

IT-Infrastruktur

SS 2015

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz A:

v1.1, 2015/03/05

- Einführung
- Seminararbeiten, Wissenschaftliches Arbeiten
- Datenformate und Wandlung

Service / Web-Seite: <http://fom.hgesser.de>

- Folien und Praktikumsaufgaben
- Vorlesungs-Videos („test, test“)
- Probeklausur gegen Semesterende
- Folien auch im Online-Campus

Prüfung und Benotung:

1. Lernfortschrittskontrolle (LFK)
2. Seminausarbeitung
3. Klausur über 120 Minuten

Fragen: direkt in der Vorlesung oder danach
oder per E-Mail: h.g.esser@gmx.de

10 ECTS-Punkte – großes Modul

- Präsenzstudium: 48 Stunden *) (= 64 UE)
- Eigenstudium: 172 Stunden
- Lernfortschr.-Kontr.: 30 Stunden
- Workload: 250 Stunden

Klausur-relevant:

- Vorlesungen
- Übungsaufgaben
- Seminarvorträge

*) Stunde
= Zeitstunde,
= 60 min.

Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1997)
Dipl.-Inform. (RWTH Aachen, 2005)
Fachjournalist (FJS Berlin, 2006)
- Chefredakteur Linux-Zeitschrift (seit 2000)
und Autor diverser Computerbücher
- seit 2006 Dozent (u. a. FOM, HS München, TH Nürnberg,
Univ. Erlangen-Nürnberg): Betriebssysteme, Rechner-
architektur, IT-Infrastruktur, Informatik-Grundlagen, System-
programmierung, Betriebssystem-Entwicklung, Software
Engineering, IT-Sicherheit
- Seit 2010 Doktorand (Univ. Erlangen-Nürnberg), Abschluss im
Laufe dieses Jahres



Inhaltsübersicht

- **A. Datenformate und Wandlung
(von DTaus bis XML)**
 - Grafik, Video
 - Text
 - Applikationsformate (z.B. GIS, CAD usw.)
 - Kommunikationsformate
 - Konverter
 - Kodierungen (ASCII, ISO-Latin-*, Unicode)
- **B. PC als Arbeitsplatz**

- **C. Ergonomie und Arbeitsschutz**
 - **D. Rechnerstrukturen**
 - Prozessor-Architekturen, Befehlssätze
 - **E. Zentrale und verteilte IT-Infrastrukturen**
 - **F. Telekommunikation**
 - Geräte, Protokolle, Dienste
→ Eigenstudium, Seminararbeiten
 - **G. Infrastruktur-Technologie**
- } Skript im Campus-System
- } ggf. weitere Literatur

Vorlesungsübersicht

Seminar

Wiss. Arbeiten

Datenformate und Wandlung

PC als Arbeitsplatz

Ergonomie und Arbeitsschutz

Rechnerstrukturen

Zentrale / verteilte IT-Infrastrukturen

Seminararbeiten

- 23 Teilnehmer laut Campus-System
- Seminar-Themen aus den Bereichen „F. Telekommunikation“ und „G. Infrastruktur-Technologie“
- Literatur im Campus-System verfügbar
 - [1] Firoz Kaderali: Skript „Kommunikationsnetze und Protokolle“
 - [2] Peter Richert: Skript „Kommunikationssysteme“
 - [3] H. R. van As: Skript „Datenkommunikation“

- Themenwahl z. B. auf Basis des Skripts „Kommunikationsnetze und Protokolle“ von Prof. Firoz Kaderali,
[http://www.kaderali.de/fileadmin/vorlesungsskripte/Buch KP \(A4\).pdf](http://www.kaderali.de/fileadmin/vorlesungsskripte/Buch_KP_(A4).pdf)
- oder eigene Themenvorschläge, gerne auch aus der eigenen Praxis
- bitte keine Doppel-Themenbelegung
- Vorträge: während der Präsenztermine am 29.05. / 30.05.
- Vortragsfolien bis 27.05. per Mail an mich

Wissenschaftliches Arbeiten

Inhalte

- Wie lese ich Fachbücher/Skripte/Papers?
- Wie schreibe ich eine Seminararbeit?
- Wie halte ich einen Seminarvortrag?

- Was ist das Themengebiet der Quelle?
- Was ist die zentrale Fragestellung?
 - Welches Thema wird im Detail behandelt?
 - Welche Aspekte sind besonders wichtig oder vernachlässigbar?
 - Welche Aspekte bleiben in der Quelle unklar?
- Literaturangaben auswerten
 - Grundlagen
 - Weiterführende Literatur

- Gliederung erstellen
 - Typisch:
 - Kap. 1: Einleitung
 - Kap. 2:
 - Kap. $n-1$:
 - Kap. n : Zusammenfassung
- (Inhalt)

- Einleitung besteht aus:
 - Motivation / Themenfeld
 - Welches Themengebiet behandelt die Arbeit
 - Warum ist es relevant?
 - Thema der (Seminar-) Arbeit
 - Fragestellung in dieser Ausarbeitung
 - Bezug auf das Paper / die Papers
 - Welche Lösung wird im Paper vorgeschlagen?
 - Struktur der Arbeit
 - Wie ist diese Arbeit aufgebaut? Roter Faden?

- Leser wollen i. d. R. nicht jedes Dokument vollständig lesen
- Darum: In der Einleitung vollständigen Überblick über die wichtigsten Inhalte geben (→ „Executive Summary“)

- In den inhaltlichen Kapiteln kann sich die Grobstruktur der Ausarbeitung wiederholen, also
 - Kap. *i.1*: Einleitung (zu Thema *i*)
 - Worum geht es?
 - Roter Faden
 - Anknüpfen an vorangehendes Kapitel
 - Kap. *i.2*: ...
 - Kap. *i.m*: Zusammenfassung
 - Worum ging es?
 - ggf. Ausblick auf Folgekapitel

Redundanz
ist OK, hilft bei
Orientierung

- Kurze Zusammenfassung der Ausarbeitung (Kenntnis der Arbeit kann hier vorausgesetzt werden)
- Falls sinnvoll: Hinweis auf offene Punkte, unbeantwortete Fragen

- Häufiges Problem:
Quelle viel zu lang / Quelle viel zu kurz
- Auswahl
- Ergänzung (weitere Literatur)
- Seminausarbeitung und Vortrag müssen nicht deckungsgleich sein
- Ausarbeitung „in sich geschlossen“ (ohne weitere Literatur verständlich)
- Frage beantworten: Warum sollte man meine Ausarbeitung (statt der Originaltexte) lesen?

- Beim Strukturieren bereits die relevanten Textstellen der Originalliteratur vermerken
 - Welcher Teil der Arbeit bezieht sich auf welchen Teil der Originalliteratur?
- Im Text eine Referenz einbauen falls ...
 - wesentliche Ideen, Zahlen, Bilder, Tabellen usw. übernommen werden
 - ganze Teile wortwörtlich übernommen werden (Zitate)
- Zitate als solche kenntlich machen („...“ [Referenz])
- Referenzen möglichst genau (am besten mit Seitenangabe, vor allem bei Büchern)

- Quellen kritisch bearbeiten
 - Nur weil etwas in einer Quelle steht, muss es nicht richtig sein → kein blindes Vertrauen
 - Technischen Beschreibungen und Erklärungen kann man auch widersprechen
 - Wenn unklar: Weitere Literatur zum selben Thema suchen, vergleichen
 - Kein direkter Zusammenhang zwischen „Zitierfähigkeit“ und Glaubwürdigkeit einer Quelle

- Im Text stehen Referenzen (Verweise) auf das Literaturverzeichnis
 - Referenzen sind meist Kürzel in eckigen Klammern
 - Unterstützung durch LaTeX
- Arten des Zitierens:
 - Aktiv: In [1] wurde gezeigt ... [1, S. 35]
 - Passiv: Maier und Müller [1] haben gezeigt ...
 - Möglichst für eine Art entscheiden und konsistent verwenden

- Informationen, die (unter Informatikern) allgemein bekannt sind, brauchen keinen Beleg durch Quellenangabe
 - „C ist eine häufig für systemnahe Programmierung verwendete Sprache.“ (ohne Beleg)
 - „Das Betriebssystem ZonkOS wurde nicht in C, sondern in Fortran implementiert [12].“ (mit Beleg)
- Zielgruppe: andere (Wirtsch.-) Informatik-Studenten, konkret: die übrigen Kursteilnehmer

- Mehrere Referenzen, Seitenangaben
 - [1, 2, 4]
 - [1–3, 5]
 - [4, S. 15, 6, S. 34] → besser zerlegen
 - LaTeX macht das automatisch
- Viele Autoren
 - Müller et al.: ab vier Autoren; in engl. Texten ab drei Autoren, vgl. Wikipedia [3]
 - im Literaturverzeichnis aber alle Autoren nennen

- Zitierweise
 - Bitte keine Fußnoten für Literatur-Verweise,
 - sondern einfach z. B. „[6]“ im Text als Verweis auf Eintrag [6] im Literaturverzeichnis
 - bei längeren Quellen (wie dem TK-Skript) immer mit Seitenangabe, in der Form „[6, S. 119]“

- alle verwendeten (und nur die!) Quellen aufführen
 - sortiert nach Nachname des Erstautors, dann nach Datum
- Eintrag enthält
 - Namen der Autoren, Titel der Publikation/des Beitrags
 - ggf. Name der Zeitschrift/Konferenz, Band, Nummer, Seitenzahl / Ort
 - Erscheinungsjahr
 - möglichst genau
- Quellen im Web: Autoren, Titel, URL und Zugriffsdatum (wenn es eine Originalquelle gibt, diese zitieren!)

- Wikipedia: u. U. akzeptable Quelle. Ganz gut für technische Dinge
- Whitepapers von Unternehmen: da fehlen oft wichtige Angaben (z. B. Autoren),
→ Firma als Hrsg.
- „keinen Schrott zitieren“

- **Negativ-Beispiel**

Bei einem Vergleich forensisch nutzbarer Informationen in Windows-Dateisystemen sieht man u. a., dass FAT [1] nur einen einzigen Zeitstempel pro Datei speichert, während NTFS [2] für jede Datei vier Zeitstempel verwaltet [3]. Ergänzend gibt ...

[1] http://de.wikipedia.org/wiki/File_Allocation_Table

[2] <http://de.wikipedia.org/wiki/NTFS>

[3] <http://articles.forensicfocus.com/2013/04/06/interpretation-of-ntfs-timestamps/>

[4] <http://books.google.de/books?id=fummOICB9IgC&pg=PA148&lpg=PA148&dq=ntfs+%22delete+time%22+timestamp&source=bl&ots=FOFB7xD3MY&sig=awkTk3V-f1dR3E6ivx46-Gc7xs4&hl=en&sa=X&ei=loFXU6mOC4LNtQbLuwE&ved=0CDwQ6AEwAg#v=onepage&q=ntfs%20%22delete%20time%22%20timestamp&f=false>

Bei einem Vergleich forensisch nutzbarer Informationen in Windows-Dateisystemen sieht man u. a., dass FAT [1] nur einen einzigen Zeitstempel pro Datei speichert, während NTFS [2] für jede Datei vier Zeitstempel (create, modify, access, change) verwaltet [3]. Der vierte Zeitstempel (change) wird für gelöschte Dateien auch als delete-Timestamp betrachtet [4, S. 148].

- [1] Microsoft: Microsoft EFI FAT32 File System Specification, 2000, Whitepaper, <http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc>, Onlineabruf am 23.04.2014
- [2] Microsoft: NTFS Technical Reference, 2003, <http://technet.microsoft.com/en-us/library/cc758691%28WS.10%29.aspx>, Onlineabruf am 23.04.2014
- [3] wpathulin: „Interpretation of NTFS Timestamps“, Forensic Focus, 2013, <http://articles.forensicfocus.com/2013/04/06/interpretation-of-ntfs-timestamps/>, Onlineabruf am 23.04.2014
- [4] Greg Gogolin: *Digital Forensics Explained*, Auerbach Publications, 2012, ISBN: 1-4398-7495-6

Positiv-Beispiel

- In eigenen Worten formulieren
- Keine Textblöcke aus der Quelle übernehmen, auch keine Wort-für-Wort-Übersetzung
- kein Copy & Paste
 - Plagiat, ist in Prüfungs- und Seminararbeiten ein Grund für Nichtbestehen; an einigen Hochschulen sogar für offizielle Rügen oder Exmatrikulation (Abschlussarbeiten)
 - fällt auf (Stilbruch etc.) und wird geprüft
 - Reines Umformulieren von Satz für Satz ist aber genauso Plagiat!

- Eigene Beiträge kennzeichnen (wenn Sie z. B. eigene Praxistests etc. machen)
- Stil: Passivkonstruktionen, Substantivierungen meiden

- Copy & Paste (wortwörtliche Wiedergabe)
- Paraphrasierung (Wiedergabe mit eigenen Worten),
- Übersetzungsplagiat
- Strukturplagiat
- „Shake and Paste“ [1]

[1] http://plagiat.htw-berlin.de/ff/schule/3_2/wie

1 INTRODUCTION

CONSIDER a set of parties who wish to correctly compute some common function F of their local inputs, while keeping their local data as private as possible, but who do not trust each other, nor the channels by which they communicate. This is the problem of *Secure Multiparty Computation* (SMC) [1]. SMC is a very general security problem, i.e., it can be used to solve various real-life problems such as distributed voting, private bidding and online auctions, sharing of signature or decryption functions and so on. Unfortunately, solving SMC is—without extra assumptions—very expensive in terms of communication (number of messages), resilience (amount of redundancy), and time (number of synchronous rounds).

TrustedPals [2] is a smart card-based security framework for solving SMC which allows much more efficient solutions to the problem. Conceptually, *TrustedPals* considers a distributed system in which processes are locally equipped with tamper-proof security modules. In practice, processes are implemented as a Java desktop application and security

links: R. Cortiñas, F. C. Freiling, M. Ghajar-Azadanlou, A. Lafuente, M. Larrea, L. Draque Penso, I. Soraluze: "Secure Failure Detection and Consensus in „TrustedPals“, IEEE TDSC (July-Aug. 2012, vol. 9 no. 4, pp. 608-623)

rechts: Khaderbasha, Shakeel, Babu: „Secure Failure Detection and Aggrements In Trusted Pals“, Int. Journal of Innovative Technologies (Vol. 01, Issue 05, Oct 2013)

1. INTRODUCTION

Consider a set of parties who wish to correctly compute some common function F of their local inputs, while keeping their local data as private as possible, but who do not trust each other, nor the channels by which they communicate. This is the problem of *Secure Multiparty Computation* (SMC) [1]. SMC is a very general security problem, i.e., it can be used to solve various real-life problems such as distributed voting, private bidding and online auctions, sharing of signature or decryption functions and so on. Unfortunately, solving SMC is without extra assumptions very expensive in terms of communication (number of messages), resilience (amount of redundancy), and time (number of synchronous rounds).

TrustedPals [2] is a smart card-based security framework for solving SMC which allows much more efficient solutions to the problem. Conceptually, *TrustedPals* considers a distributed system in which processes are

- Jedes Bild hat eine Bildunterschrift
 - beschreibt das Bild – ohne den restlichen Text vorauszusetzen
 - ist keine Kopie eines Satzes aus dem Text
- Für Tabellen gilt dasselbe analog

- Ziel: 13–15 Seiten,
 - zzgl. Anhang, Verzeichnisse
- Format / Layout:
 - DIN-A4, 12pt Times New Roman, 1,5-zeilig, Blocksatz, autom. Silbentrennung
 - Abstände: oben 3 cm, unten 2 cm, links 4 cm, rechts 2 cm
 - Seitenzahlen: oben (zwischen Textblock und Papierrand)
 - kein Inhaltsverzeichnis

- Konsistent sein
 - beim Zitieren
 - bei der Verwendung von Fachbegriffen
- Sich an die (deutsche) Sprache halten
 - englische Wörter übersetzen und (wenn nötig) den Originalbegriff beim ersten Mal nennen
 - steht das Wort im Duden? Dann ist es deutsch
 - Fachbegriffe wie IP-Stack, Bridge, Gateway sind ok (1x erklären)
 - teilweise Geschmacksache („I/O“ oder „E/A“?)
- Sparsame Typographie

Häufige Fehler (1)

- Beschwerden, dass der Stoff zu umfangreich ist. Aus jedem Thema lässt sich problemlos ein 10-, 20- oder 60-minütiger Vortrag gestalten. Das ist eben die Vortragskunst.
- Eine Motivation wird vernachlässigt.
- Sich nicht mit dem roten Faden oder der „Message“ auseinanderzusetzen.
- Es werden Dinge vorausgesetzt, die bei Teilen des Publikums nicht bekannt sind.
- Fachbegriffe werden nicht erklärt, Abkürzungen nicht ausgeführt.

Quelle: Weicker & Weicker, Lehrstuhl 4, Univ. Erl.-Nbg. [5]

Häufige Fehler (2)

- Gliederung wird anhand von Stichworten erstellt, die dann in jeweils ein bis zwei Sätzen ausgeführt werden. Meist geht hierbei der rote Faden verloren: die Arbeit wirkt abgehackt und unverständlich
- Gliederung der Folien für den Vortrag oder Gliederung des Vortrags für die Ausarbeitung verwenden
- Zu viel Zeit auf Folien verwenden, aber das Gesprochene improvisieren
- Den Folieninhalt nur vorlesen oder sich an den Folien entlang zu hangeln ohne den Vortrag lebendig werden zu lassen
- Mit Powerpoint-Effekten über inhaltliche Defizite hinwegtäuschen

Quelle: Weicker & Weicker, Lehrstuhl 4, Univ. Erl.-Nbg. [5]

- [1] Marcus Deininger, Horst Lichter, Jochen Ludewig, Kurt Schneider: Studienarbeiten. Ein Leitfaden zur Vorbereitung, Durchführung und Betreuung von Studien-, Diplom-, Abschluss- und Doktorarbeiten am Beispiel Informatik. vdf, 5. Auflage, 2005.
- [2] Albers et al.: Gute wissenschaftliche Praxis für das Verfassen wissenschaftlicher Qualifikationsarbeiten, http://www.hochschulverband.de/cms1/uploads/media/Gute_wiss._Praxis_Fakultaetentage.pdf, Juli 2012.
- [3] Wikipedia zu „et al.“: http://de.wikipedia.org/wiki/Et_al.
- [4] Peter Rechenberg: Technisches Schreiben. Hanser, 2. Auflage, 2003.
- [5] Karsten Weicker, Nicole Weicker: Seminarrichtlinien am Lehrstuhl 4, https://www4.cs.fau.de/Lehre/SS11/PS_KVBK/info/Weicker.pdf
- [6] Markus Rath: Leitfaden zur Anfertigung von Seminar-, Studien- und Diplomarbeiten, http://www.wi-inf.uni-duisburg-essen.de/FGFrank/documents/Lehre/Leitfaden_WissArbeit.pdf

- FOM-Dokument:
„Leitfaden zur formalen Gestaltung von Seminar- und Abschlussarbeiten“
- über Campus-System verfügbar (von Kursseite aus verlinkt)

Datenformate und Wandlung

Gliederung

- Daten und Informationen
- Zahlensysteme
- ASCII und andere Zeichenkodierungen (ISO-Latin-*, Unicode / UTF)
- Digitalisierung (Bilder, Audio, Video)
- Kompression

- Datenformate
 - Textformate
 - Grafikformate
 - Konverter
- Kommunikationsformate
- Applikationsformate
- Archive und Programmpakete

A. Datenformate und Wandlung

Was bedeutet „Information“?

