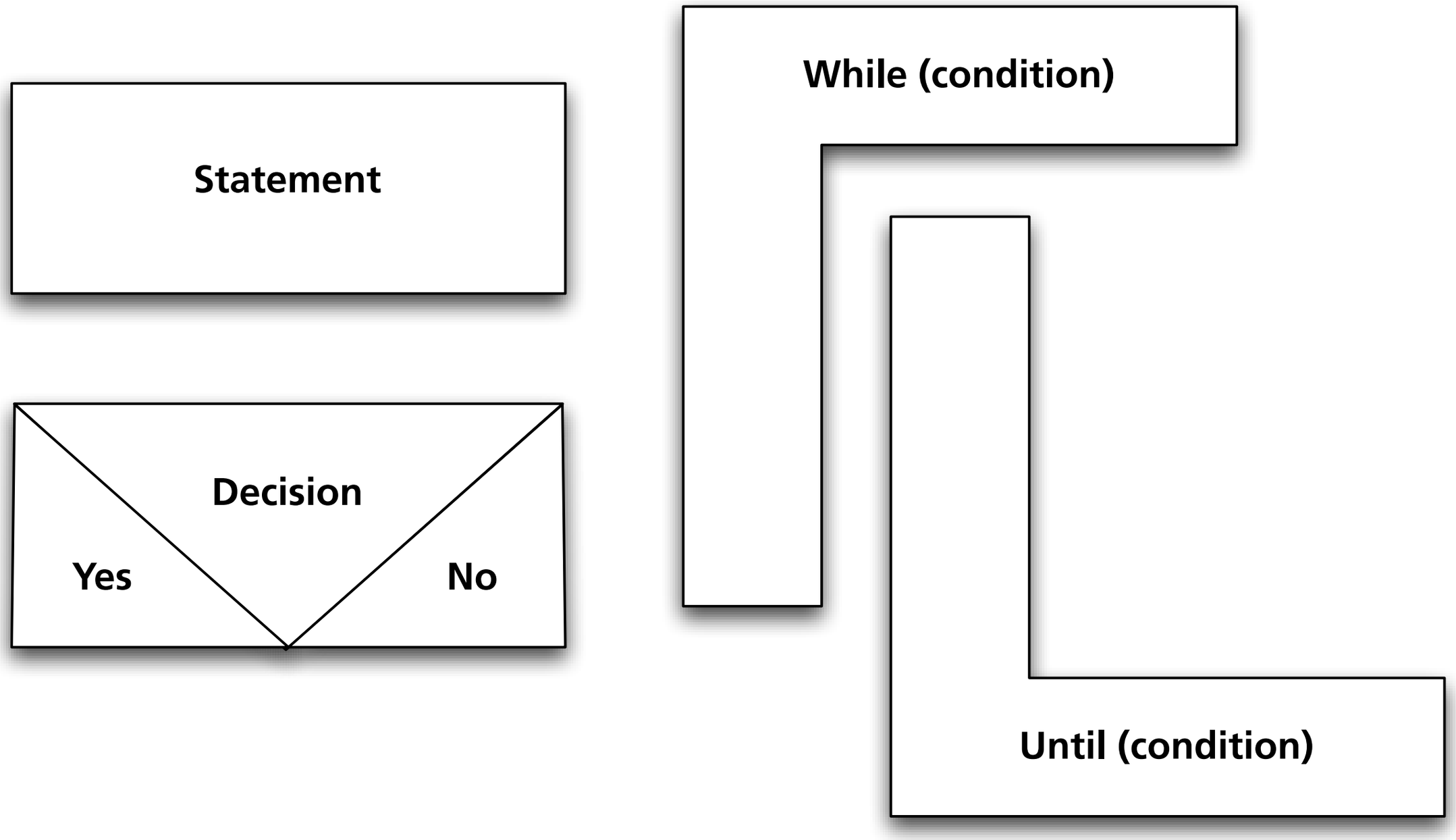
Der Euklidische Algorithmus (um 300 v. Chr. beschrieben dient zur Ermittlung des grössten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen A und B.

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



Graphische Notation zur Beschreibung von Algorithmen. Statements sind einzelne, atomare (nicht zerlegbare) Anweisungen an den Computer. Decisions (Verzweigungen) bestehen aus einer Condition (Bedingungen) und einem then ([wenn ...] dann ...) und else (...sonst) Teil. Loops (Schleifen) wiederholen bestimmte Statements oder Decisions. Decisions und Loops kann man in einander verschachteln.

