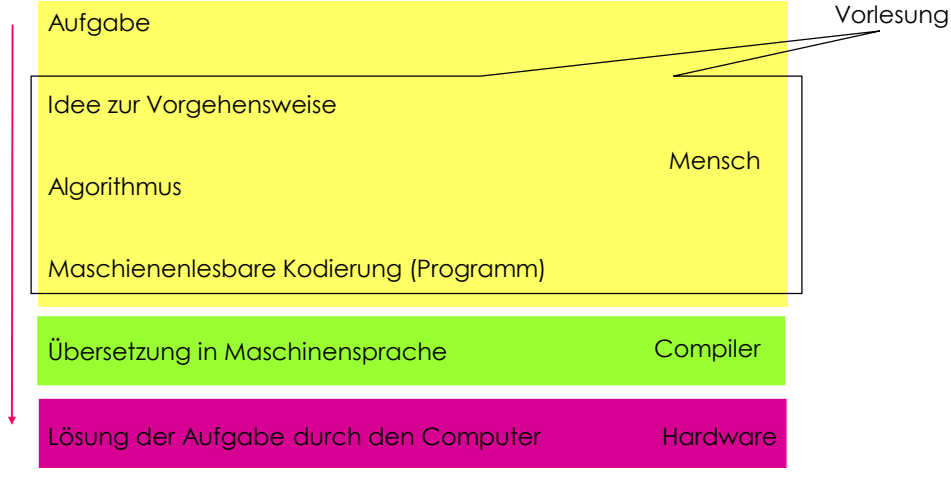
Essentielle Grundlagen der Informatik

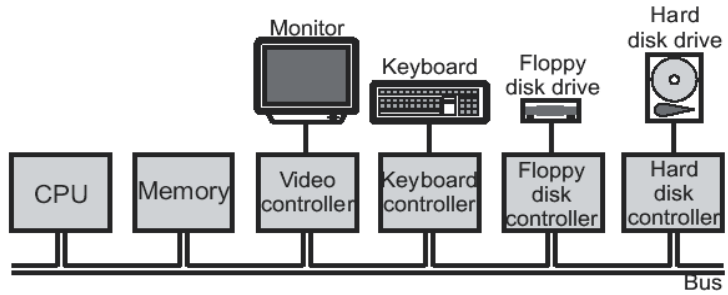
Was heisst Programmieren?

Exaktes Instruieren eines Computers, eine bestimmte Aufgabe zu lösen.



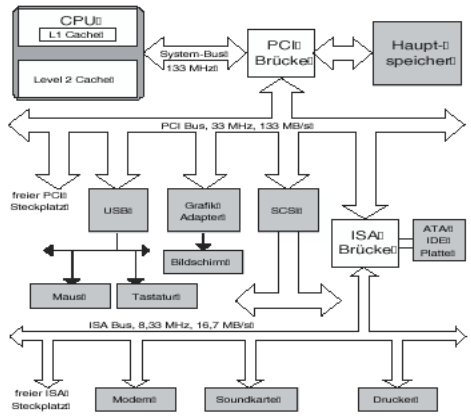
Organisation der Hardware

Architektur eines einfachen Computersystems mit Bus



System-Architektur der Hardware

Architektur eines PC Systems mit mehreren Bussen an Brücken



Software wird unterschieden in

Anwendersoftware erlaubt die Lösung allgemeinster Aufgabenstellungen.

- z. B. Textverarbeitung, Tabellenkalkulation, Bildbearbeitung, Buchhaltung, Produktionsplanung, Lohn und Gehaltsabrechnung, Spiele...

Systemsoftware hilft beim Betrieb des Rechners und bei der Konstruktion der Anwendersoftware.

- Systemsoftware umfasst neben Datenbanksystemen, Übersetzern (compiler) etc. in jedem Fall das Betriebssystem.