- eine Form von Wissen
- Information hat immer eine Bedeutung
 - Text in einer unbekannten Fremdsprache: bedeutungslos
 - Bedeutung also abhängig vom Betrachter
 - Zusammenhang wichtig (etwa: Text aus einem fremden Fachgebiet)
- Information informiert:
 - Herr Müller informiert Frau Meier
 - Ich informiere mich (selbst) über ...

Informationen...

- vielseitig, z. B.
 - Fakten(wissen)
 - Konzepte (die Antwort auf: „Was sind Informationen?“ ist auch eine Information)
 - Anleitungen (z. B. Algorithmen)
- Information an sich hat keine „standardisierte“ Darstellung, etwa in Schriftform

Weitergabe von Informationen

- Viele Wege denkbar
 - im direkten Gespräch erzählen
 - eine Vorlesungsstunde darüber halten
 - aufschreiben (Buch? Notizzettel? Tafel?)
 - etwas (ohne Worte) vorführen

und „Daten“?

- Repräsentation von Information(en)
- mit oder ohne Struktur

Name	Telefon	Ort
Anton Müller	089/1234567	München
Berta Meier	0241/7343	Aachen
Christian Bauer	0211/64834	Düsseldorf
Dagmar Grün	02131/46734	Neuss

Wegbeschreibung: aus dem Münchener HBf nach Ankunft links raus, nach dem Ausgang links halten bis zur nächsten großen Kreuzung, diagonal die Kreuzung überqueren, Eingang der FOM liegt an Hauptstraße, im Gebäude in den ersten Stock zur Bibliothek

Daten laut Wikipedia:

„In der Informatik und Datenverarbeitung versteht man Daten als (Maschinen-) lesbare und bearbeitbare in der Regel **digitale Repräsentation von Information**. Die Information wird dazu meist zunächst in Zeichen (bzw. Zeichenketten) **kodiert**, deren Aufbau strengen Regeln folgt, der so genannten **Syntax**. Um aus Daten Informationen zu gewinnen, müssen sie in einem **Bedeutungskontext** interpretiert werden.“

Quelle: <http://de.wikipedia.org/wiki/Daten>

- **Repräsentation**

- eine Form der Darstellung, die gewisse Regeln einhält
- z. B.: 5 vs. „fünf“ vs. IIII vs. ~~IIII~~ vs. 
- für den Betrachter: offensichtlich alle gleichwertig
- Digitale Repräsentation: „im Computer“
- Frage: **Was** kann man **wie** im Computer speichern?

- **Kodierung durch Zeichen (-ketten)**
 - Was ist ein Zeichen?
 - Vorschlag: A-Z, a-z, 0-9 und Leerzeichen – gut?
 - Wie speichert der Computer Zeichen?
 - Prinzipiell kennt der Rechner nur zwei Werte:
0 und 1
 - Für alles andere: Umwandlung in Kombinationen
aus Nullen und Einsen

- **Bit:** 0 / 1 an / aus wahr / falsch
(binary digit – Binärziffer)
- **Bit-Folge:** Zeichenkette, die aus Nullen und Einsen besteht, z. B. 00101, 10
- **Byte:** Bit-Folge aus acht Bits (auch: Oktett),
z. B. 01101011, 11111111, 00000000
- Führende Nullen kann man weglassen:
00000010 = 10
- **Wie viele (unterschiedliche) Bytes gibt es?**

- Bitfolgen auch als Zahlenwerte auffassen:
 $1_b = 1$, $10_b = 2$, $11_b = 3$, $100_b = 4$, $101_b = 5 \dots$
- **Dualzahlen:** Zahlensystem, das zur Darstellung von Zahlen nur die Ziffern 0 und 1 verwendet. Auch **Binärzahlen** genannt
- Rechnen mit Dualzahlen funktioniert wie mit normalen Zahlen. Hier:
 - Addieren + Subtrahieren
 - Multiplizieren

- Addieren:

$$\begin{array}{r}
 001001001 \\
 + 010101110 \\
 \hline
 = 011110111
 \end{array}$$

Übertrag

- Subtrahieren:

$$\begin{array}{r}
 011010111 \\
 - 010101110 \\
 \hline
 = 000101001
 \end{array}$$

Übertrag

- Multiplizieren

101001 x 1001

101001

000000

000000

101001

1

Übertrag

101110001

Probe:

$$101001_b = 41$$

$$1001_b = 9$$

$$41 \times 9 = 369$$

$$101110001_b = 369$$

- Grundlage: Wie funktionieren unsere Zahlen?
- $4982 = 4 \times 1000 + 9 \times 100 + 8 \times 10 + 2 \times 1$
 $= 4 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 2 \times 10^0$
 (von rechts nach links: Einer, Zehner, Hunderter, Tausender, ...)
- Bei Dualzahlen geht das genauso:
- $10011_b = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
 $= 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
 $= 19$

Umrechnung dual $\rightarrow$ dezimal (2/3)

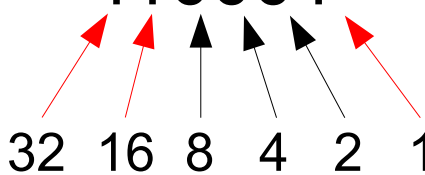
umgekehrt: dezimal $\rightarrow$ dual etwas komplizierter
49 als Dualzahl?

49	:	2	=	24,	Rest	1	Einser
24	:	2	=	12,	Rest	0	Zweier
12	:	2	=	6,	Rest	0	Vierer
6	:	2	=	3,	Rest	0	Achter
3	:	2	=	1,	Rest	1	
1	:	2	=	0,	Rest	1	

$\rightarrow$ 110001_b (32+16+1)

Warum geht das? Rechnen Sie mal direkt

$$110001_b : 2 \dots \quad 110001_b = 2 \times 11000_b + 1$$

- oder durch „Kopfrechnen“:
 - kleinste 2er-Potenz (2, 4, 8, 16, ...), die in die Zahl 49 „passt“: 32 (nächst größere: 64)
 - also $49 = 32 + ? = 32 + 17$
 - $17 = 16 + ? = 16 + 1$
 - Damit: $49 = 32 + 16 + 1$
 $= 100000 + 010000 + 000001$
 $= 110001$


32 16 8 4 2 1

Bit, Byte, (Doppel-) Wort

Einheit	Erklärung	Werte	Anzahl Werte
Bit	Binary Digit	0, 1	2
Byte	8 Bit	0 – 255	256
Wort	2 Byte, 16 Bit	0 – 65535	65536
Doppelwort	2 Worte (4 Byte)	0 – 4294967295	4294967296

- Was der Computer gut kann:
Bits und Bytes speichern; auch (Doppel-)
Worte und längere Bitfolgen (mehrere Bytes
verwenden)
- Problem: allgemeine Daten speichern
- Lösung: „beliebige“ Daten auf Bitfolgen
abbilden

- Beispiel: „A-Z“ als 1-26 speichern, binär:
A = 00001, B = 00010, ...

A	00001	H	01000	O	01111	V	10110	unbenutzt 00000 11011 11100 11101 11110 11111
B	00010	I	01001	P	10000	W	10111	
C	00011	J	01010	Q	10001	X	11000	
D	00100	K	01011	R	10010	Y	11001	
E	00101	L	01100	S	10011	Z	11010	
F	00110	M	01101	T	10100			
G	00111	N	01110	U	10101			

- Also 5 Bit pro Buchstabe
- HALLO → 01000 00001 01100 01100 01111

- Praxis: „nur Großbuchstaben“ unbrauchbar
- Kleinbuchstaben, Zahlen, Sonderzeichen
- erster Standard: ASCII, enthält die Zeichen:
0-9, A-Z, a-z, das Leerzeichen „ “ und:
! " # \$ % & \ ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~

- **ASCII = American Standard Code for Information Interchange:**

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

- ASCII – und was ist mit Umlauten?
 - „einfach drauf verzichten“ – ae, oe, ue, Ae, Oe, Ue und ss stoeren hoechstens die Uebergenaen
 - Zeichen aus der Zeichentabelle werfen und durch Umlaute ersetzen, etwa
 - [→ ä,] → ö, \ → ü,
 - { → Ä, } → Ö, | → Ü usw.
 - ASCII verwendet nur 7 Bit (0-127); über achtes Bit neuen Zeichensatz mit Sonderzeichen definieren, z. B. ISO-8859-15 (Westeuropa mit €-Zeichen). Dort: 0-127 wie ASCII, 128-255 Zusatzzeichen

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
NBSP	ı	¢	£	€	¥	Š	§	š	©	ª	«	¬	ŠHY	®	ˆ
°	±	²	³	Ž	μ	¶	·	ž	¹	º	»	Œ	œ	ÿ	ı
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Quelle: http://de.wikipedia.org/wiki/ISO_8859-15

- Noch zwei weitere Zahlensysteme
- Bisher:
 - Dualzahlen – Basis 2
 - Dezimalzahlen – Basis 10
- Jetzt:
 - Oktalzahlen – Basis 8
 - Hexadezimalzahlen – Basis 16

Okta- und Hexadezimalzahlen (2/9)

Okta- und Hexadezimalzahlen

Basis 8, also

8 Ziffern:

0, 1, 2, 3,

4, 5, 6, 7

$$7_o + 1_o = 10_o$$

oktal	dez.	binär	oktal	dez.	binär
0	0	0	14	12	1100
1	1	1	15	13	1101
2	2	10	16	14	1110
3	3	11	17	15	1111
4	4	100	20	16	10000
5	5	101	...		
6	6	110	77	63	111111
7	7	111	100	64	1000000
10	8	1000	101	65	1000001
11	9	1001	102	66	1000010
12	10	1010	103	67	1000011
13	11	1011	104	68	1000100

Hexadezimalzahlen

- Basis 16, also 16 Ziffern:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
10 11 12 13 14 15

- $9_h + 1_h = A_h$,
 $F_h + 1_h = 10_h$ (mit Wert 16)

Oktal- und Hexadezimalzahlen (4/9)

hex.	dez.	binär	hex.	dez.	binär	hex.	dez.	binär
0	0	0	10	16	10000	20	32	100000
1	1	1	11	17	10001	21	33	100001
2	2	10	12	18	10010	22	34	100010
3	3	11	13	19	10011	23	35	100011
4	4	100	14	20	10100	24	36	100100
5	5	101	15	21	10101	25	37	100101
6	6	110	16	22	10110	26	38	100110
7	7	111	17	23	10111	27	39	100111
8	8	1000	18	24	11000	...		
9	9	1001	19	25	11001	FF	255	11111111
A	10	1010	1A	26	11010	100	256	100000000
B	11	1011	1B	27	11011	101	257	100000001
C	12	1100	1C	28	11100	102	258	100000010
D	13	1101	1D	29	11101	...		
E	14	1110	1E	30	11110	FFF	4095	111111111111
F	15	1111	1F	31	11111	1000	4096	1000000000000

- Umrechnen Dual $\leftrightarrow$ Oktal
und Dual $\leftrightarrow$ Hexadezimal
ist also leicht:
- Jede Ziffer einer Oktalzahl wird zu drei Bits:
 $307_o = 011\ 000\ 111_b$
- Jede Ziffer einer Hex.-Zahl wird zu vier Bits:
 $3AF_h = 0011\ 1010\ 1111_b$

Umrechnen in Dezimalzahlen: Wie bei Dualzahlen, aber mit anderer Basis

- $1101_b = 1 \times \underline{2}^3 + 1 \times \underline{2}^2 + 0 \times \underline{2}^1 + 1 \times \underline{2}^0 = 13$
- $1734_o = 1 \times \underline{8}^3 + 7 \times \underline{8}^2 + 3 \times \underline{8}^1 + 4 \times \underline{8}^0 = 988$
- $1A3F_h = 1 \times \underline{16}^3 + 10 \times \underline{16}^2 + 3 \times \underline{16}^1 + 15 \times \underline{16}^0 = 6719$
- zur Erinnerung im Dezimalsystem:
 $3921 = 3 \times 10^3 + 9 \times \underline{10}^2 + 2 \times \underline{10}^1 + 1 \times \underline{10}^0 = 6719$

Umrechnen von Dezimal- in Oktal- oder Hexadezimalzahlen:

- entweder in zwei Schritten – erst in Dualzahlen umrechnen, dann 3er- bzw. 4er-Gruppen zusammenfassen:

$$80 = 001\ 010\ 000_b = 120_o = 0101\ 0000_b = 50_h$$

- oder durch schriftliches Dividieren wie bei Umrechnen in Dualzahl, aber diesmal mit Divisor 8 oder 16

942 als Hexadezimalzahl?

$942 : 16 = 58,$	Rest	14 (E)	Einser (16^0)
$58 : 16 = 3,$	Rest	10 (A)	16er (16^1)
$3 : 16 = 0,$	Rest	3	256er (16^2)

→ $3AE_h$ ($3 \times 256 + 10 \times 16 + 14 \times 1$)

Warum geht das? Rechnen Sie mal direkt

$$3AE_h : 16 \dots \quad 3AE_h = 16 \times 3A_h + E_h$$

942 als Oktalzahl?

942 : 8	= 117,	Rest	6	Einser	(8 ⁰)
117 : 8	= 14,	Rest	5	8er	(8 ¹)
14 : 8	= 1,	Rest	6	64er	(8 ²)
1 : 8	= 0,	Rest	1	512er	(8 ³)

→ **1656_o** (1 x 512 + 6 x 64 + 5 x 8 + 6 x 1)

Wie bei Hexadez.-Zahlen: Rechnen Sie mal direkt

$$1656_o : 8 \dots \quad 1656_o = 8 \times 165_o + 6_o$$

Warum hexadezimal? (1/3)

- deckt praktische (häufig benutzte) Bereiche ab
- z. B.: Home-Computer mit 64 KByte RAM
 $64 \text{ KByte} = 64 \times 1024 \text{ Byte} = 65\,536 \text{ Byte}$
→ hexadezimal: 10000_{h} Byte
also Adressen: $0000_{\text{h}} - \text{FFFF}_{\text{h}}$
- z. B.: PC mit 1 GByte RAM
 $1 \text{ GByte} = 1024 \times 1024 \times 1024 \text{ Byte}$
 $= 1\,073\,741\,824 \text{ Byte}$
→ hexadezimal: 40000000_{h} Byte
also Adressen: $00000000_{\text{h}} - 3\text{FFFFFFF}_{\text{h}}$

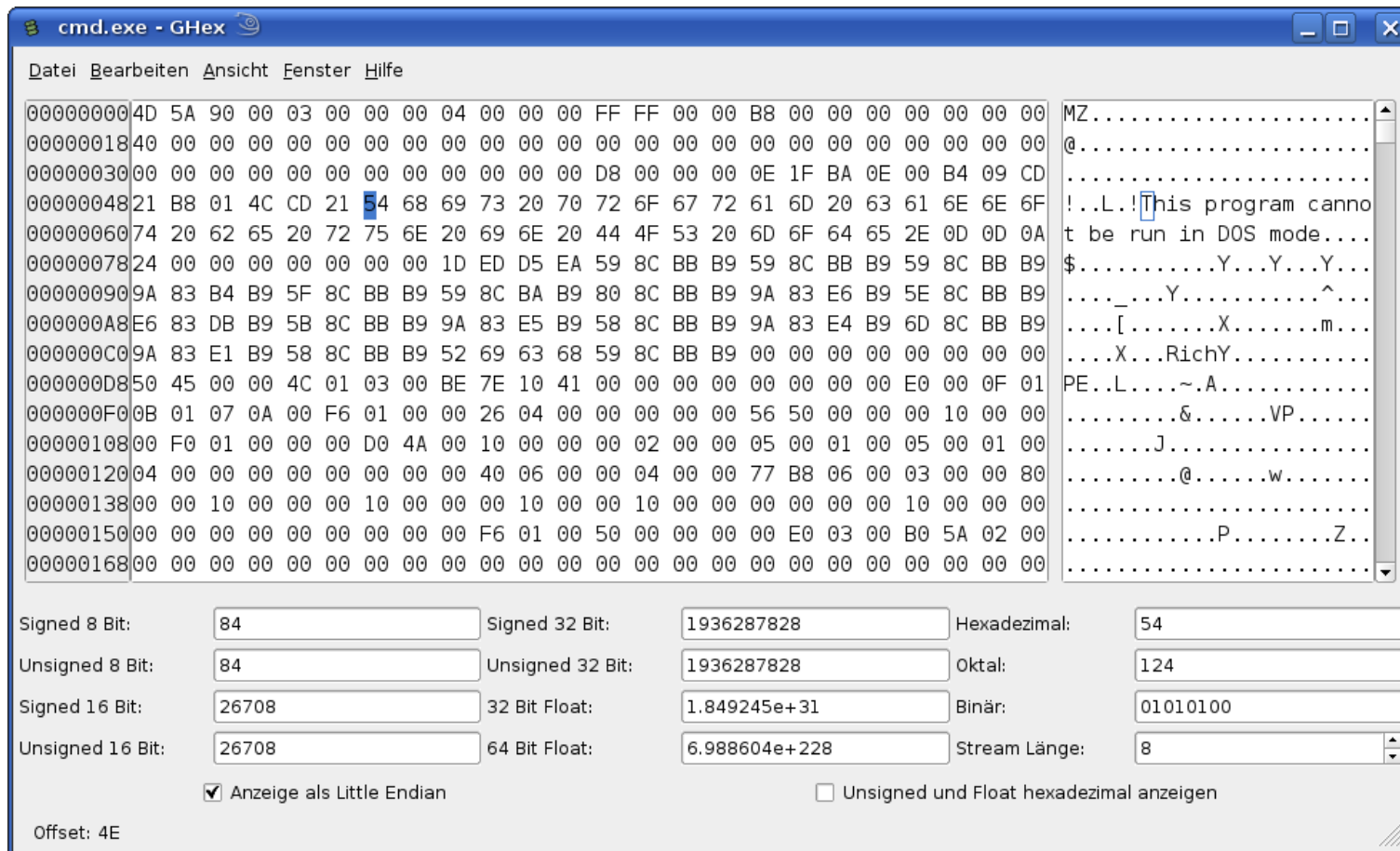
Warum hexadezimal? (2/3)