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 beigeben und 3-4 Minuten unter Rühren glasig werden lassen.

3. Mit dem Wein ablöschen und solange köcheln lassen, bis die Flüssigkeit verdunstet ist.

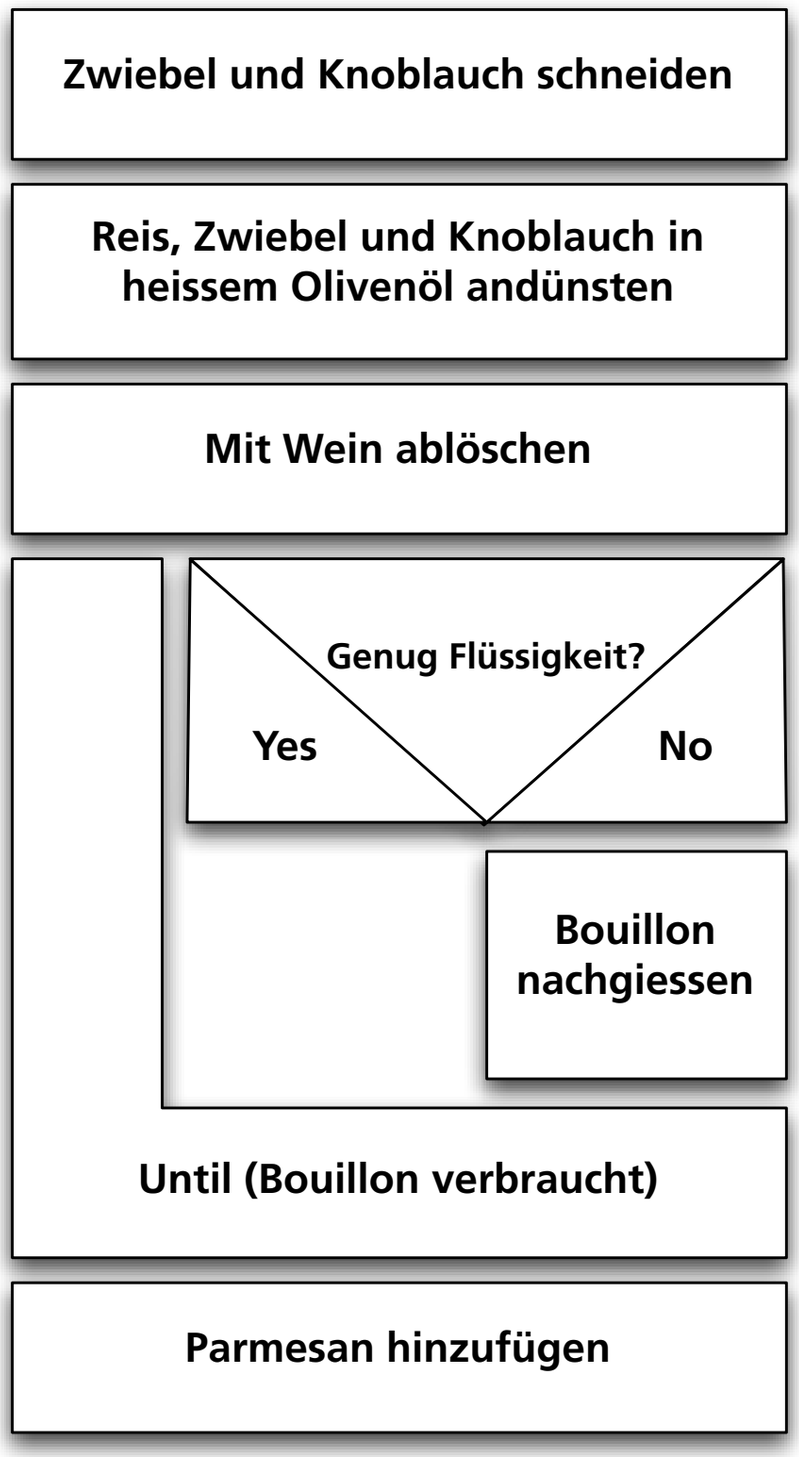
4. Den Safran beigeben.