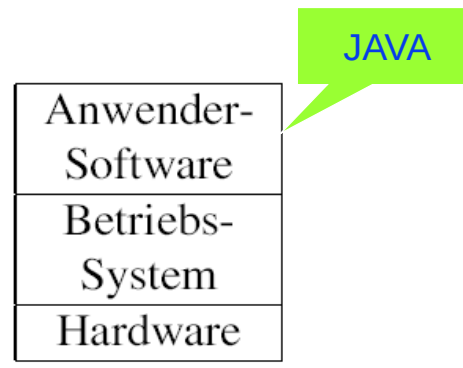
Betriebssystem

Das **Betriebssystem** (operating system) isoliert die Anwendersoftware von der Hardware: das Betriebssystem kommuniziert mit der Hardware und die Anwendersoftware auf dem Betriebssystem.

Das Betriebssystem verwaltet die Ressourcen der Hardware (wie z. B. Geräte, Speicher und Rechenzeit) und es stellt der Anwendersoftware eine abstrakte Schnittstelle (die Systemaufrufschnittstelle) zu deren Nutzung zur Verfügung.

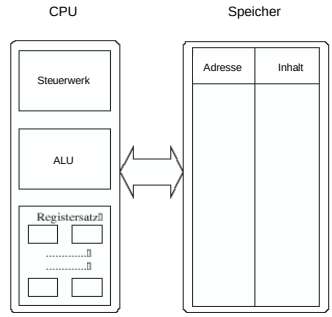
Dadurch vereinfacht es die Nutzung der Ressourcen und schützt vor Fehlbedienungen.

- Betriebssysteme, die es mit diesem Schutz nicht so genau nehmen, führen zu häufigen **Systemabstürzen** (system crash).



Kern aller heutigen Computer

Grundsätzlicher Aufbau verschiedener Rechnersysteme ist ähnlich:



Wahlfreier Zugriff

Von Neumann Architektur

Speicher

Kleinste Speichereinheit hat 2 Zustände

- 1 Bit
- Zustände werden i.A. mit 0 und 1 bezeichnet

Mit 2 Speichereinheiten 2^2=4 Zustände darstellbar

Bit 1	Bit 0	
0	0	Zustand 0
0	1	Zustand 1
1	0	Zustand 2
1	1	Zustand 3

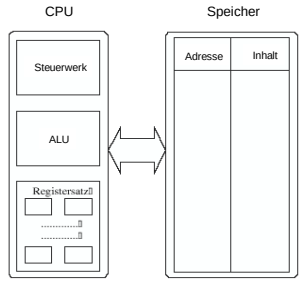
Mit 8 Bit 2^8=256 Zustände darstellbar

- 8 Bit = 1 Byte
- Heutzutage sind Bytes die kleinsten *adressierbaren* Speichereinheiten
  - Kleinere Einheiten müssen aus einem Byte extrahiert werden

z.B. 0010 0101

Speicher & Adressierung

Von Neumann Architektur



Adresse	Inhalt (byte)
1.	0101 0010
2.	1100 1100
3.	1001 1000
4.	0000 0100
5.	1111 1001
6.	0010 1011
7.	.....
..	

Wieviel Speicheradressen hat ein Computer?

Maximal soviel der Datenbus codieren kann.

- PC's mit 32 Bit Datenbus
  $2^{32} = 4 * 2^{30} = 4 * 2^{10} * 2^{10} * 2^{10} = 4$ Giga Zustände (Adressen)
- bei 64 Bit Datenbus $2^{64} = 2^{34}$ Giga $\approx 10^{11}$ Giga Zustände (Adressen)

Daten und Befehle

Daten — Menge addressierbarer Speicherzellen

Daten sind binär gespeichert (z.B. 17 = 10001)
Binärspeicherung ist universel (Zahlen, Texte,Filme, Ton …)
1 Byte := 8 Bit

Befehle — Operationen mit den Speicherzellen

Maschinensprache		Hochsprache
ACC ← x	// Lade Zelle x	z = x + y
ACC ← ACC + y	// Addiere y	
z ← ACC	// Speichere Ergebnis in Zelle z	

Variablen und der Typ von Variablen

Menschen benennen Dinge gerne mit Namen statt mit numerischen Adressen, so kennt jede Programmiersprache das Konzept einer **Variablen** als abstraktes Analogon zu einer Speicherstelle.