- Kodierung von Bytes (z. B.: ASCII-Zeichen), also 8 Bit, durch zwei Hex-Zahlen:

hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	
00	0	10	16	20	32	30	48	40	64	50	80	60	96	70	112	F0	240	
01	1	11	17	21	33	31	49	41	65	51	81	61	97	71	113	F1	241	
02	2	12	18	22	34	32	50	42	66	52	82	62	98	72	114	F2	242	
03	3	13	19	23	35	33	51	43	67	53	83	63	99	73	115	F3	243	
04	4	14	20	24	36	34	52	44	68	54	84	64	100	74	116	F4	244	
05	5	15	21	25	37	35	53	45	69	55	85	65	101	75	117	F5	245	
06	6	16	22	26	38	36	54	46	70	56	86	66	102	76	118	...	F6	246
07	7	17	23	27	39	37	55	47	71	57	87	67	103	77	119		F7	247
08	8	18	24	28	40	38	56	48	72	58	88	68	104	78	120		F8	248
09	9	19	25	29	41	39	57	49	73	59	89	69	105	79	121		F9	249
0A	10	1A	26	2A	42	3A	58	4A	74	5A	90	6A	106	7A	122		FA	250
0B	11	1B	27	2B	43	3B	59	4B	75	5B	91	6B	107	7B	123		FB	251
0C	12	1C	28	2C	44	3C	60	4C	76	5C	92	6C	108	7C	124		FC	252
0D	13	1D	29	2D	45	3D	61	4D	77	5D	93	6D	109	7D	125		FD	253
0E	14	1E	30	2E	46	3E	62	4E	78	5E	94	6E	110	7E	126		FE	254
0F	15	1F	31	2F	47	3F	63	4F	79	5F	95	6F	111	7F	127		FF	255

Warum hexadezimal? (3/3)

- Für viele Aufgaben ist Hex-Darstellung der Standard, z. B. Analyse von Binärdateien:



- Sie kennen nun bereits vier Zahlensysteme:

System	Basis	Ziffern
• Dezimalsystem	10	0 – 9
• Dualsystem	2	0, 1
• Oktalsystem	8	0 – 7
• Hexadezimalsystem	16	0 – 9, A – F
• Das Ganze lässt sich auf beliebige Basen verallgemeinern: Basis n mit n Ziffern; z. B. Basis 32 mit Ziffern 0 – 9, A (10) – V (31)		

1. Stellen Sie den folgenden Text:

Infrastruktur

- im Dezimalformat (normales 10er-System),
- im Hexadezimalformat, Oktalformat und Dualformat dar. Wählen Sie dafür eine sinnvolle Reihenfolge!

Dazu benötigen Sie:

- ASCII-Tabelle (1. Schritt)
- Hex/Oktal/Dual-Tabellen (2. Schritt)

2. a) Wie viele 16-stellige Bitfolgen gibt es?
b) Wie viele verschiedene Worte („Doppel-Bytes“) gibt es?
3. Rechnen Sie die Hexadezimalzahl **A01F_h** in Dual- und Oktaldarstellung um.
4. Wenn Sie wissen, dass **FF_h = 100_h – 1** gilt, wie können Sie dann schnell **FFF_h** und **FFFF_h** in Dezimalzahlen umrechnen?

- Aus welchen Zahlensystemen kann die folgende Zahl stammen? Aus welchen nicht? Warum? **41823**
 - Berücksichtigen Sie: Dual-, Oktal-, Dezimal- und Hexadezimalsystem.
- Können Sie **2049** (dezimal) ganz schnell in eine Dualzahl umrechnen?
- Wie sieht das Wort „Übung“ in ASCII-Darstellung aus? (Achtung: Fangfrage)

- Was ist größer: 74_h oder 75 ?
- Wie viele Hexadezimalzahlen liegen zwischen $7A7_h$ und $8B8_h$? (Zählen Sie die beiden „begrenzenden“ Zahlen mit.)

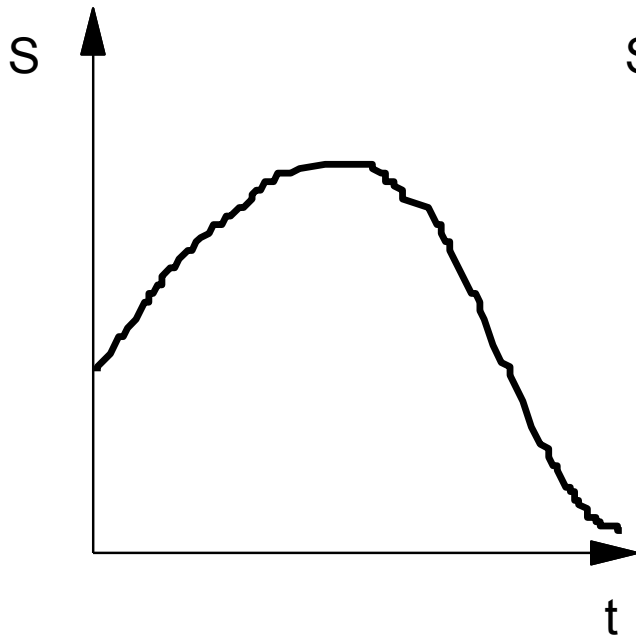
Name (Symbol)	SI-konforme Bedeutung ¹⁾	„klassische“ Bedeutung	Unterschied
Kilobyte (kB)	10^3 Byte = 1.000 Byte	2^{10} Byte = 1.024 Byte	2,4 %
Megabyte (MB)	10^6 Byte = 1.000.000 Byte	2^{20} Byte = 1.048.576 Byte	4,9 %
Gigabyte (GB)	10^9 Byte = 1.000.000.000 Byte	2^{30} Byte = 1.073.741.824 Byte	7,4 %
Terabyte (TB)	10^{12} Byte = 1.000.000.000.000 Byte	2^{40} Byte = 1.099.511.627.776 Byte	10,0 %
Petabyte (PB)	10^{15} Byte = 1.000.000.000.000.000 Byte	2^{50} Byte = 1.125.899.906.842.624 Byte	12,6 %
Exabyte (EB)	10^{18} Byte = 1.000.000.000.000.000.000 Byte	2^{60} Byte = 1.152.921.504.606.846.976 Byte	15,3 %
Zettabyte (ZB)	10^{21} Byte = 1.000.000.000.000.000.000.000 Byte	2^{70} Byte = 1.180.591.620.717.411.303.424 Byte	18,1 %

¹⁾ SI: Internationales Einheitensystem (Système International d'Unités)

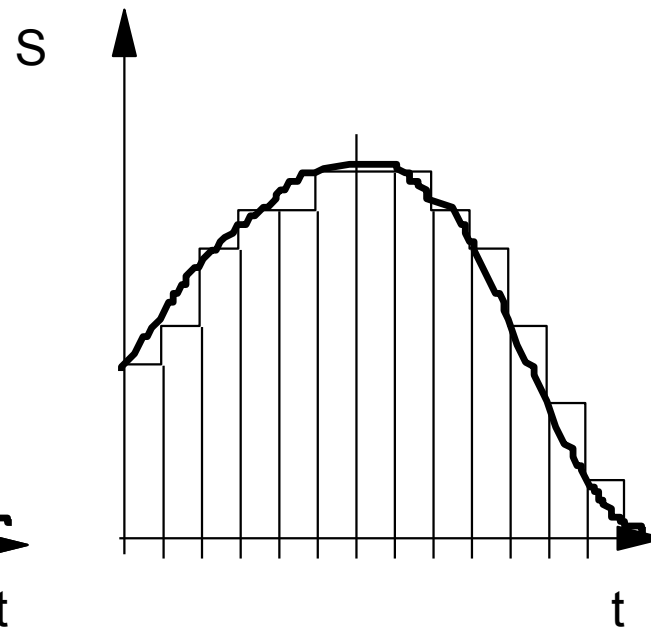
Quelle: Wikipedia, „Byte“

- Bisher: Einfache Texte (z. B. im ASCII- oder ISO-8859-15-Format) in Bytes gespeichert
- Was tun mit analogen Daten?
 - Wie speichert ein digitaler Fotoapparat Bilder?
 - Wie speichert ein MP3-Player Musik?
 - Was unterscheidet Schallplatte und CD?
- Digitalisierung von analogen Daten

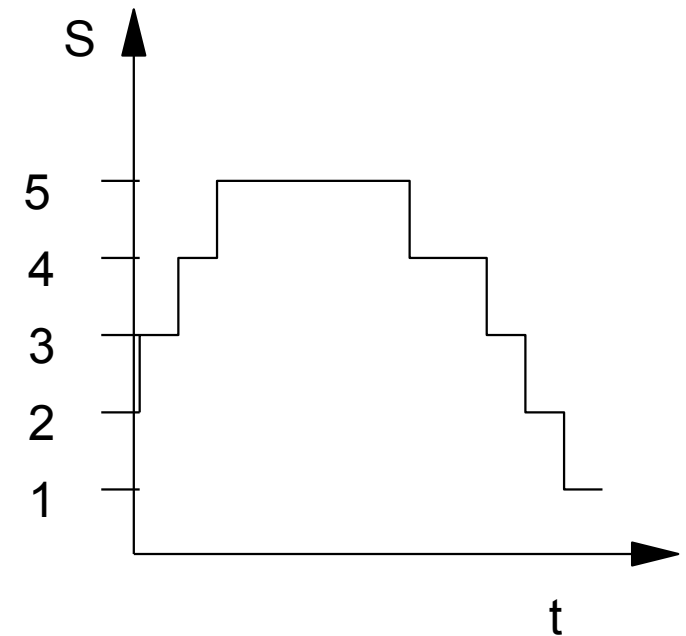
- **Rasterung (Diskretisierung):**
Abtasten (Sampling) an diskreten zeitlichen oder örtlichen Punkten
 - diskret = nur endlich viele Werte
 - Gegenteil: kontinuierlich, z.B. Werte aus $\mathbb{Q}$
- **Quantisierung** der abgetasteten Werte
(runden auf wenige mögliche Werte)
- **Digitalisierung** (Kodierung): Darstellung des abgetasteten und quantisierten Signals als Digitalcode



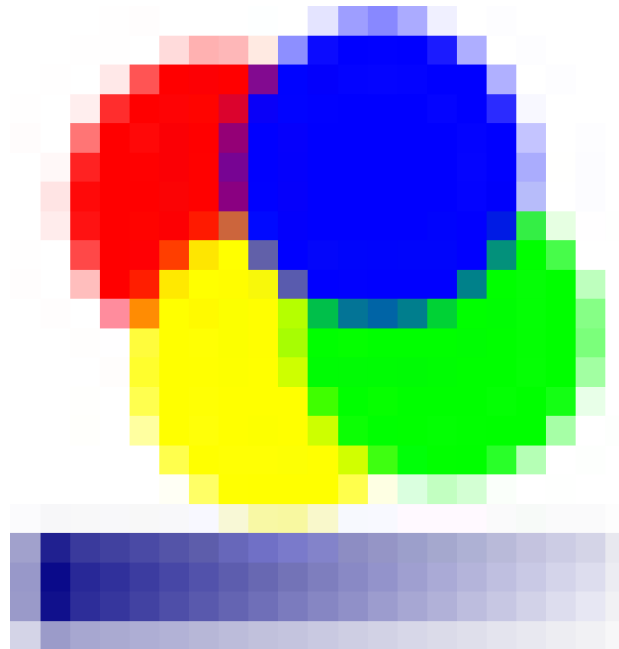
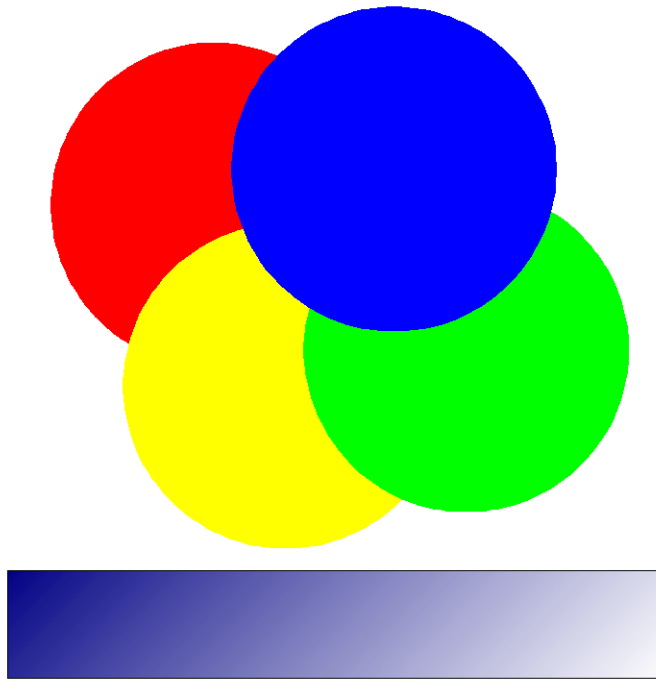
analoges Signal



„gesampelt“: Proben
zu festen Zeitpunkten
gezogen – noch mit
exakten Werten



quantisiert: alle
Messwerte auf
1, 2, 3, 4, 5
gerundet



gleichmäßige Färbung
über ganze Bereiche



begrenzt viele (diskrete)
Farbabstufungen

Kodierung von Farbinformation: Die Farbe eines Bildpunktes ist eine additive Mischung der 3 Grundfarben Rot, Grün und Blau → für Darstellung des Farbbildes wird die Rot-, Grün- und Blau-Intensität für jeden Bildpunkt separat digitalisiert.

- Text: ASCII/ANSI/Unicode (.txt), RTF (.rtf), Word (.doc), LibreOffice (.odt)
- Tabellen: Excel (.xls), LibreOffice (.ods)
- Dokumente: .doc, .pdf, HTML (.html / .htm), ...
- strukturierte Daten: XML
- Rastergrafik: Bitmap (.bmp), Graphics Interchange Format (.gif), JPEG, TIFF, PNG
- CAD: VDA-FS, IGES, STEP
- Audiodateien: MP3 (.mp3), Ogg Vorbis (.ogg)
- Bildfolgen (Video): MPEG

- Reduktion der Datenmenge durch Kompression
 - verlustfrei
 - verlustbehaftet
- Fehlerhafte Darstellung aufgrund ungleicher Zeichenvorräte bei Sender und Empfänger (z. B. bei Darstellung von nationalen Sonderzeichen)
- Fehlerhafte Übertragung
→ Codes mit Fehlererkennung, Fehlerkorrektur
- Verschlüsselung, Signatur

- Einfaches Verfahren:
 - „AAAAABBBBBCCCCCCCCCDD“ → „5A2B8C2D“
(kürzer)
 - aber: „ABCDAB“ → „1A1B1C1D1A1B“ (länger)
- verlustbehaftet:
 - „IT-Infrastruktur“ → „it infrastruktur“
(kleineres Alphabet: a-z + Leerzeichen = 27 Zeichen, weniger Bits pro Zeichen)
 - „IT-Infrastruktur“ → „TNFRSTRKTR“
(gleiches Prinzip; hier: Verzicht auf Vokale)

- Besseres Verfahren: Buchstabenhäufigkeiten
 - Welche Buchstaben kommen wie häufig vor?
Reihenfolge („top ten“) festlegen, z. B.
a, e, i, o, u, n, t, s, r, p, k, l, m, ..., q, y
- Dann Kodierung
 - von „häufigen“ Buchstaben in kurze Bitfolgen
 - und von „seltenen“ Buchstaben in lange Bitfolgen
 - z. B. a → 000, e → 001, i → 010, o → 011,
u → 1000, n → 1001, t → 1010, s → 1011,
r → 11000, k → 11001, l → 11010, m → 11011,
..., q → 11111001, y → 11111010

- Beispiele mit obiger Tabelle:
 - „nein“ → 1001 001 010 1001
 - „test“ → 1010 001 1011 1010
 - „mlml“ → 11011 11010 11011 11010
 - „qyqy“ → 11111001 11111010 11111001 11111010
- Je „unwahrscheinlicher“ ein Wort, desto länger seine Kodierung

- Übertragung von digitalen Daten (etwa über eine Telefonleitung oder ein Datenkabel zwischen zwei PCs) ist oft fehlerhaft:
 - einzelne Bits „kippen“ bei der Übertragung
 - einzelne Bits gehen bei der Übertragung komplett verloren
- Fehlererkennung: Übertragen von zusätzlichen Informationen, mit denen Fehler erkannt werden

- Einfachste Möglichkeit: Alles doppelt senden
 - Empfänger vergleicht die beiden empfangenen Informationen, die identisch sein sollten – sind sie es nicht, hat es einen Fehler gegeben
 - Wahrscheinlichkeit, dass die Daten zweimal mit exakt demselben Fehler übertragen werden, ist klein
 - Verfahren ist aber sehr aufwendig:
 - Verdopplung der übertragenen Datenmenge
 - d. h. Halbierung der Übertragungsgeschwindigkeit

- Idee: Daten mit **Prüfsummen** versehen
 - einfaches Beispiel: 7 Bit (ASCII-Zeichen) um sog. **Paritäts-Bit** ergänzen:
 - Im ASCII-Code ist das höchstwertige (ganz links) stehende Bit immer 0
 - Verwende dieses Bit wie folgt:
 - Wenn die Summe der übrigen Bits ungerade ist, setze es auf 1
 - Wenn die Summe der übrigen Bits gerade ist, setze es auf 0

ASCII + Paritätsbit

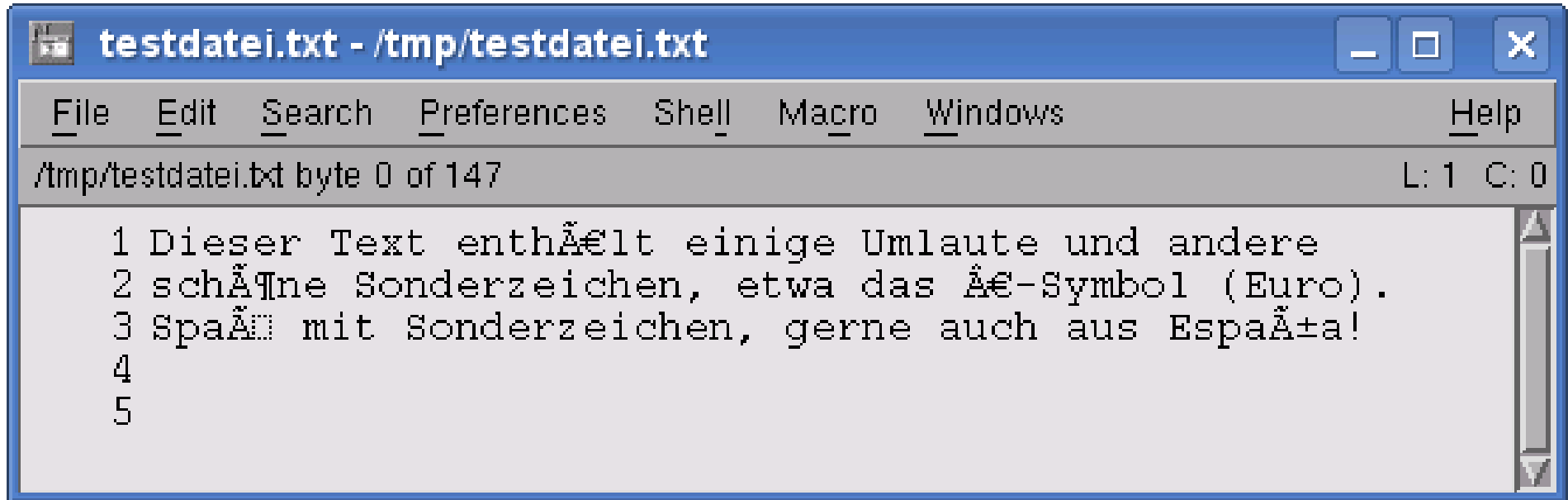
	ASCII	ASCII binär (nur 7 Bit)	Anzahl Einsen	ASCII + Paritätsbit
A	65	01000001	gerade	0 1000001
B	66	01000010	gerade	0 1000010
C	67	01000011	ungerade	1 1000011
D	68	01000100	gerade	0 1000100
E	69	01000101	ungerade	1 1000101

- In den Bytes **mit** Paritätsbit: Anzahl Einsen immer gerade
- „Kippt“ ein Bit, fällt der Fehler auf
- Kippen zwei Bits, dann nicht...