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

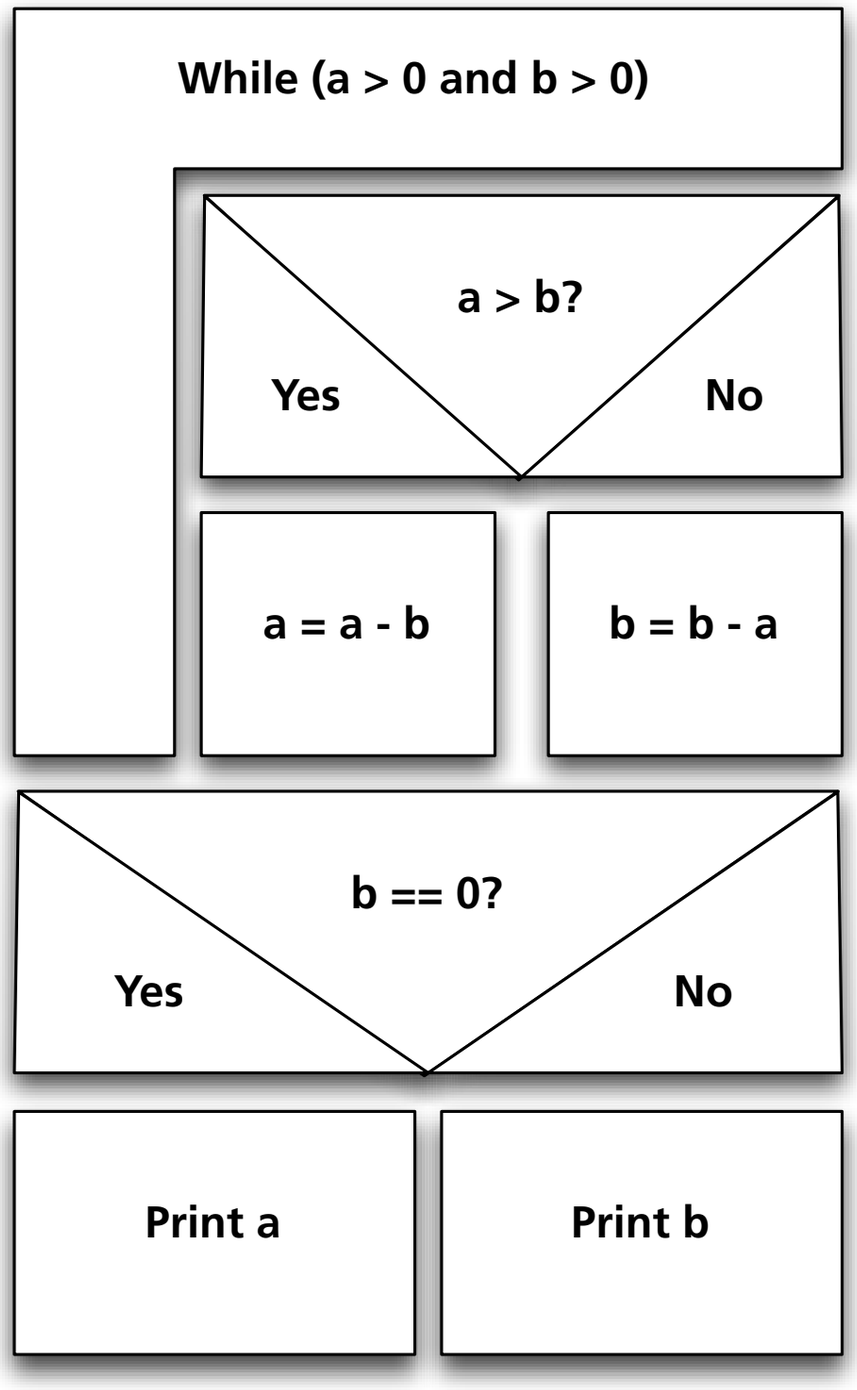


Aufgabe: Der Euklidische Algorithmus als NSD

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

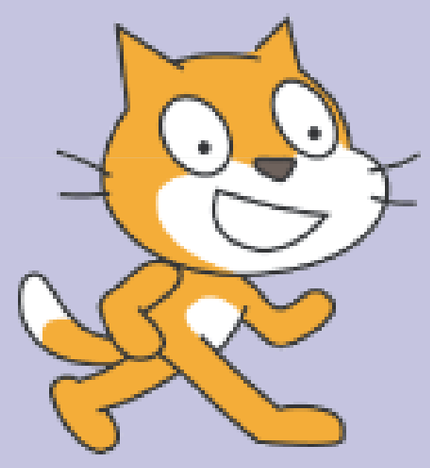
Lösung: Der Euklidische Algorithmus als NSD