Eine Variable hat einen symbolischen **Namen**,

- hinter dem eine **Adresse** verborgen ist,
- und der **Wert** (value) der Variable ist der Wert des dort gespeicherten Bitmusters.
  - Um diesen erschließen zu können, hat die Variable einen **Typ** (type), der bei ihrer Vereinbarung angegeben werden muss.

Binärcodierung elementarer Datentypen

Unterscheide

- Zahl-Wert
- Zahl-Bezeichner

Zu ein- und demselben Zahl-Wert kann es verschiedene Bezeichner geben, z. B.

- Fünf, 5, V, 101

Da es unendlich viele Zahl-Werte gibt, ist es sinnvoll, sich eine Systematik zur Erzeugung von eindeutigen Bezeichnern zu schaffen

- Die auch das Rechnen mit Zahlen unterstützt
  - Verwendung von römischen Zahlen bietet keine gute Unterstützung

Binärcodierung elementarer Datentypen

Ein Zahlsystem (number system) besteht aus

- endlich vielen Ziffern (digits) und
- einer Vorschrift,
  - wie Zeichenreihen, die aus diesen Ziffern gebildet wurden, als Zahl-Werte zu interpretieren sind

Arabische Zahlsysteme zur Basis β

- Natürliche Zahl z wird geschrieben als Polynom

$$z = \sum_{i=0}^{n-1} z_i \beta^i$$

  - Dabei $0 \leq z_i < \beta$

Datentypen & Variable

Adresse	Inhalt (byte)
1. 0000 0000 0000 0000 … 0000 0000 0000 0000	0101 0010
2. 0000 0000 0000 0000 … 0000 0000 0000 00001	1100 1100
3. 0000 0000 0000 0000 … 0000 0000 0000 00010	1001 1000
4. 0000 0000 0000 0000 … 0000 0000 0000 00011	0000 0100
5. 0000 0000 0000 0000 … 0000 0000 0000 00100	1111 1001
6. 0000 0000 0000 0000 … 0000 0000 0000 00101	0010 1011
7. 0000 0000 0000 0000 … 0000 0000 0000 00111	.....
..	

Wie ist der Inhalt des Speichers zu interpretieren?

Dem Bitmuster ist seine Bedeutung (Text, Zahlen oder Musik) nicht fest zugeordnet.

Je nach Variablendefinition müssen unterschiedlich viele Bytes interpretiert werden und gleiche Bitmuster können unterschiedliche Bedeutungen haben!

Variablen haben immer einen Datentypen

Elementare Datentypen in Java

byte	8 Bit Zahl $-2^7 \ldots 2^7-1$ (-128, ... , 127)
short	16 Bit-Zahl $-2^{15} \ldots 2^{15}-1$ (-32768, ... , 32767)
int	32 Bit-Zahl $-2^{31} \ldots 2^{31}-1$ (-2 147 483 648, ... , 2 147 483 647)
long	64 Bit-Zahl $-2^{63} \ldots 2^{63}-1$
float	32 Bit IEEE-754-1985 Gleitkommazahl
double	64 Bit IEEE-754-1985 Gleitkommazahl
char	16 Bit Unicode
boolean	Wahrheitswert, *false* oder *true*

Gleitkommazahlen engl. „Floating-Point“

Darstellung einer Floating-Point-Zahl: $z = (-1)^v \cdot \text{Mantisse} \cdot 2^{\text{Exponent}}$

Floating-Point-Zahlen nach IEEE 754-1985

- 32 Bit `float`
- 64 Bit `double`

float	v	30 - Exponent - 23	22 - Mantisse - 0
	1 Bit	8 Bit	23 Bit
double	v	62 - Exponent - 52	51 - Mantisse - 0
	1 Bit	11 Bit	52 Bit

Algorithmus

Schrittweises, präzises Verfahren zur Lösung eines Problems

Problem: Summiere die Zahlen von 1 bis *max*. $sum = \sum_{i=1}^{i=max} i$

Name — Parameter

SummiereZahlenVon1bis*max* (↓ *max*, ↑*sum*)