- Sie haben folgende Nachricht (binär) erhalten: 01001000, 11100001, 11101100, 11101110, 01101111
- Prüfen Sie zunächst, welche Zeichen ohne „Bit-Kipper“ übertragen wurden
 - Für alle korrekt empfangenen schlagen Sie in der ASCII-Tabelle den zugehörigen Buchstaben nach,
 - für alle fehlerhaft empfangenen setzen Sie ein Fragezeichen ein.

65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g
72	H	104	h
73	I	105	i
74	J	106	j
75	K	107	k
76	L	108	l
77	M	109	m
78	N	110	n
79	O	111	o
80	P	112	p
81	Q	113	q
82	R	114	r
83	S	115	s
84	T	116	t
85	U	117	u
86	V	118	v
87	W	119	w
88	X	120	x
89	Y	121	y
90	Z	122	z

- ASCII bereits vorgestellt
- Problem: Regionale Zeichen (äöüßñš...) nicht in ASCII-Tabelle
- Erweiterungen der ASCII-Tabelle
 - 8-bittig: Zeichen 128-255 verwenden
 - 16-bittig: Platz für „alles“, inkl. Kyrillisch, Chinesisch, Japanisch etc.
 - „Mischform“ 8- und 16-bittig



```
testdatei.txt - /tmp/testdatei.txt
File Edit Search Preferences Shell Macro Windows Help
/tmp/testdatei.txt byte 0 of 147 L: 1 C: 0
1 Dieser Text enth€lt einige Umlaute und andere
2 sch€tne Sonderzeichen, etwa das €-Symbol (Euro).
3 Spa€ mit Sonderzeichen, gerne auch aus Espa±a!
4
5
```

Was ist hier passiert?

- Es gibt etliche Kodierungen für Textdateien
- Man muss wissen, in welcher Kodierung eine Datei gespeichert wurde
- Alternative: selbst-beschreibende Datei

```
<html>
<head>
<title>Eine Webseite</title>
<meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1">
...
```

Wichtige Kodierungen:

- **Latin-1** (iso-8859-1, ISO/IEC 8859-1, westeuropäisch)
- **Latin-9** (iso-8859-15, ISO/IEC 8859-15, westeuropäisch)
 - gegenüber Latin-1 mehr Buchstaben, weniger math. Symbole
 - u.a. € und Š, š, Ž, ž, Œ, œ, Ÿ

; ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯
 ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿
 À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï
 Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß
 à á â ã ä å æ ç è é ê ë ì í î ï
 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

Unicode

- Ziel: Alle Zeichen aus allen Schriftarten der Welt unterstützen
- bis zu 1.114.112 Zeichen in 17 Codebereichen („Planes“) mit je 65.536 (64K) Zeichen
- aktuell (Unicode 5.1): 100.713 Zeichen
- wichtigster Codebereich: **BMP** (Basic Multilingual Plane) mit allen aktuellen Schriftsystemen
- aber: keine 21-bittige Kodierung, sondern ...



Unicode-Formate / UTF-16, UTF-8

- UTF-8
 - ASCII-Zeichen (0–127) „normal“ (8-bittig) kodiert
 - andere Zeichen 16-, 24- oder 32-bittig (also über zwei, drei oder vier Bytes) kodiert
 - z. B. deutsche Umlaute: je zwei Bytes
- UTF-16
 - Zeichen der **BMP** 16-bittig (mit 2 Bytes) kodiert
 - sonstige Zeichen 32-bittig (mit 4 Bytes) kodiert

Beispiele: Umlaute ÄÖÜ, äöü, ß.
Euro: €. Britisches Pfund: £.
Französische «Anführungen»

(in Latin-9 erzeugt)

Latin-9-Kodierung:

00000000	42	65	69	73	70	69	65	6c	65	3a	20	55	6d	6c	61	75	Beispiele: Umlau
00000010	74	65	20	c4	d6	dc	2c	20	e4	f6	fc	2c	20	df	2e	0a	te ÄÖÜ, äöü, ß..
00000020	45	75	72	6f	3a	20	a4	2e	20	42	72	69	74	69	73	63	Euro: €. Britisc
00000030	68	65	73	20	50	66	75	6e	64	3a	20	a3	2e	0a	46	72	hes Pfund: £..Fr
00000040	61	6e	7a	f6	73	69	73	63	68	65	20	ab	41	6e	66	fc	anzösische «Anfü
00000050	68	72	75	6e	67	65	6e	bb									hrungen»

UTF-8-Kodierung:

00000000	42	65	69	73	70	69	65	6c	65	3a	20	55	6d	6c	61	75	Beispiele: Umlau
00000010	74	65	20	c3	84	c3	96	c3	9c	2c	20	c3	a4	c3	b6	c3	te Ã.Ã.Ã., Ã€Ã¶Ã
00000020	bc	2c	20	c3	9f	2e	0a	45	75	72	6f	3a	20	c2	a4	2e	€, Ã...Euro: Â€.
00000030	20	42	72	69	74	69	73	63	68	65	73	20	50	66	75	6e	Britisches Pfun
00000040	64	3a	20	c2	a3	2e	0a	46	72	61	6e	7a	c3	b6	73	69	d: Â£..FranzÃ¶si
00000050	73	63	68	65	20	c2	ab	41	6e	66	c3	bc	68	72	75	6e	sche Â«AnfÃ¶hrun
00000060	67	65	6e	c2	bb												genÂ»

UTF-16-Kodierung:

00000000	fe ff 00 42 00 65 00 69	00 73 00 70 00 69 00 65	pÿ.B.e.i.s.p.i.e
00000010	00 6c 00 65 00 3a 00 20	00 55 00 6d 00 6c 00 61	.l.e.:. .U.m.l.a
00000020	00 75 00 74 00 65 00 20	00 c4 00 d6 00 dc 00 2c	.u.t.e. .Ä.Ö.Ü.,
00000030	00 20 00 e4 00 f6 00 fc	00 2c 00 20 00 df 00 2e	. .ä.ö.ü.,. .ß..
00000040	00 0a 00 45 00 75 00 72	00 6f 00 3a 00 20 00 a4	...E.u.r.o.:. .€
00000050	00 2e 00 20 00 42 00 72	00 69 00 74 00 69 00 73	B.r.i.t.i.s
00000060	00 63 00 68 00 65 00 73	00 20 00 50 00 66 00 75	.c.h.e.s. .P.f.u
00000070	00 6e 00 64 00 3a 00 20	00 a3 00 2e 00 0a 00 46	.n.d.:. .£.....F
00000080	00 72 00 61 00 6e 00 7a	00 f6 00 73 00 69 00 73	.r.a.n.z.ö.s.i.s
00000090	00 63 00 68 00 65 00 20	00 ab 00 41 00 6e 00 66	.c.h.e. .«.A.n.f
000000a0	00 fc 00 68 00 72 00 75	00 6e 00 67 00 65 00 6e	.ü.h.r.u.n.g.e.n
000000b0	00 bb		.»

Dateigrößen:

test-latin9.txt	88 Bytes
test-utf8.txt	101 Bytes
test-utf16.txt	178 Bytes

- Neben reinen Textformaten (ASCII, ISO-Latin-*, UTF):
Textformate mit Auszeichnungen (Markup)
 - HTML (**H**yper**T**ext **M**arkup **L**anguage)
(Webseiten)
 - LaTeX (Textsatz-Auszeichnungssprache)
 - SGML (**S**tandard **G**eneralized **M**arkup **L**anguage)
 - XML (**E**xtensible **M**arkup **L**anguage)
(Grundlage für viele Markup-Sprachen)
 - Wikitext (Texte in Wikis)

- Markup-Sprachen ergänzen den reinen Text mit Markup-Elementen
- Zweierlei Markup möglich:
 - „Grafisches“ Markup (fett, kursiv, Schriftgröße)
 - Struktur-Markup (Kapitelanfang, Zitat)
 - Kombination (mit Stylesheets)
- Markup unabhängig von Kodierung der Textinhalte; oft beliebige Kodierung möglich

Beispiele: HTML, LaTeX, Wikitext

Beispiel für	HTML	LaTeX	Wikitext
Überschrift	<code><h1>Überschrift</h1></code>	<code>\section{Überschrift}</code>	<code>= Überschrift =</code>
Aufzählung	<pre> Punkt 1 Punkt 2 Punkt 3 </pre>	<pre> \begin{itemize} \item Punkt 1 \item Punkt 2 \item Punkt 3 \end{itemize} </pre>	<pre> * Punkt 1 * Punkt 2 * Punkt 3 </pre>
fetten Text	<code>fett</code>	<code>\textbf{fett}</code>	<code>'''fett'''</code>
kursiven Text	<code><i>kursiv</i></code>	<code>\textit{kursiv}</code>	<code>''kursiv''</code>

Quelle: Wikipedia, <http://de.wikipedia.org/wiki/Auszeichnungssprache#Beispiele>

- Grober Aufbau eines HTML-Dokuments:

```
<html>
  <head>
    <title>Titel des Dokuments</title>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Ein Absatz Text reicht für das Beispiel</p>
  </body>
</html>
```

- unterteilt in Head und Body
- öffnende und schließende Tags <xyz>, </xyz>

- < und > sind Kontrollzeichen, die ein Tag einleiten / beenden
- Um „<“ oder „>“ im Text zu verwenden:
 - < (less than) „<“
 - > (greater than) „>“
 - Beispiel:
Das Tag **<html>** → Das Tag <html>
- Sonderzeichen: ä (ä), Ä (Ä) etc., ß (ß, „S-Z-Ligatur“); € (€)

- Cascading Stylesheets (CSS):
 - trennen die inhaltliche von der grafischen Auszeichnung
 - Texte verwenden Tags mit Klassenangabe
 - Stylesheet enthält Layout-Definition für das Tag und die Klasse
- Position des Stylesheets
 - wahlweise direkt in der HTML-Datei
 - oder in separater css-Datei

- Ziel: Drei Absätze in verschiedenen Zeichenformatierungen
 - Arial, 20 pt
 - Courier, 15 pt
 - Times, 10 pt

Das ist ein Stylesheet-Test. (Arial 20pt)

Das ist ein Stylesheet-Test. (Courier 15pt)

Das ist ein Stylesheet-Test. (Times 10pt)

- Formatierung ohne CSS, direkt mit `style=...`

```
<html>
<head><title>Test ohne CSS-Datei</title></head>
<body>
```

```
<p style="font-family: arial;
        font-size: 20pt">
```

```
Das ist ein Stylesheet-Test. (Arial 20pt)
</p>
```

```
<p style="font-family: courier;
        font-size: 15pt">
```

```
Das ist ein Stylesheet-Test. (Courier 15pt)
</p>
```

```
<p style="font-family: times;
        font-size: 10pt">
```

```
Das ist ein Stylesheet-Test. (Times 10pt)
</p>
```

- Formatierung mit CSS-Datei

```
<html>
<head><title>Test mit CSS-Datei</title>
<link rel="stylesheet" type="text/css"
      href="/stylesheet.css"> </head>
<body>

<p class="arial20">
Das ist ein Stylesheet-Test. (Arial 20pt)
</p>

<p class="courier15">
Das ist ein Stylesheet-Test. (Courier 15pt)
</p>

<p class="times10">
Das ist ein Stylesheet-Test. (Times 10pt)
</p>
```

Stylesheet-Datei:

```
p.arial20 {
    font-family: arial;
    font-size: 20pt;
}

p.courier15 {
    font-family: courier;
    font-size: 15pt;
}

p.times10 {
    font-family: times;
    font-size: 10pt;
}
```

- Aufbau der Einträge in der CSS-Datei

Es geht um das `<p>`-Attribut

in einer Klasse `arial20`

`p.arial20 {`

`font-family: arial;`

`font-size: 20pt;`

`}`

Das sind die
Eigenschaften

- Hier keine vollständige Einführung in HTML und CSS
- Gute Dokumentation im Web: SelfHTML
<http://de.selfhtml.org/>

- TeX (sprich: „Tech“) ist eine Dokumentenbeschreibungssprache, LaTeX ein Makropaket für TeX. Man benutzt meist LaTeX
- Auszeichnung von Kapitelüberschriften, Zitaten, ... über LaTeX-Befehle, z. B.
`\section{Abschnitt},`
`\begin{quotation} ... \end{quotation}`

- kleines Beispieldokument

```
\documentclass{article}  
\begin{document}
```

```
\tableofcontents    % erzeuge Inhaltsverzeichnis
```

```
\section{Erster Abschnitt}
```

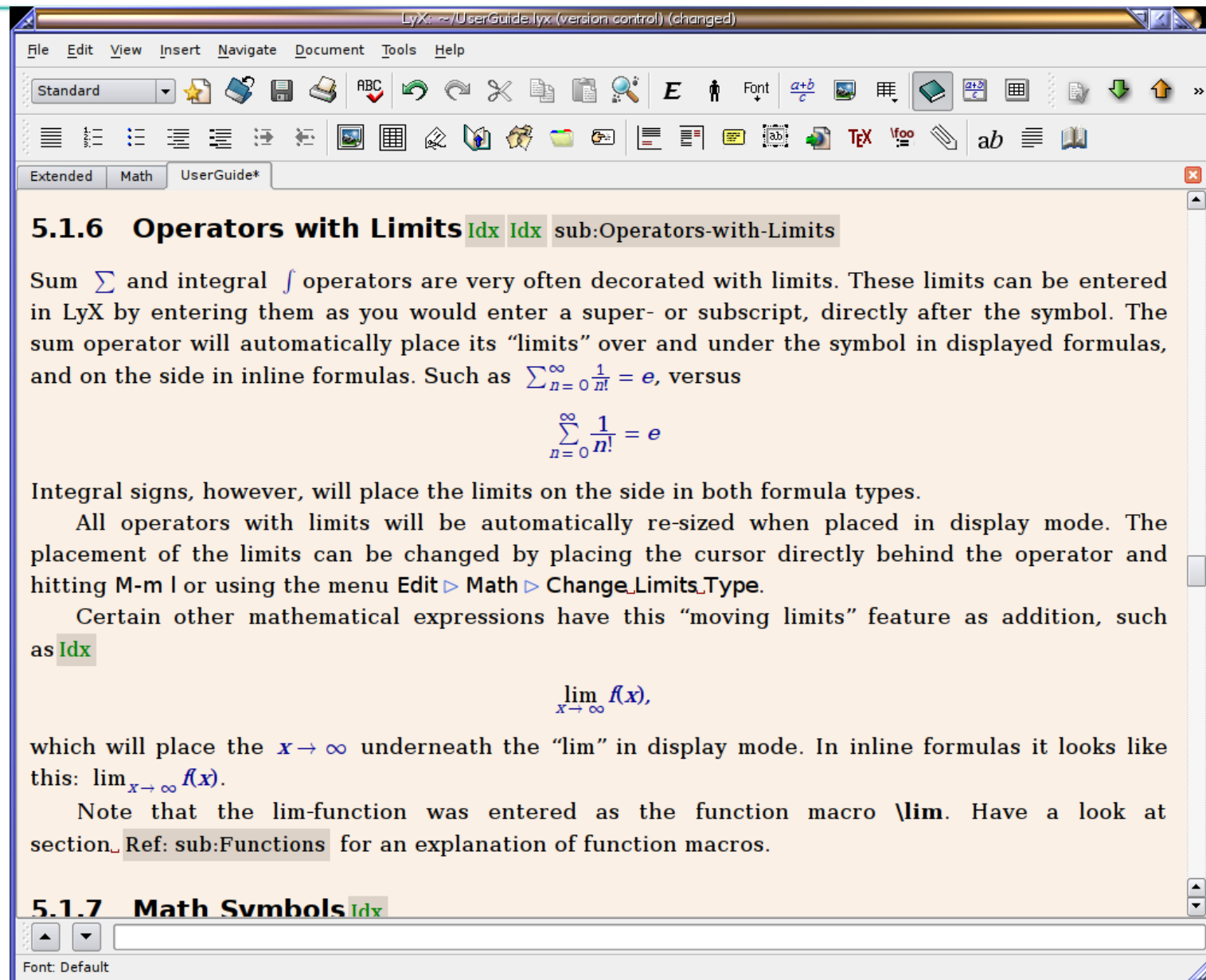
Hallo, das ist ein Beispieltext mit einem
\emph{hervorgehobenen} und einem \textbf{fett}
geschriebenen Wort.

```
\end{document}
```