Getting Started with

SCRATCH

version 1.4



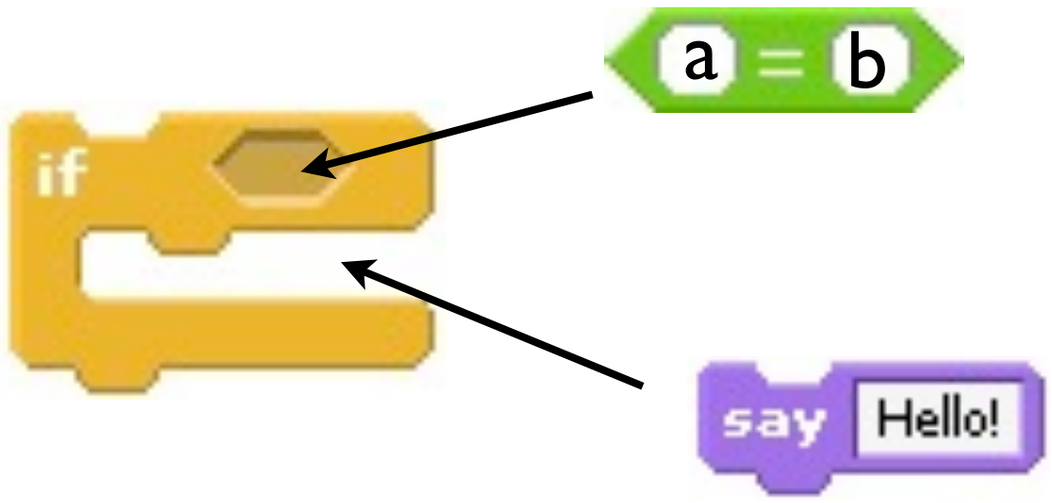
http://scratch.mit.edu

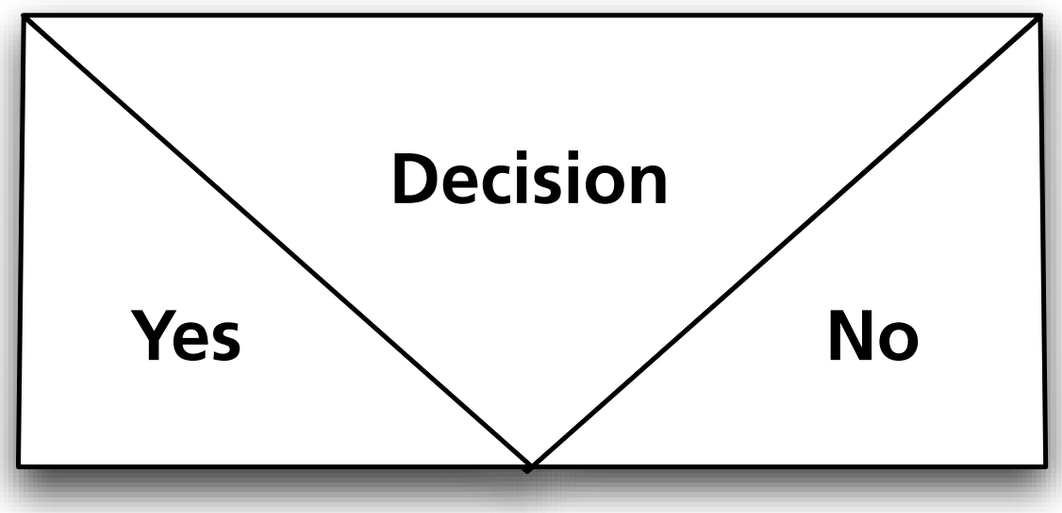
http://info.scratch.mit.edu/Support/Reference_Guide_1.4