1. *sum* ← 0
2. *zahl* ← 1
3. Wiederhole, solange *zahl* ≤ *max* — Folge
   - 3.1 *sum* ← *sum* + *zahl*
   - 3.2 *zahl* ← *zahl* + 1

Programm = Beschreibung eines Algorithmus in einer Programmiersprache

Variablen

Sind benannte Behälter für Werte

x	y
99	3

Können ihren Wert ändern

x ← x + 1

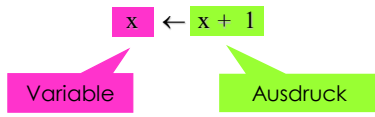
x
100

Haben einen Datentyp = Menge erlaubter Werte

Variablentyp	Werte	
Zahl	17, 54, ...	- in eine Zahlenvariable passen nur Zahlen
Zeichen	'a', 'x', ...	- in eine Zeichenvariable passen nur Zeichen

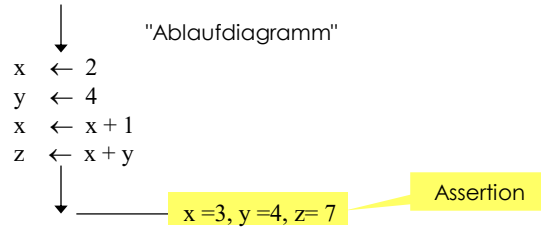
Anweisungen

Wertzuweisung



1. Werte Ausdruck aus
2. Weise seinen Wert der Variablen zu:

Anweisungsfolge (auch Sequenz)

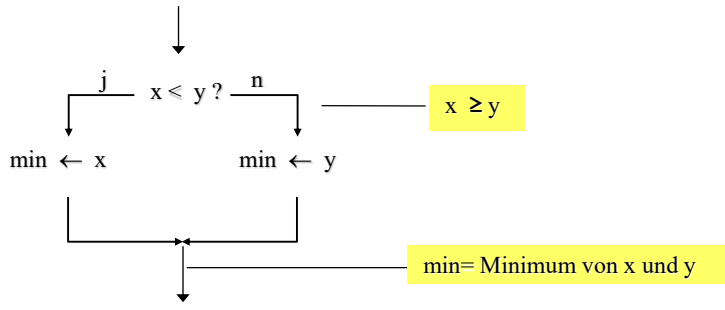


Assertion (Zusicherung)
Aussage über den Zustand des Algorithmus an einer bestimmten Stelle

Anweisungen

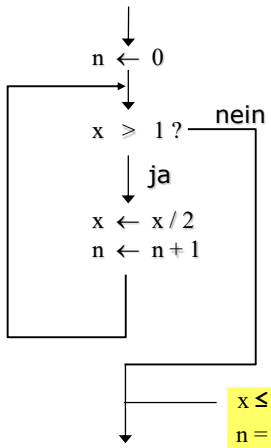
Auswahl (auch Verzweigung, Abfrage, Selektion)

Beispiel: Suche das Minimum der zwei Zahlen x und y.

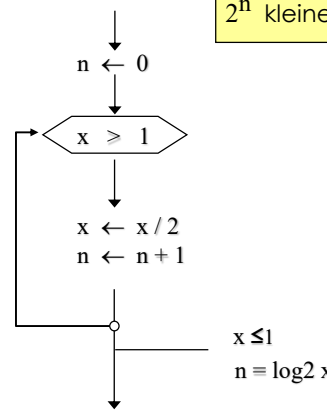


Anweisungen

Wiederholung (auch Schleife, Iteration)

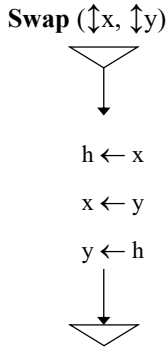


Beispiel: Suche die grösste ganze Zahl n mit 2^n kleiner oder gleich x .