- TeX/LaTeX selbst arbeitet wie ein Compiler und erzeugt aus den Eingabedateien (*.tex) PDF-Dateien
- Wie bei Programmiersprachen: Syntax- und komplexere Fehler möglich, die zum Abbruch führen können
- LaTeX kann aber auch unabhängig vom LaTeX-Compiler als eigenständiges Datenformat verstanden werden → Converter *latex2html*

LyX (LaTeX-GUI)

Quelle Bild:
<http://www.lyx.org>



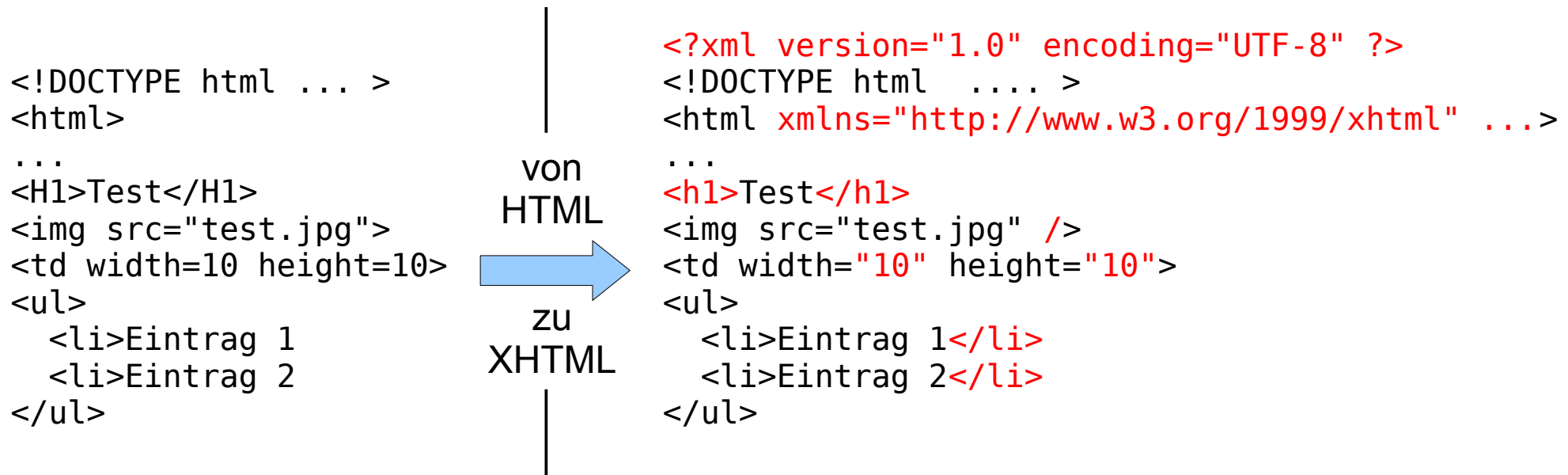
- **HTML:** Markup-Sprache mit fest vorgegebenen Tags (z. B. <head>, , <p>, <table> etc.)
- **XML:** Extensible Markup Language
- **XML** erlaubt es, eigene Markup-Sprachen mit HTML-ähnlicher Syntax zu definieren
- unterscheiden zwischen
 - Dokument, das die Regeln für die eigene Sprache definiert (→ **DTD**, Document Type Definition)
 - Dokument, das nach diesen Regeln erstellt (ausgezeichnet) wurde

- Eine DTD ist nicht zwingend nötig – wenn es eine gibt, erlaubt sie das **Validieren** von XML-Dokumenten
- Ohne DTD nur allgemeine Syntaxprüfung möglich

Syntaxfehler (nicht „wohlgeformt“)	DTD-Fehler (nicht „gültig“)	alles korrekt
<pre><person> <name>Müller</name> <ort>München</tor> <tel v="0123456" /> </person></pre>	<pre><person> <wer>Müller</wer> <wo>München</wo> <tel v="0123456" /> </person></pre>	<pre><person> <name>Müller</name> <ort>München</ort> <tel v="0123456" /> </person></pre>
<pre><person> <name>Meier</name> <ort>Augsburg</ort> <tel v="0987641"> </person></pre>	<pre><leute> <name>Meier</name> <ort>Augsburg</ort> <tel v="0123456" /> </leute></pre>	<pre><person> <name>Meier</name> <ort>Augsburg</ort> <tel v="0987641" /> </person></pre>
<pre><person> <name>Huber <ort>Erlangen </name> </ort> <tel v="0448"></tel> </person></pre>	<pre><person> ! <ort>Erlangen</ort> <tel v="0448" /> </person></pre> (im letzten Eintrag fehlt ein vorgeschriebenes <name> Element)	<pre><person> <name>Huber</name> <ort>Erlangen</ort> <tel v="0448" /> </person></pre>

- XML erlaubt Transformation von Dokumenten in andere Darstellungen
- **XSLT**: XSL Transformation (**XSL** = Extensible Stylesheet Language)
- Syntax recht komplex; siehe http://de.wikipedia.org/wiki/XSL_Transformation

- XHTML: Neuformulierung von HTML in XML
- Abweichungen minimal, z. B.:
 - alle Tags müssen geschlossen werden
 - Attribute immer in Anführungen
 - nur Kleinschreibung



- **DocBook:** XML-basiertes Format für komplexe Dokumente, z. B.
 - Bücher
 - technische Dokumentation

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN" ...>
<book lang="de">
  <bookinfo>
    <title>
      Ein kurzes Buch
    </title>
  </bookinfo>
  <chapter>
    <title>
      Das erste Kapitel
    </title>
    <para>
      Ein einziger Absatz füllt das erste Kapitel.
    </para>
  </chapter>
</book>
```

- SGML: **S**tandard **G**eneralized **M**arkup Language
- gleicher Ansatz wie bei XML, erlaubt aber noch komplexere Dinge, z. B. Unterdokumente
- genauer:

„The Extensible Markup Language (XML) is a subset of SGML [...]. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.“

Quelle:
<http://www.w3.org/TR/2006/REC-xml-20060816/>

- Auch Word- und LibreOffice-Textdateien bestehen aus Inhalt + Formatierung
- Alte Versionen von Word und OpenOffice (LibreOffice-Vorläufer) verwenden Binärformate zum Speichern der Informationen (Dateiendungen: **.doc*, **.sdw*)
- Neue Versionen erzeugen Zip-Archive, die XML-Dateien enthalten (Dateiendungen: **.docx*, **.odt*)

Binäre Textdokumente (Word & Co.) ...

- ... sind (von Anwendern) nicht lesbar
- ... können von Fremdprogrammen nur mit Hilfe von Importfiltern gelesen werden, wenn der Aufbau dokumentiert ist
- ... sind schlimmstenfalls einfache „Memory Dumps“ der Anwendungen

```
$ hexdump -C linux.doc
```

```
00000000 d0 cf 11 e0 a1 b1 1a e1 00 00 00 00 00 00 00 00 |ÐÏ.ài±.á.....|
00000010 00 00 00 00 00 00 00 00 3e 00 03 00 fe ff 09 00 |.....>...þÿ..|
00000020 06 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 |.....|
00000030 cb 01 00 00 00 00 00 00 00 10 00 00 cd 01 00 00 |Ë.....Í...|
00000040 01 00 00 00 fe ff ff ff 00 00 00 00 c7 01 00 00 |....þÿÿÿ....Ç...|
00000050 c8 01 00 00 c9 01 00 00 ca 01 00 00 ff ff ff ff |È...É...Ê...ÿÿÿÿ|
[...]
```

```
00034110 0b 00 00 00 4e 6f 72 6d 61 6c 2e 64 6f 74 00 20 |....Normal.dot. |
00034120 1e 00 00 00 11 00 00 00 20 48 61 6e 73 2d 47 65 |..... Hans-Ge|
00034130 6f 72 67 20 45 df 65 72 00 00 64 00 1e 00 00 00 |org Eßer..d....|
00034140 02 00 00 00 31 00 61 6e 1e 00 00 00 13 00 00 00 |....1.an.....|
00034150 4d 69 63 72 6f 73 6f 66 74 20 57 6f 72 64 20 39 |Microsoft Word 9|
```

```
$ strings -e S linux.doc
```

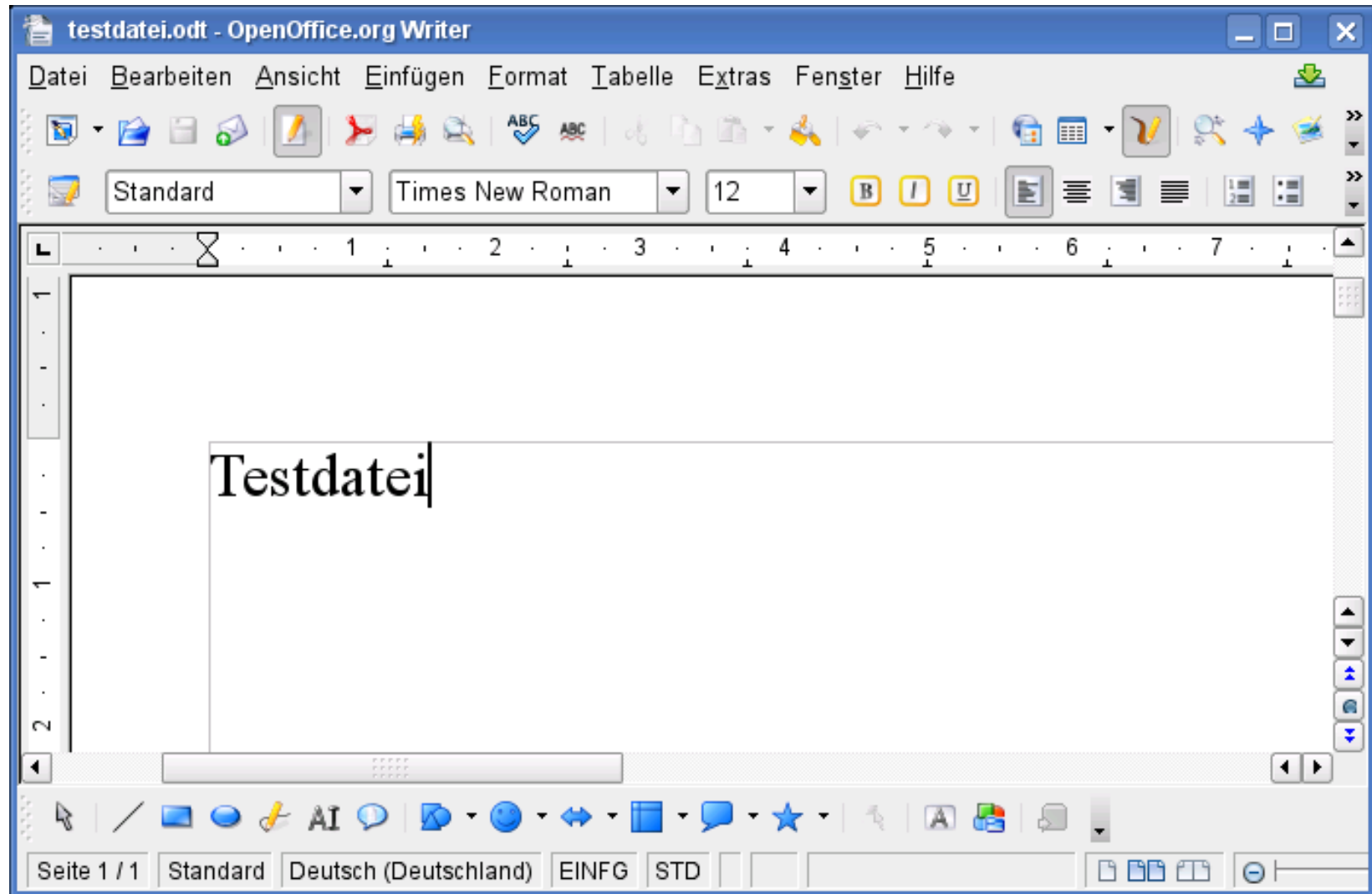
```
Linux
zum Nachschlagen
SuSE Linux 8.x
Hans-Georg Eßer
Inhalt
  TOC \o "1-3" \h \z
  HYPERLINK \l "_Toc13267047"
Inhalt
  PAGEREf _Toc13267047 \h
  HYPERLINK \l "_Toc13267048"
```

```
Vorwort
  PAGEREf _Toc13267048 \h
  HYPERLINK \l "_Toc13267049"
[...]

Normal.dot
  Hans-Georg Eßer
Microsoft Word 9.0
[...]
```

- Neuere Versionen (Word: **.docx*; OOO: **.odt*) speichern im XML-Format
- Vorteile:
 - einfacheres Einlesen / Bearbeiten durch Fremdprogramme
 - ermöglicht automatisches Generieren durch andere Tools, etwa auf einer Webseite ein Button „Seite als *docx*-Datei speichern“

XML-Format: LibreOffice (1/6)



```
$ unzip -l /tmp/testdatei.odt
```

```
Archive:  /tmp/testdatei.odt
```

Length	Date	Time	Name
39	09-09-10	19:01	mimetype
0	09-09-10	19:01	Configurations2/statusbar/
0	09-09-10	19:01	Configurations2/accelerator/current.xml
0	09-09-10	19:01	Configurations2/floater/
0	09-09-10	19:01	Configurations2/popupmenu/
0	09-09-10	19:01	Configurations2/progressbar/
0	09-09-10	19:01	Configurations2/menubar/
0	09-09-10	19:01	Configurations2/toolbar/
0	09-09-10	19:01	Configurations2/images/Bitmaps/
2683	09-09-10	19:01	content.xml
532	09-09-10	19:01	manifest.rdf
10302	09-09-10	19:01	styles.xml
1097	09-09-10	19:01	meta.xml
725	09-09-10	19:01	Thumbnails/thumbnail.png
8387	09-09-10	19:01	settings.xml
1989	09-09-10	19:01	META-INF/manifest.xml
25754			16 files

Inhalt von *content.xml*:

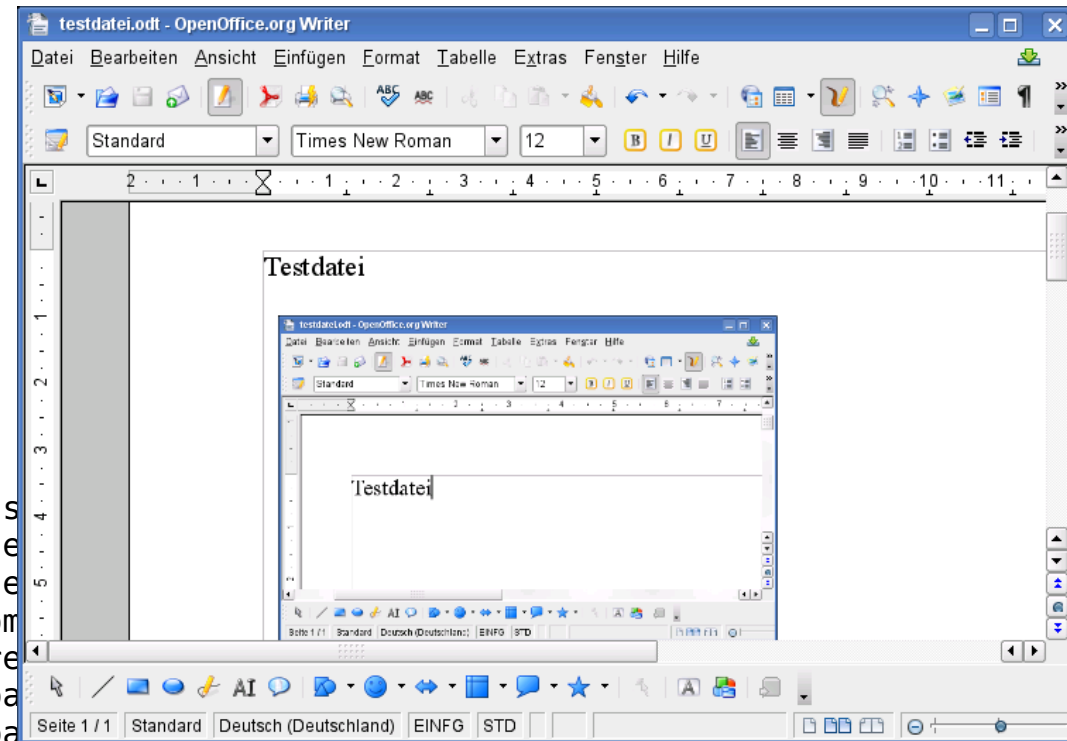
```
<office:document-content xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0" xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0" xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0" xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0" xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:forms:1.0" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0" xmlns:number="urn:oasis:names:tc:opendocument:xmlns:number:1.0" xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0" xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0" xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0" xmlns:math="http://www.w3.org/1998/Math/MathML" xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0" xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0" xmlns:ooo="http://openoffice.org/2004/office" xmlns:ooow="http://openoffice.org/2004/writer" xmlns:oooc="http://openoffice.org/2004/calc" xmlns:dom="http://www.w3.org/2001/xml-events" xmlns:xforms="http://www.w3.org/2002/xforms" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:rpt="http://openoffice.org/2005/report" xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:grddl="http://www.w3.org/2003/g/data-view#" xmlns:field="urn:openoffice:names:experimental:ooo-ms-interop" office:version="1.2" grddl:transformation="http://docs.oasis-open.org/office/1.2/xslt/odf2rdf.xsl">
  <office:scripts/>
  <office:font-face-decls>
    <style:font-face style:name="Times New Roman" svg:font-family="'Times New Roman'" style:font-family-generic="roman" style:font-pitch="variable"/>
    <style:font-face style:name="Arial" svg:font-family="Arial" style:font-family-generic="swiss" style:font-pitch="variable"/>
    <style:font-face style:name="Arial1" svg:font-family="Arial" style:font-family-generic="system" style:font-pitch="variable"/>
  </office:font-face-decls>
  <office:automatic-styles/>
  <office:body>
    <office:text>
      <text:sequence-decls>
        <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
      </text:sequence-decls>
      <text:p text:style-name="Standard">
        Testdatei
      </text:p>
    </office:text>
  </office:body>
</office:document-content>
```

Jetzt eine Datei mit Bild:

```
$ unzip -l /tmp/testdatei.odt
```

```
Archive: /tmp/testdatei.odt
```

Length	Date	Time	Name
39	09-09-10	19:14	mimetype
0	09-09-10	19:14	Configurations2/status
0	09-09-10	19:14	Configurations2/accele
0	09-09-10	19:14	Configurations2/floate
0	09-09-10	19:14	Configurations2/popup
0	09-09-10	19:14	Configurations2/progre
0	09-09-10	19:14	Configurations2/menuba
0	09-09-10	19:14	Configurations2/toolba
0	09-09-10	19:14	Configurations2/images/Bitmaps/
37829	09-09-10	19:14	Pictures/10000000000002BB000001C9C86EB88C.png
3599	09-09-10	19:14	content.xml
532	09-09-10	19:14	manifest.rdf
10591	09-09-10	19:14	styles.xml
1097	09-09-10	19:14	meta.xml
4718	09-09-10	19:14	Thumbnails/thumbnail.png
8914	09-09-10	19:14	settings.xml
2190	09-09-10	19:14	META-INF/manifest.xml
69509			17 files



Inhalt von *content.xml* (Auszug):

```
<office:body>
  <office:text>
    <text:sequence-decls>
      <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
    </text:sequence-decls>
    <text:p text:style-name="Standard">
      Testdatei
    </text:p>
    <text:p text:style-name="Standard"/>
    <text:p text:style-name="Standard">
      <draw:frame draw:style-name="fr1" draw:name="Grafik1"
        text:anchor-type="paragraph" svg:x="0.235cm" svg:y="0cm"
        svg:width="7.491cm" svg:height="4.898cm" draw:z-index="0">
        <draw:image xlink:href="Pictures/100000000000002BB000001C9C86EB88C.png"
          xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad"/>
        </draw:frame>
      </text:p>
    </office:text>
  </office:body>
```

Inhalt von *meta.xml* (Metadaten):

```
<office:document-meta office:version="1.2" grddl:transformation=
"http://docs.oasis-open.org/office/1.2/xslt/odf2rdf.xsl">
  <office:meta>
    <meta:initial-creator>Hans-Georg Eßer</meta:initial-creator>
    <meta:creation-date>2010-09-09T21:01:27</meta:creation-date>
    <dc:date>2010-09-09T21:14:40</dc:date>
    <dc:creator>Hans-Georg Eßer</dc:creator>
    <meta:editing-duration>PT00H13M05S</meta:editing-duration>
    <meta:editing-cycles>2</meta:editing-cycles>
    <meta:generator>
      OpenOffice.org/3.2$Linux OpenOffice.org_project/320m12$Build-9483
    </meta:generator>
    <meta:document-statistic meta:table-count="0" meta:image-count="1"
      meta:object-count="0" meta:page-count="1" meta:paragraph-count="1"
      meta:word-count="1" meta:character-count="9"/>
  </office:meta>
</office:document-meta>
```

- Zwei Welten bei Grafikformaten:
 - Rastergrafiken (jpg, png, tif, bmp, gif, xpm)
 - Vektorgrafiken (eps, emf, svg, ps)
 - Gigantische Liste:
<http://de.wikipedia.org/wiki/Grafikformat>

- pixel-basiert
- Größe einer Grafik abhängig von
 - Breite x Höhe in Pixeln
 - Farbtiefe (8 Bit, 16 Bit pro RGB-Farbkanal etc.)
 - Kompressionsverfahren

- Rechenbeispiel

- Bild: 800 x 600 Punkte
- RGB, 8 Bit pro Farbkanal
(also insgesamt 24 Bit, 24 bpp: bits per pixel)
- $800 \times 600 \times 24 = 11.520.000 \text{ Bit} = 1.440.000 \text{ Byte}$

```
$ identify test.bmp
```

```
test.bmp BMP 800x600 8-bit DirectClass 1.831mb
```

```
$ convert test.bmp test.rgb
```

```
$ ls -l test.*
```

```
-rw-r--r-- 1 esser users 1920122 10. Sep 10:37 test.bmp
-rw-r--r-- 1 esser users 1440000 10. Sep 10:41 test.rgb
```

- Primitiv-Format: XPM
 - **X Pixmap**
 - vor allem für Icons
 - „lesbar“
 - nicht komprimiert
- Beispiel (48x20 Pixel):

[illegible]

- Anderes Beispiel:
FOM-Logo
(Ausschnitt)



```

/* XPM */
static char *fom_s[] = {
/* columns rows colors chars-per-pixel */
"55 60 256 2",
"   c black",
".   c #010101",
"X   c #020202",
[...]
"PX c #FDFDFD",
"IX c #FEFEFE",
"UX c gray100",
/* pixels */

```

[illegible]

- „normale“ Rasterformate sind binär und **komprimieren** die Daten
- „Pixelgrafiken“ sind Rastergrafiken
- zwei Arten von Kompression
 - verlustfrei (z. B. bei png, tif)
→ Originalbild wiederherstellbar
 - verlustbehaftet (z. B. bei jpg)
→ Teile der Bildinformation gehen verloren

- **JPEG**-Format (entwickelt von der **J**oint **P**hotographic **E**xperts **G**roup)
- verlustbehaftete Kompression
- Qualitätsstufen



fom.png
(Original)
4575



fom-85.jpg
(85% Qual.)
1988



fom-70.jpg
(70% Qual.)
1540



fom-50.jpg
(50% Qual.)
1302



fom-25.jpg
(25% Qual.)
985



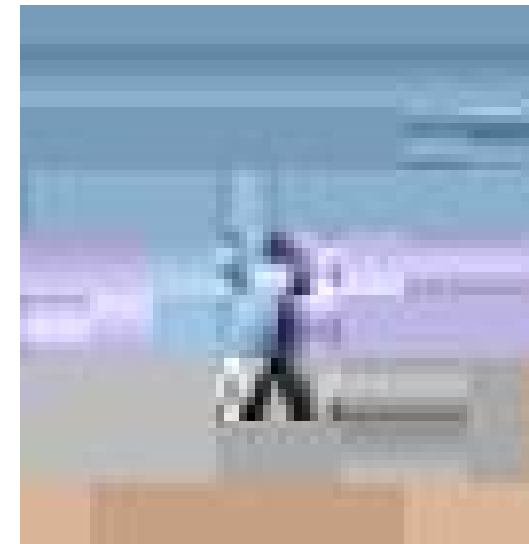
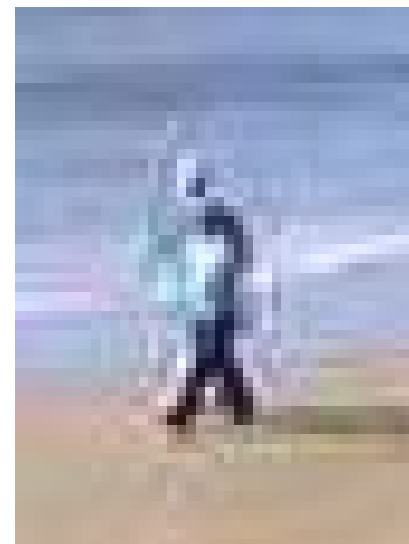
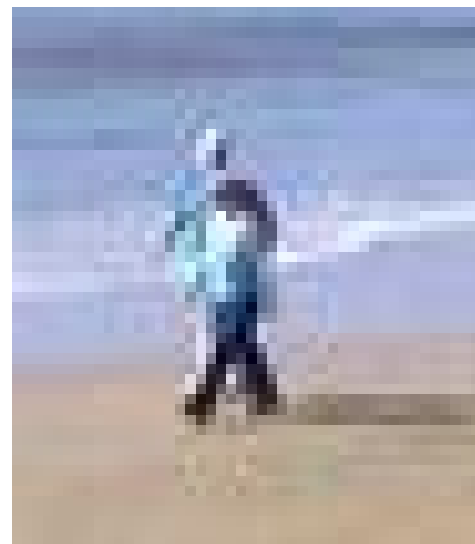
fom-05.jpg
(5% Qual.)
553



fom-00.jpg
(0% Qual.)
461

JPEG-Format / *.jpg (2/2)

- am besten für Fotos geeignet



Original
392.190 Bytes

JPEG, 85 %
107.748 Bytes

JPEG, 30 %
36.383 Bytes

JPEG, 5 %
8.632 Bytes

- **PNG: Portable Network Graphics**
- als Ersatz für das *gif*-Format vor allem für Grafiken auf Webseiten eingesetzt
- patentfrei
- verlustfreie Komprimierung
 - viel besser als bei *gif*-Dateien

- **TIFF: Tagged Image File Format**
- verlustfreie Kompression
(außer in einem Spezialformat → JPEG)
- Standardformat im DTP-Bereich: unterstützt
 - RGB-Format (**R**ed, **G**reen, **B**lue)
 - CMYK-Format
(**C**yan, **M**agenta, **Y**ellow, **K**ey / ~~Black~~)

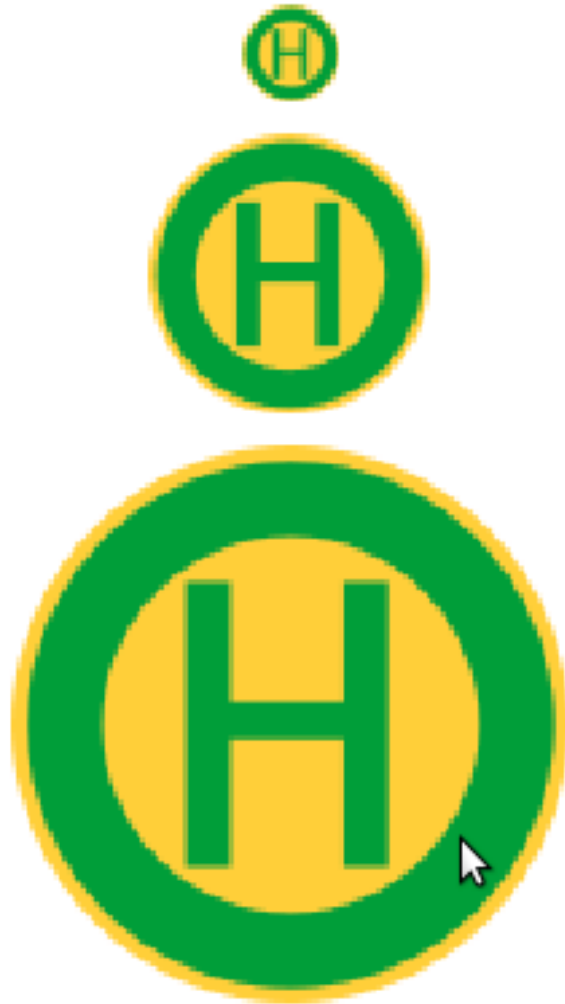
Welche Kompression?

- Entscheidung über Bildformat (und damit: über Kompression) hängt von Bildtyp ab
 - Fotos:
große Vielfalt an Farbtönen, weiche Übergänge
→ JPEG
 - Diagramme, Grafiken, ins Rasterformat gewandelte Vektorgrafiken:
geringe Farbzahl, harte Kanten
→ PNG, TIFF etc.
- Falsches Format zu wählen, bedeutet:
 - zu hoher Platzbedarf oder
 - zu geringe Qualität

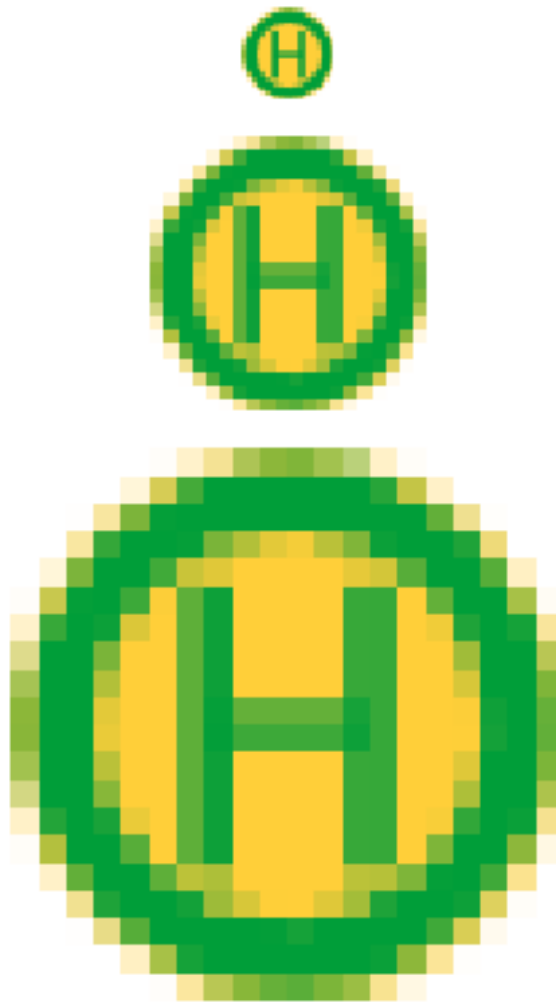
- Idee: Bilder nicht in Pixel zerlegen, sondern „beschreiben“, z. B.

Bild mit Abmessungen 10 cm x 10 cm, weißer Hintergrund, roter Kreis mit Durchmesser 1 cm und Mittelpunkt bei Koordinaten (3 cm, 4 cm). Rand des Kreises blau, Dicke 1 mm

- allgemein: grafische „Primitive“ (Linie, Kreis, Rechteck etc., Pfad, Polygon, Text)
- Beispielformate, SVG, PS (PostScript), EPS
- Vorteil: Beliebige Skalierbarkeit



Vektorgrafik



Rastergrafik

Quelle:
<http://de.wikipedia.org/wiki/Vektorgrafik>

SVG: Scalable Vector Graphics

- Freies Format (patentfrei)
- ein weiteres XML-Format:

```

<circle
  cx="1275"
  cy="1511"
  r="708"
  style="stroke:#000000;
        stroke-width:8;"
/>

```

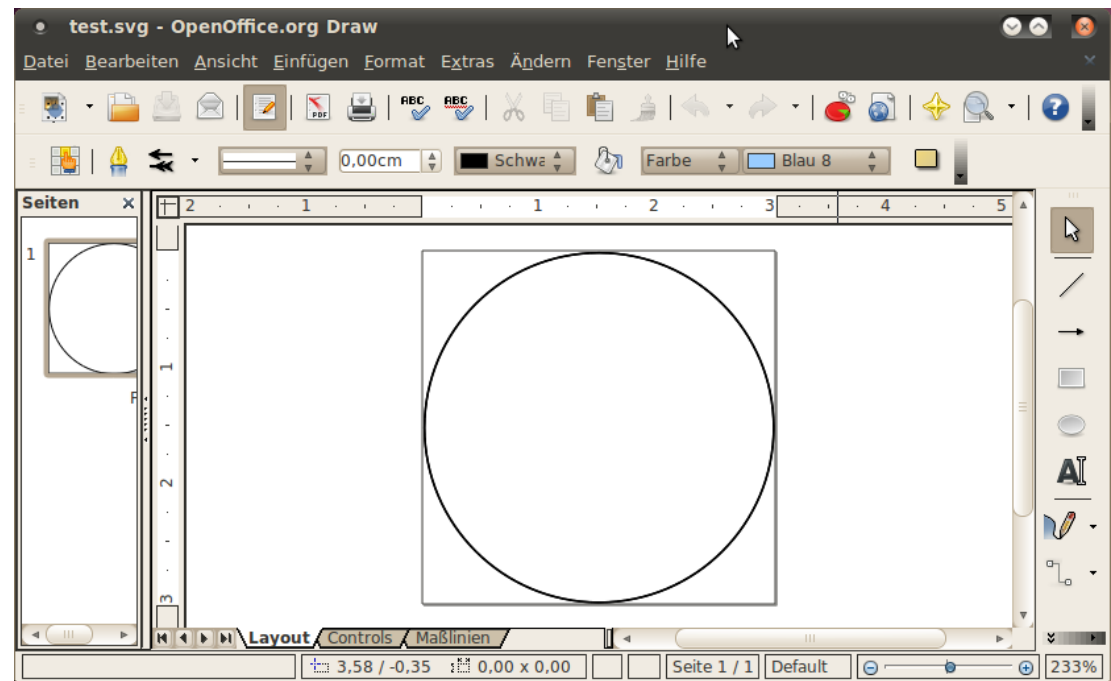
X- und Y-Koordinaten
des Mittelpunkts

Kreisradius

Stilangaben: Farbe, Breite

SVG-Format / *.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="1.2in"
height="1.2in" viewBox="558 794 1434 1434">
  <g style="stroke-width:.025in; fill:none">
    <!-- Circle -->
    <circle cx="1275" cy="1511" r="708"
      style="stroke:#000000;stroke-width:8;"/>
  </g>
</svg>
```

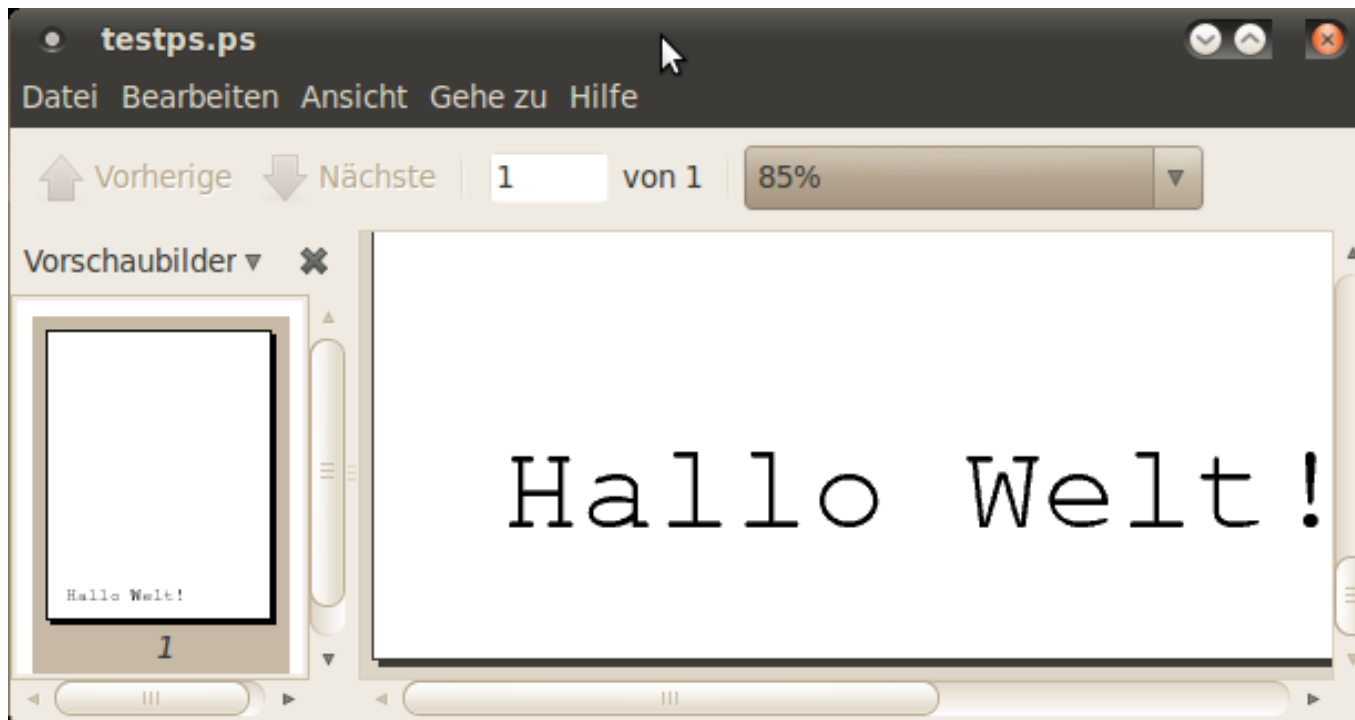


- altes Format: 1984 (Adobe)
- ursprünglich: Seitenbeschreibungssprache (für Laserdrucker und Druckmaschinen)
- wird heute durch PDF verdrängt
- Linux setzt (noch) in seinem Drucksystem PostScript ein
- Variante: **EPS** (**E**ncapsulated **P**ost**S**cript)
- kann auch Rastergrafik-Elemente enthalten
- PostScript ist eine Programmiersprache