Algorithmen in Scratch: Statements

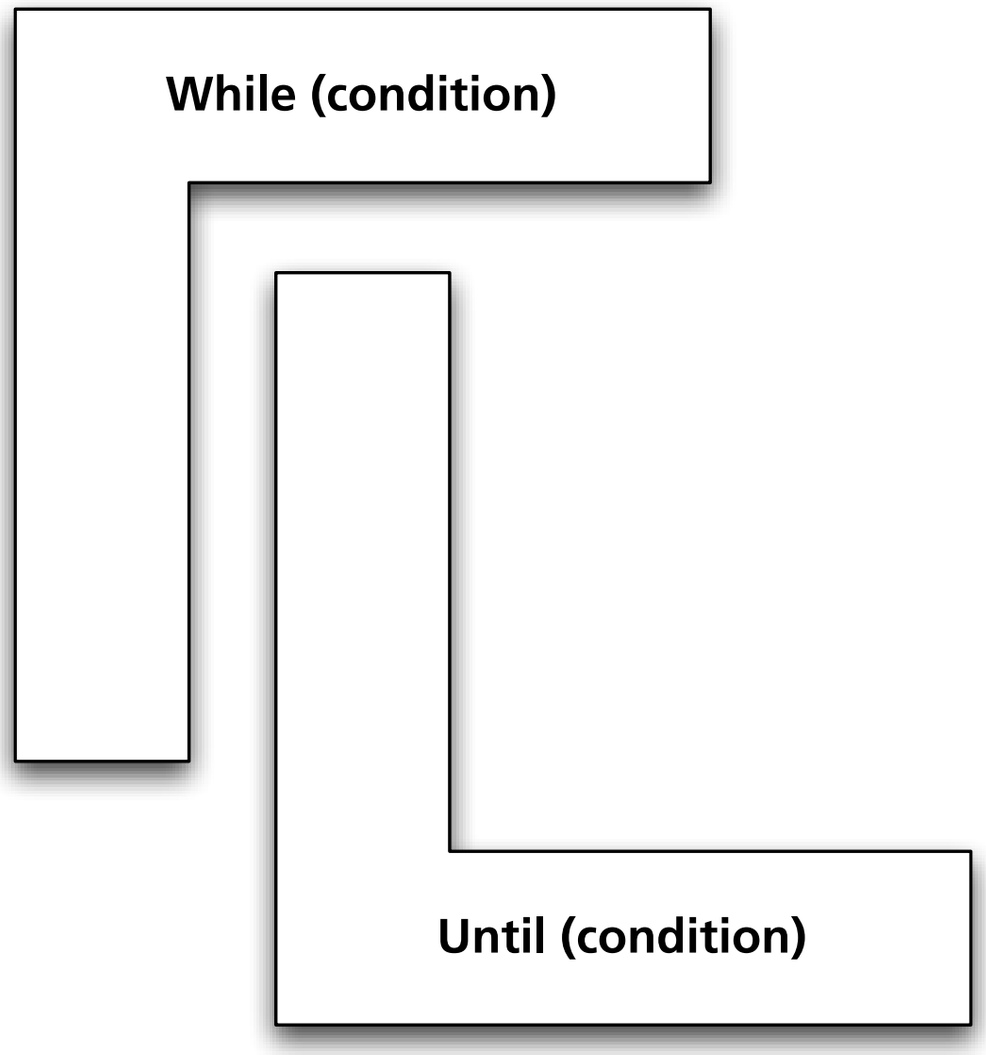
Statement

Algorithmen in Scratch: Decisions

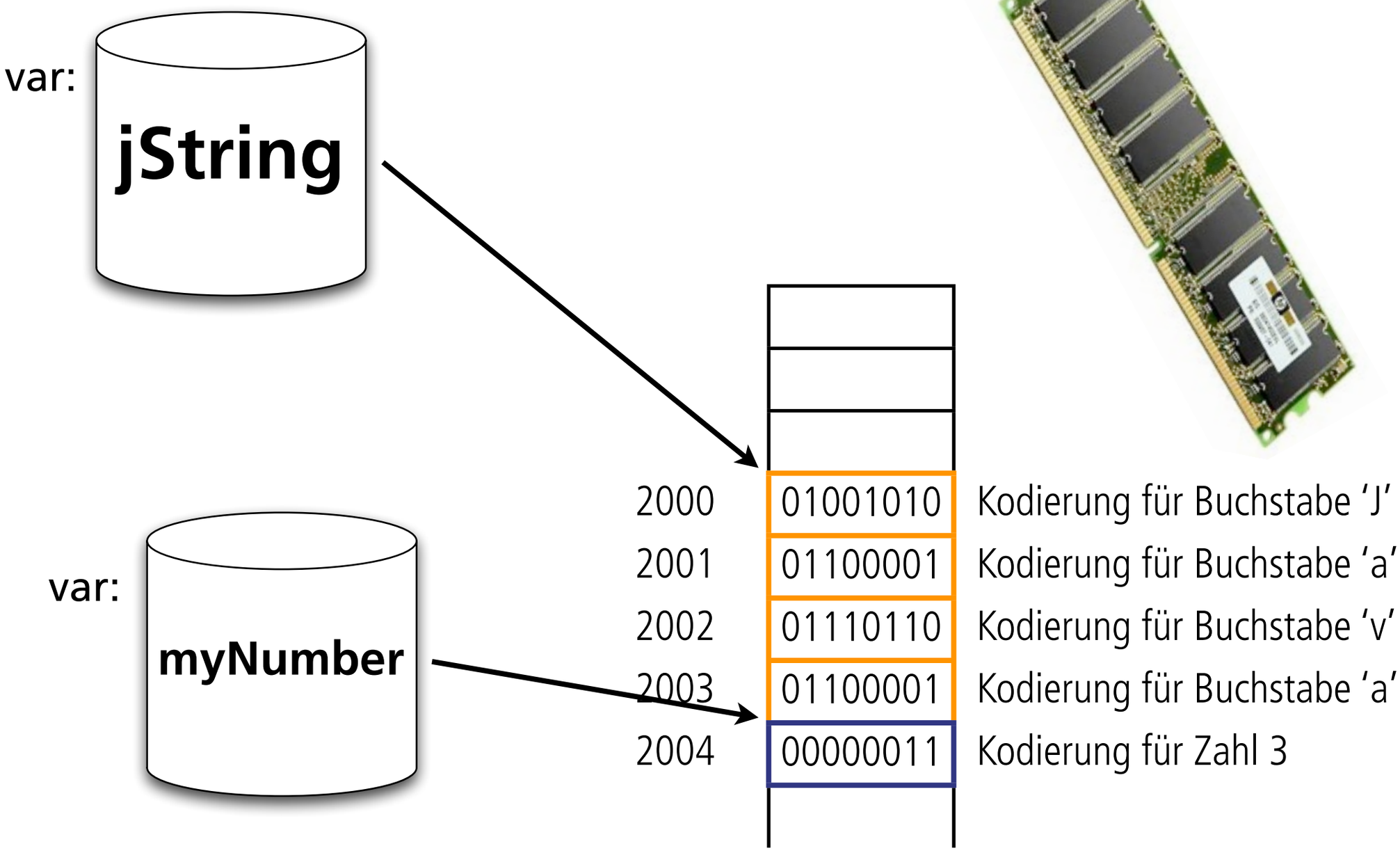




Algorithmen in Scratch: Loops

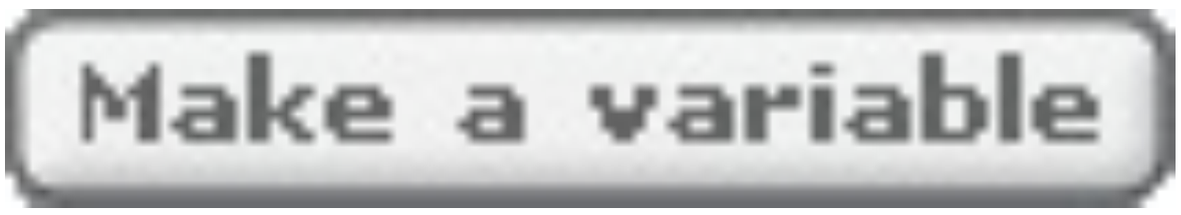


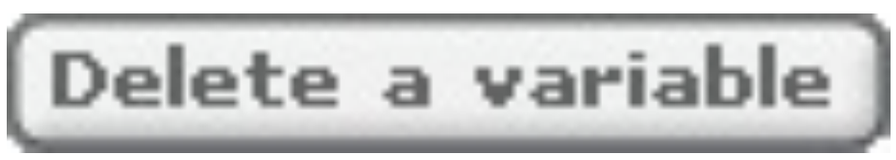
Variablen in einem Programm



Variablen sind Behälter (oder Wegweiser?) zu bestimmten Stellen im Speicher, an denen Informationen gespeichert sind, die man gerne wiederverwenden möchte.

Variablen in Scratch





SCRATCH 1.4 INTERFACE

SHARE

SPRITE ROTATION STYLE

CURRENT SPRITE INFO

TABS
Edit scripts, costumes, or sounds.