Alternative Darstellung

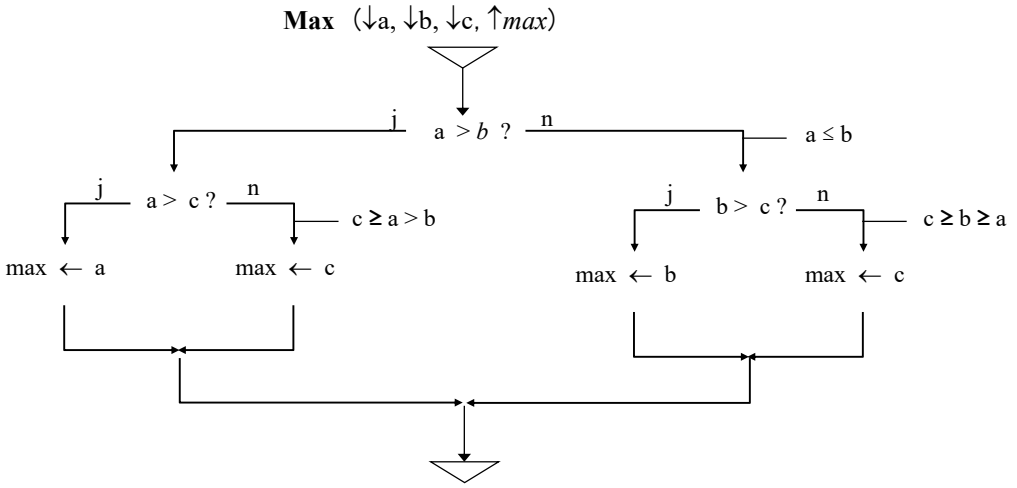
Beispiel: Vertausche zwei Variableninhalte



Schreibtischtest

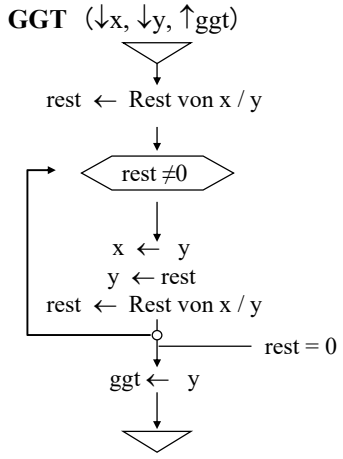
x	y	h
~~3~~	~~2~~	3
2	3	

Beispiel: Bestimme Maximum dreier Zahlen



Beispiel: Euklidscher Algorithmus

Berechnet den größten gemeinsamen Teiler zweier Zahlen x und y



Schreibtischtest

x	y	rest
~~28~~	~~20~~	8
~~20~~	~~8~~	4
8	4	0

Warum funktioniert dieser Algorithmus?

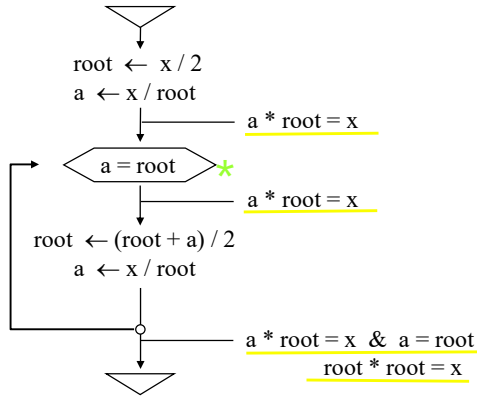
(*ggt* teilt x) & (*ggt* teilt y)
ggt teilt (x - y)
ggt teilt (x - q* y)
ggt teilt rest
ggt(x, y) = *ggt*(y, rest)

Beispiel: Berechne Quadratwurzel von x

1. Näherung:
root ← x / 2
a ← x / root

2. Näherung:
root ← (root + a) / 2
a ← x / root

SquareRoot (↓x, ↑root)



Schreibtischtest

x	root	a
10	5	2
	3,5	2,85471
	3.17857	3,14607
	3.16232	3,16223
	3.16228	3,16228

Kommazahlen sind meist nicht exakt gleich, daher besser

| a- root | < 0.0000001