PostScript-Format / *.ps (2/2)

```
%!  
/Courier findfont      % Schrift auswählen  
50 scalefont          % auf Schriftgröße 50 skalieren  
setfont               % zum aktuellen Zeichensatz machen  
50 50 moveto          % (50, 50) als aktuelle Schreibposition setzen  
(Hallo Welt!) show    % und dort den Text ausgeben  
showpage              % Seite ausgeben
```

(Quelle Listing: <http://de.wikipedia.org/wiki/PostScript>)

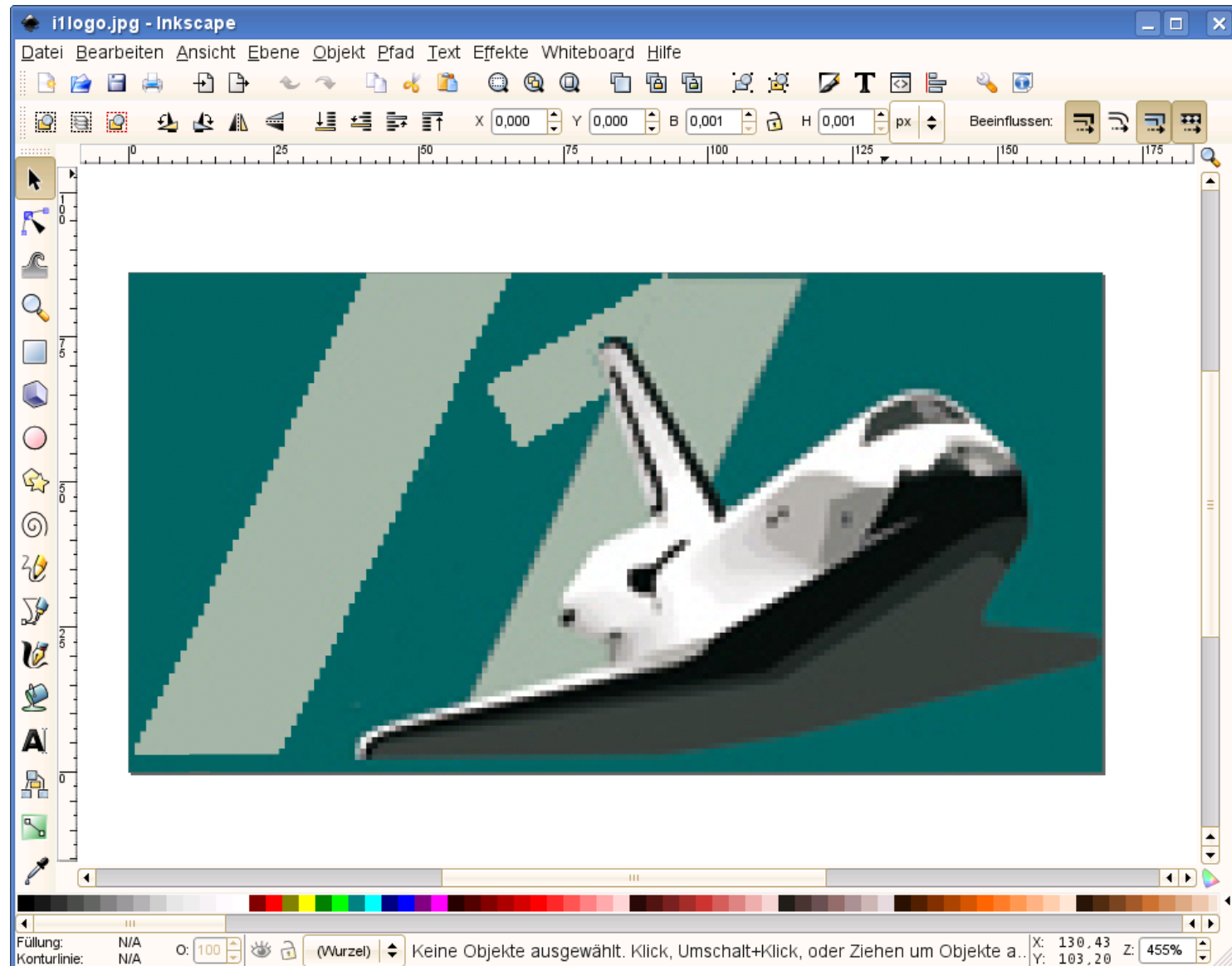


- PDF: Portable Document Format
- 1993 (Adobe), seit 2008 „offener Standard“
- Seitenbeschreibungssprache
- ähnliches Datenmodell wie PostScript, mit komprimierter Kodierung
- einbetten von Schriftarten
- Hyperlinks (intern & extern)
- Zugriffsschutz (Verschlüsselung/Passwort, Copy & Paste, Drucken)

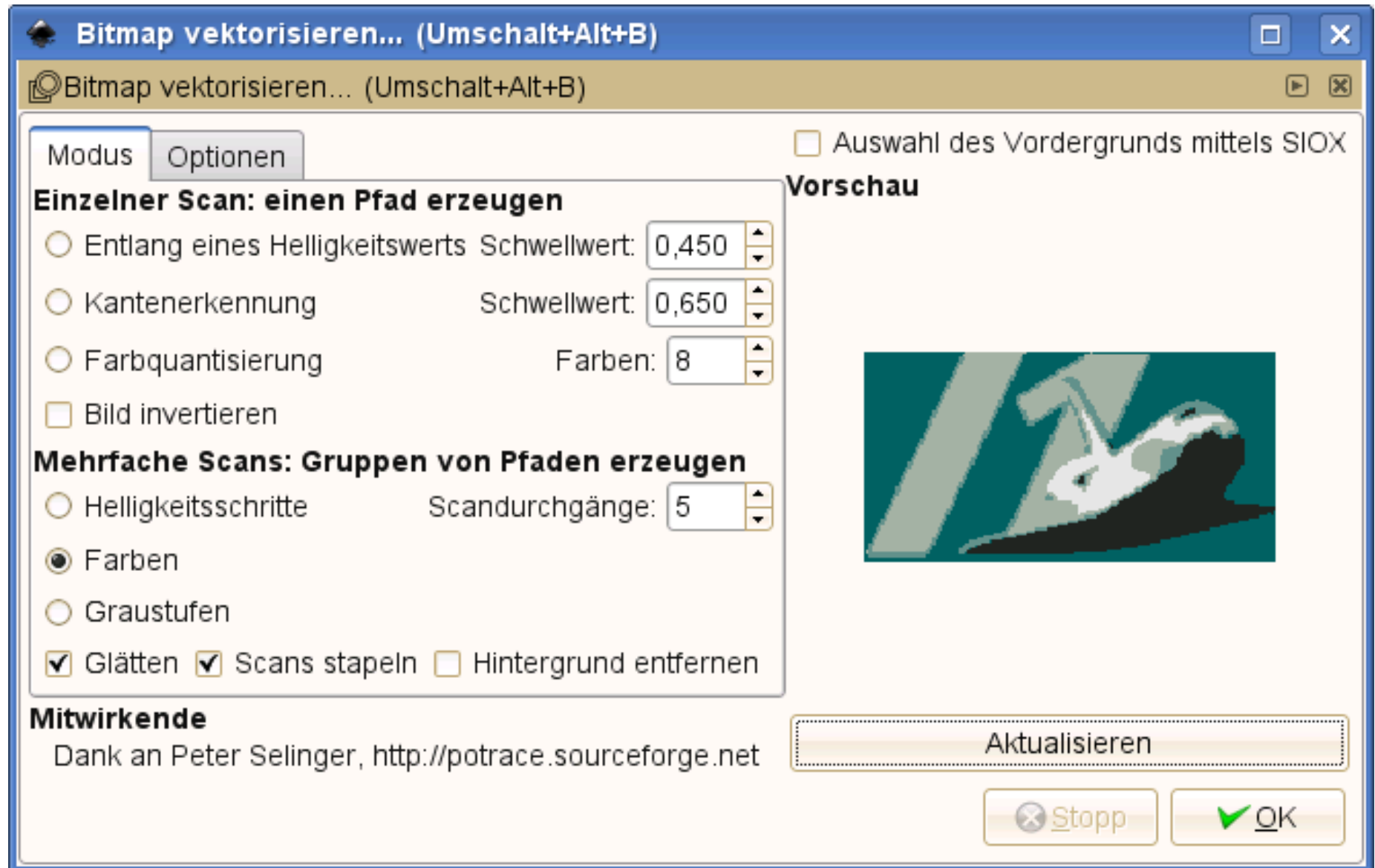
- interaktiv: Lesezeichen, Formulare, Kommentare (im Reader)
- unsichtbare Textebenen (für gescannte und mit OCR analysierte Dokumente)
- eingeschränkt: Nacharbeiten an PDF-Datei möglich
- einfache Konvertierung PDF ↔ PostScript
- PDF heute ein Standardformat für Dokumentenweitergabe und Druckdaten
- PDF/A: PDF Archive, für Langzeitarchivierung

Vektorisieren (1/3)

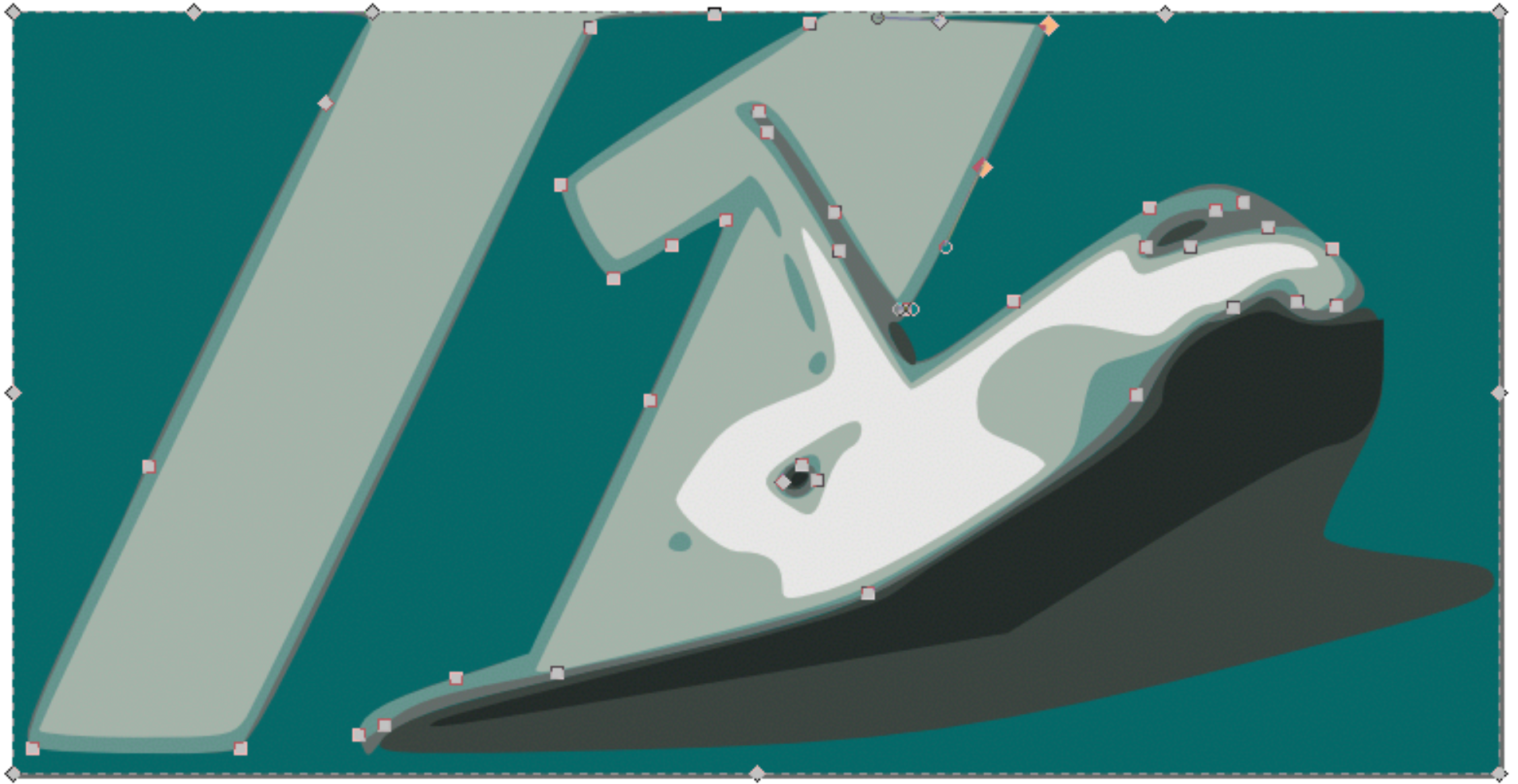
Inkscape
(www.inkscape.org) kann
Bitmap-
Grafiken
vektorisieren



Vektorisieren (2/3)



Ergebnis:



Aktuelle Acrobat-Version hat interessantes Feature „ClearType“:

- Dokument einscannen und als PDF speichern
- OCR auf PDF-Datei anwenden
- neue Fonts (Schriftartdateien) aus erkannten Buchstaben erzeugen
- Pixel-Buchstaben in PDF-Datei durch Buchstaben in diesen Fonts ersetzen

- „Video“ = Audio + Video
- Bilder (Frames), pro Frame ein „volles Bild“
- simpelste Variante: MJPEG (Motion JPEG)
- separates Speichern von Ton und Bild – oder „gemischt“

- **MPEG: Moving Picture Experts Group**
- MPEG-1: erstes Videoformat der MPEG
- verschiedene Frame-Typen:
 - I-Frame: intra coded picture. Standbild (wie MJPEG)
 - P-Frame: predictive coded picture. Bild hängt von vorherigen Bildern ab
 - B-Frame: bidirectional coded picture. Bild hängt von vorherigen und folgenden (!) Bildern ab
 - D-Frame
- Audio: Audio Layer 1–3 (MP1–MP3)

- Codec = **C**oder/**D**ecoder
- viele Container-Formate (z. B. AVI), die Video- und Audio-Streams in diversen Kodierungen aufnehmen
- abspielen eines Videos:
 - Analyse des Container-Formats
 - Lesen der Metadaten (welche Codecs?)
 - suchen der Decoder für Audio und Video
 - Streams dekodieren (Wiedergabe)

- Ziel: Daten aus einem Format in ein anderes umwandeln
- Beispiele
 - UTF-8-Text → ISO-Latin9-Text
 - PNG-Bild → JPEG-Bild
 - MP3-Audiodatei → Ogg-Vorbis-Datei
 - Word-Dokument → LibreOffice-Dokument
 - PostScript → PDF
 - PNG-Bild → UTF-8-Text (sinnvoll?)
 - MP3-Audiodatei → UTF-8-Text (sinnvoll?)