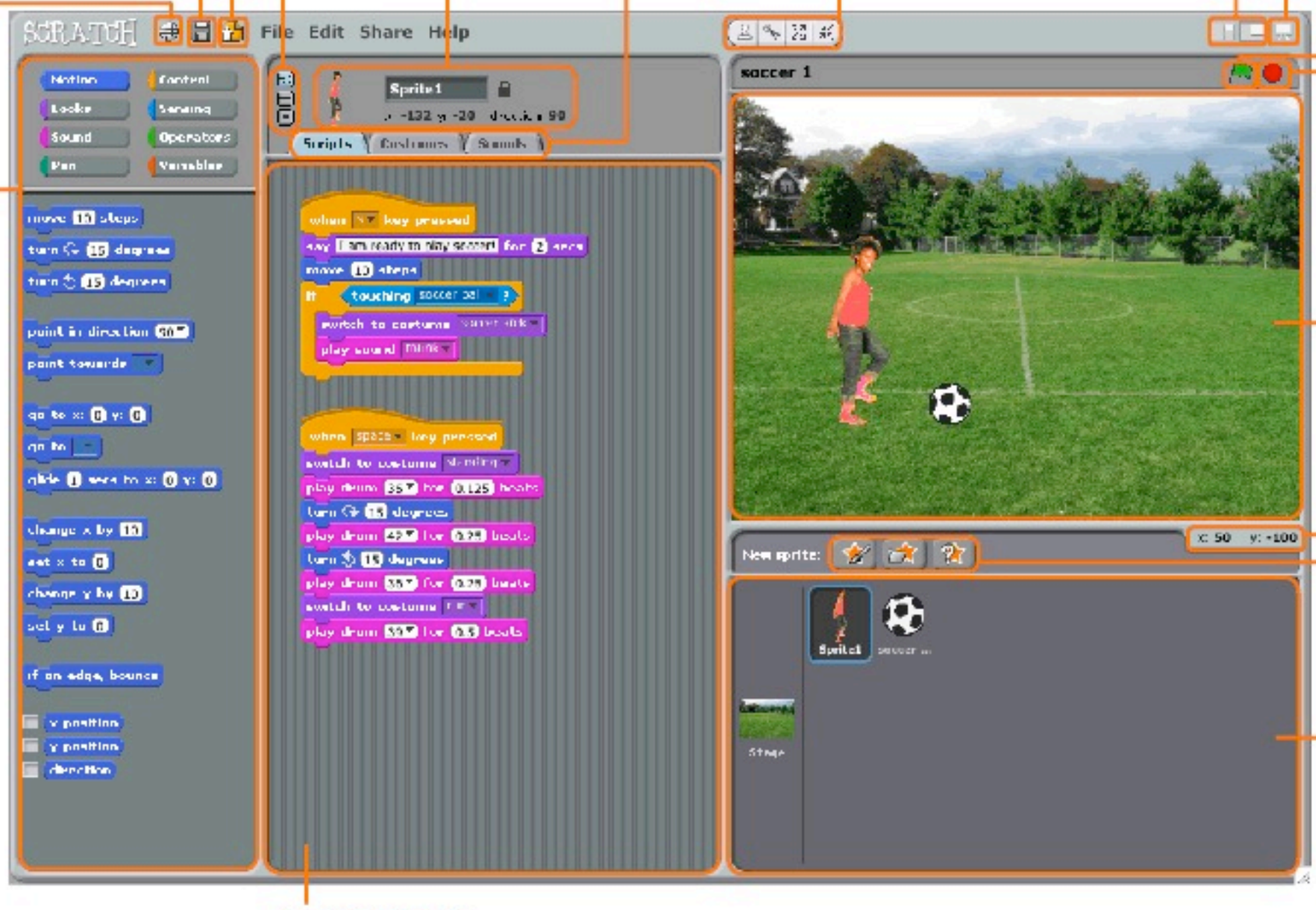
TOOLBAR

VIEW MODE
Change to large or small stage view.

PRESENTATION MODE
Present your project.

SAVE

LANGUAGE



BLOCKS PALETTE
Blocks for programming your sprites.

GREEN FLAG
A way to start scripts.

STOP SIGN
Stops all scripts.

STAGE
Where your Scratch creations come to life.

MOUSE X-Y DISPLAY
Shows location of cursor.

NEW SPRITE BUTTONS
Create a new character or object for your project.

SPRITE LIST
Thumbnails of all your sprites. Click to select and edit a sprite.

SCRIPTS AREA
Drag blocks in, snap them together into scripts.