Probleme

- Quell- oder Zielformat evtl. nicht vollständig bekannt (→ proprietäre Dateiformate)
- Quell- oder Zielformat evtl. nicht gleich „mächtig“
- Verlust durch Kompression
- Verlust von Struktur (z. B. DocBook → PDF)

Import- und Export-Filter

- Öffnen- und Speichern-Funktion von Anwendungen mit Import-/Export-Feature
- Qualität hängt von Güte der Filter ab

- Verschiedene Modelle
 - Nachricht: Sender → Empfänger
 - Dialog
 - Gruppenkommunikation
- Klartext vs. „binäre“ Nachrichten

- Format/Standard für Kommunikation
 - Netzwerkprotokolle
- Dateiformat für den Austausch von Informationen
 - PDF für Dokumente
 - spezialisierte Formate, etwa DTaus im Bankenbereich

Eine kleine Auswahl

- HTTP (Webserver)
- FTP (FTP-Fileserver)
- SMTP (Mail-Versand)
- POP3, IMAP (Mail-Empfang)
- IRC (Chatten)
- SSL (Verschlüsselung)
- Time Protocol (Zeitabfrage)

Grundlagen / Allgemeines

- meist Kommunikation zwischen zwei Seiten
- typisch: Client-Server-Prinzip
- verbindungslos
 - A schickt Anfrage an B
 - B schickt Antwort an A
 - Ende
- verbindungsorientiert
 - A und B bauen Verbindung auf
 - A und B kommunizieren über die Verbindung, bis sie geschlossen wird

- Kommunikation folgt einem festen, vorgegebenen Protokoll
- Client und Server müssen sich exakt an das Protokoll halten, damit Kommunikation gelingt
- wie im „richtigen Leben“
 - A: „Wie möchten Sie bezahlen?“ –
B: „mit Kreditkarte“ – *erfolgreich*
 - A: „Wie möchten Sie bezahlen?“ –
B: „Heute ist es sehr warm“ – *nicht erfolgreich*

- HTTP: **H**ypertext **T**ransfer **P**rotocol
- Client: Webbrowser (Firefox, IE, Opera etc.)
- Server: Webserver (Apache, Microsoft IIS etc.)
- verbindungsorientierte Kommunikation (TCP)
- Anfrage: HTTP-Request
- Antwort: HTTP-Response

HTTP-Requests

- GET (Daten: Header + Dokument holen)
- POST (Daten an den Server senden, Formulare)
- HEAD (nur Header holen)

HTTP GET

- Anfrage:

```
GET /ii-ws2010/fom-itis-datenformate-01.pdf HTTP/1.1
Host: fom.hgesser.de
```

- Antwort:

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 10:33:23 GMT
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6
Last-Modified: Fri, 03 Sep 2010 13:40:06 GMT
ETag: "3c630f7e-aec13-118b9980"
Accept-Ranges: bytes
Content-Length: 715795
Content-Type: application/pdf
```

```
%PDF-1.4
[...]
```

- Angabe des Hosts

Host: fom.hgesser.de

war früher (HTTP 1.0) nicht nötig und nicht möglich

- Moderne Webserver hosten mehrere Domains unter derselben IP-Adresse (z. B. fom.hgesser.de, hm.hgesser.de, hgesser.com) – ohne Angabe des Hostnamens keine Unterscheidung möglich

HTTP HEAD: holt nur den Header

- Anfrage:

```
HEAD /ii-ws2010/fom-itis-datenformate-01.pdf HTTP/1.1
Host: fom.hgesser.de
```

- Antwort:

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 10:37:00 GMT
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6
Last-Modified: Fri, 03 Sep 2010 13:40:06 GMT
ETag: "3c630f7e-aec13-118b9980"
Accept-Ranges: bytes
Content-Length: 715795
Content-Type: application/pdf
```

HTTP POST: überträgt Zusatzdaten

- Anfrage:

```
POST /search.html HTTP/1.1
Host: fom.hgesser.de
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
```

```
search=Infrastruktur
```

- Antwort:

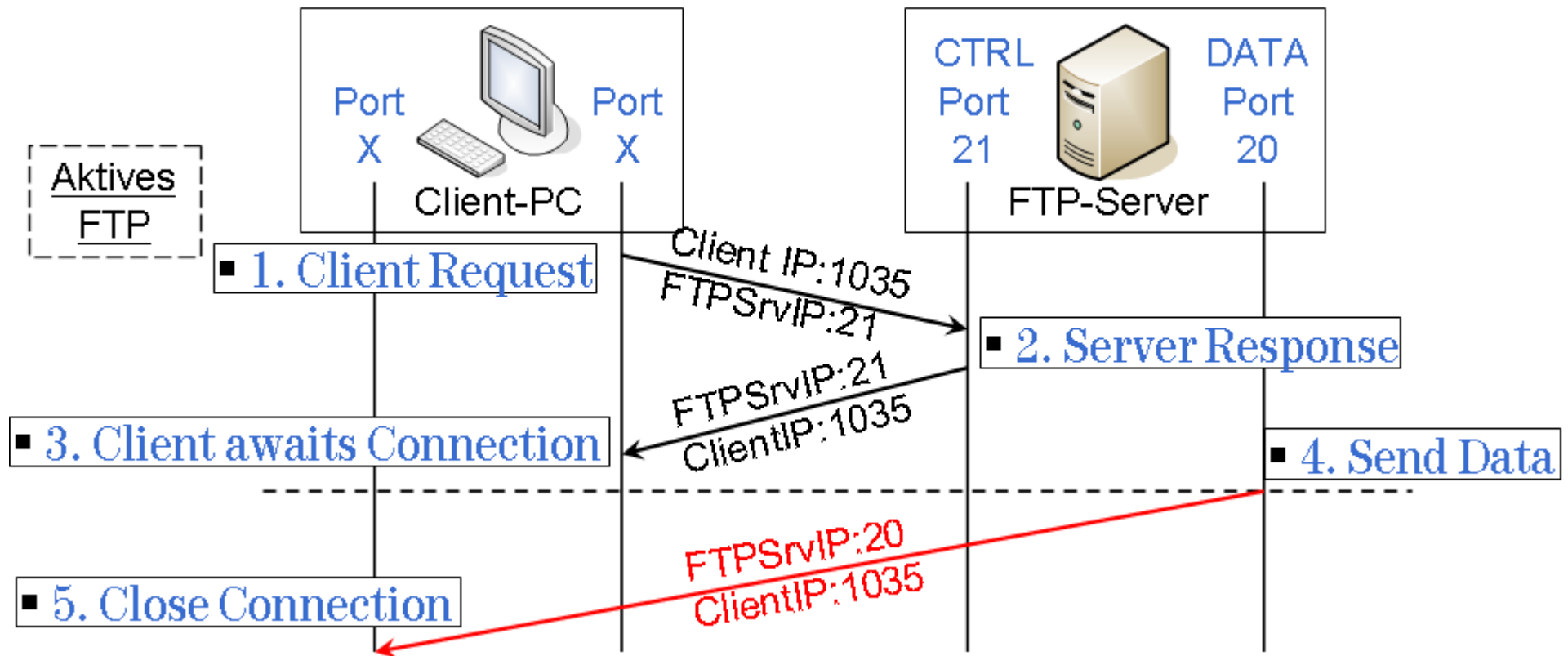
```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 10:37:00 GMT
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6
Last-Modified: Fri, 03 Sep 2010 13:40:06 GMT
[...]
```

- Verbindung bleibt nach GET/HEAD/POST geöffnet
 - Client kann sofort weitere Anfragen schicken (z. B. für Bilder auf der HTML-Seite)
 - reduziert den Aufwand für Verbindungsauf- und -abbau (TCP-Verbindungen)
- Bei Fehler (z. B. falsche Syntax in Anfrage) wird Verbindung geschlossen. Rückmeldung:
HTTP/1.1 400 Bad Request

- HTTP Status Codes
 - immer in der ersten Zeile der Antwort
 - 200 OK
 - 301 Moved Permanently
 - 304 Not Modified
 - 400 Bad Request
 - 401 Unauthorized
 - 403 Forbidden
 - 404 Not Found
 - 500 Internal Server Error
 - 503 Service Unavailable
 - weitere: <http://de.wikipedia.org/wiki/HTTP-Statuscode>
- Antwort steht nur im Header!

- **FTP: File Transfer Protocol**
- Client: FTP-Client (z. B. Webbrowser oder spezielles FTP-Programm)
- Server: FTP-Server
- verbindungsorientierte Kommunikation (TCP) mit zwei Kanälen
 - Kontrollkanal (Port 21)
 - und Datenkanal (Port 20)
- aktives FTP (Direktverbindung) vs. passives FTP (Firewall)

FTP (2/3)



Quelle Bild: http://de.wikipedia.org/wiki/File_Transfer_Protocol

- Authentifizierung (Username, Passwort)
- alternativ: „anonymous FTP“
- verschiedene Übertragungsvarianten
 - ASCII
 - Binary

- **SMTP: Simple Mail Transfer Protocol**
- Kommunikation im Textmodus, SMTP-Port 25
- Kommandos:
 - HELO („Begrüßung“)
 - MAIL FROM: *Adresse*
 - RCPT TO: *Adresse*
 - DATA
 - dann die Nachricht (erst die Mail-Header)

Mail-Versand mit SMTP (2/2)

```
esser@netbook:~$ telnet hgesser.de smtp
Trying 217.160.135.96...
Connected to hgesser.de.
Escape character is '^]'.
220 hgesser.de ESMTP Exim 3.36 #1 Sat, 11 Sep 2010 00:28:31 +0200
helo absender.de
250 hgesser.de Hello mnch-123.pool.mediaways.net [93.133.124.239]
mail from: h.g.esser@gmx.de
250 <h.g.esser@gmx.de> is syntactically correct
rcpt to: esser@hgesser.de
250 <esser@hgesser.de> verified
data
354 Enter message, ending with "." on a line by itself
Hallo, das ist eine Nachricht.
.
250 OK id=10uC5Z-0005cW-00
quit
221 s15337257.onlinehome-server.info closing connection
Connection closed by foreign host.
```

- veraltetes (aber angenehm einfaches) Protokoll, mit dem man nach der Uhrzeit fragen kann
- Kontakt zu Port 13 (daytime) eines Servers, der einen Daytime-Service anbietet
- Server liefert Zeit als lesbaren Text

```
esser@netbook:~$ netcat hgesser.de daytime  
Sat Sep 11 00:23:55 2010
```

- nicht mit NTP verwechseln ...

Und wieder: Eine kleine Auswahl

- RSS (News-Feeds)
- DTaus (Banküberweisungen)
- PDF/PostScript
- CSV (Tabellendaten)
- EPS (Vektorgrafik-Austauschformat)
- RTF (Textdatei-Austauschformat)

- **RSS: Really Simple Syndication**
- RSS-Feed: XML-Format, enthält Titel, Kurzfassungen und Links zu Nachrichten
- typisch bei Blogs, News-Seiten u. ä.
- Idee:
 - News-Feed abonnieren
 - regelmäßig Infos zu neuen Beiträgen erhalten
 - Bei Klick auf Link den vollen Artikel lesen


```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
  <channel>
    <title>IT-Infrastruktur News</title>
    <link>http://fom.hgesser.de/</link>
    <description>Infos zu IT-Infrastruktur / FOM</description>
    <language>de-de</language>
    <copyright>Hans-Georg Eßer</copyright>
    <pubDate>Sat, 11 Sep 2010 2:43:19</pubDate>

    <item>
      <title>Folien zu RSS fertig</title>
      <description>Kurze Zusammenfassung: RSS ...</description>
      <link>http://fom.hgesser.de/rss-folien/</link>
      <author>Hans-Georg Eßer</author>
      <guid>http://fom.hgesser.de/id00012</guid>
      <pubDate>Sat, 11 Sep 2010 2:43:19</pubDate>
    </item>

    <item> ... </item>
  </channel>
</rss>
```



- Datenträgeraustauschverfahren
 - ursprünglich: mit Disketten („Datenträger-...“)
 - auch: online
 - Kunde $\leftrightarrow$ Bank und Bank $\leftrightarrow$ Bank
- Datei besteht aus
 - A-Satz: „Datenträgervorsatz“ (Header), 128 Byte
 - C-Sätze: Buchungen („Zahlungsaustausch“),
187 + 29 · n Byte
 - E-Satz: „Datenträgernachsatz“ (Prüfsumme), 128 Byte

A-Satz

1	0	4 Zeichen	Länge des Datensatzes, immer "0128"
2	4	1 Zeichen	Datensatz-Typ, immer 'A'
3	5	2 Zeichen	Art der Transaktionen "LB" für Lastschriften Bankseitig "LK" für Lastschriften Kundenseitig "GB" für Gutschriften Bankseitig "GK" für Gutschriften Kundenseitig
4	7	8 Zeichen	Bankleitzahl des Auftraggebers
5	15	8 Zeichen	CST, "00000000", nur belegt, wenn Diskettenabsender Kreditinstitut
6	23	27 Zeichen	Name des Auftraggebers
7	50	6 Zeichen	aktuelles Datum im Format DDMMJJ
8	56	4 Zeichen	CST, " " (Blanks)
9	60	10 Zeichen	Kontonummer des Auftraggebers
10	70	10 Zeichen	Optionale Referenznummer
11a	80	15 Zeichen	Reserviert, 15 Blanks
11b	95	8 Zeichen	Optionales Ausführungsdatum, DDMMJJJJ.
11c	103	24 Zeichen	Reserviert, 24 Blanks
12	127	1 Zeichen	Währungskennzeichen: " " = DM, "1" = Euro
-- 128 Zeichen			

Quelle: Dokumentation zum Programm *dtaus*
(<http://www.infodrom.org/projects/dtaus/>)

C-Satz (1/3)

1	0	4	Zeichen	Länge des Datensatzes, $187 + x * 29$ (x..Anzahl Erweiterungsteile)
2	4	1	Zeichen	Datensatz-Typ, immer 'C'
3	5	8	Zeichen	Bankleitzahl des Auftraggebers (optional)
4	13	8	Zeichen	Bankleitzahl des Kunden
5	21	10	Zeichen	Kontonummer des Kunden
6	31	13	Zeichen	1. Zeichen "0" 2. - 12. Zeichen interne Kundennummer oder Nullen 13. Zeichen "0" Die interne Nummer wird vom erstbeauftragten Institut zum endbegünstigten Institut weitergeleitet. Die Weitergabe der internen Nummer an den Überweisungsempfänger ist der Zahlstelle freigestellt.

Quelle dieser und der folgenden zwei Seiten:
Dokumentation zum Programm *dtaus* (<http://www.infodrom.org/projects/dtaus/>)

C-Satz (2/3)

7a	44	2 Zeichen	Art der Transaktion
7b	46	3 Zeichen	----- "-----
			"04000" Lastschrift des Abbuchungsauftragsverfahren
			"05000" Lastschrift des Einzugsermächtigungsverfahren
			"05005" Lastschrift aus Verfügung im elec. Cash-System
			"05006" Wie 05005 mit ausländischen Karten
			"05015" Lastschrift aus Verfügung im elec. Cash-System - POZ
			"51000" Überweisungs-Gutschrift
			"53000" Überweisung Lohn/Gehalt/Rente
			"54XXJ" VL mit Sparzulage
			XX = 00 oder %-Satz der Sparzulage
			J = Endziffer des Jahres für diese Leistung
			"56000" Überweisung öffentlicher Kassen
8	49	1 Zeichen	Reserviert, " " (Blank)
9	50	11 Zeichen	Betrag
10	61	8 Zeichen	Bankleitzahl des Auftraggebers
11	69	10 Zeichen	Kontonummer des Auftraggebers

C-Satz (3/3)

12	79	11	Zeichen	Betrag in Euro einschließlich Nachkommastellen, nur belegt, wenn Euro als Währung angegeben wurde (A12, C17a), sonst Nullen
13	90	3	Zeichen	Reserviert, 3 Blanks
14a	93	27	Zeichen	Name des Kunden
14b	120	8	Zeichen	Reserviert, 8 Blanks
--	--	128	Zeichen	
15	128	27	Zeichen	Name des Auftraggebers
16	155	27	Zeichen	Verwendungszweck
17a	182	1	Zeichen	Währungskennzeichen: " " = DM, "1" = Euro
17b	183	2	Zeichen	Reserviert, 2 Blanks
18	185	2	Zeichen	Anzahl der Erweiterungsdatensätze, "00" bis "15"
19	187	2	Zeichen	Typ (1. Erweiterungsdatensatz) "01" Name des Kunden "02" Verwendungszweck "03" Name des Auftraggebers
20	189	27	Zeichen	Beschreibung gemäß Typ
21	216	2	Zeichen	wie C19, oder Blanks, (2. Erweiterungsdatensatz)
22	218	27	Zeichen	wie C20, oder Blanks
23	245	11	Zeichen	11 Blanks
--	--	--	--	Ende des ersten Erweiterungsdatensatzes
--	--	--	--	256 Zeichen

- Daten erzeugen mit Tool *dtaus*
(<http://www.infodrom.org/projects/dtaus/>)
- Umwandeln mit
`dtaus -dtaus -c control.txt`
- Ergebnis in Datei *dtaus0.txt*

control.txt:

```
BEGIN {
  Art      GK
  Name     Martin Schulze
  Konto    123545
  BLZ      2004002
  Ausfuehrung 23.12.2001
  Euro
}

{
  Transaktion Gutschrift
  Name       Martha Schulze
  Konto      98832
  BLZ        2004003
  Betrag     20.00
  Zweck      Gebuehr Wohnheimnetz
  Text       Anschluss u. 11+12.97
}
```


Banking: DTaus (7/7)

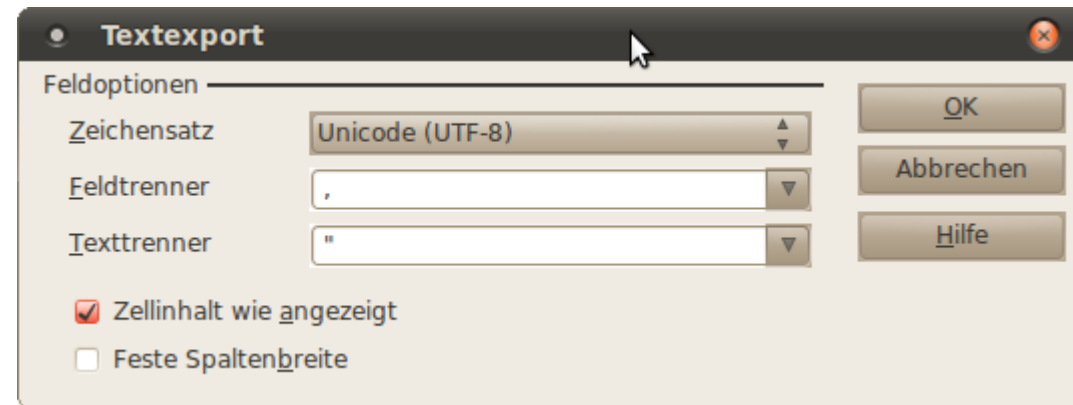
00000000	30	31	32	38	41	47	4b	30	32	30	30	34	30	30	32	30	0128AGK020040020
00000010	30	30	30	30	30	30	30	4d	41	52	54	49	4e	20	53	43	0000000MARTIN SC
00000020	48	55	4c	5a	45	20	20	20	20	20	20	20	20	20	20	20	HULZE
00000030	20	20	31	30	30	39	31	30	20	20	20	20	30	30	30	30	100910 0000
00000040	31	32	33	35	34	35	30	30	30	30	30	30	30	30	30	30	1235450000000000
00000050	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	32	2
00000060	33	31	32	32	30	30	31	20	20	20	20	20	20	20	20	20	3122001
00000070	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	31	1
00000080	30	32	31	36	43	30	32	30	30	34	30	30	32	30	32	30	0216C02004002020
00000090	30	34	30	30	33	30	30	30	30	30	39	38	38	33	32	30	0400300000988320
000000a0	30	30	30	30	30	30	30	30	30	30	30	30	35	31	30	30	00000000000005100
000000b0	30	20	30	30	30	30	30	30	30	30	30	30	30	30	32	30	0 000000000000020
000000c0	30	34	30	30	32	30	30	30	30	31	32	33	35	34	35	30	0400200001235450
000000d0	30	30	30	30	30	30	32	30	30	30	20	20	20	4d	41	52	0000002000 MAR
000000e0	54	48	41	20	53	43	48	55	4c	5a	45	20	20	20	20	20	THA SCHULZE
000000f0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
00000100	4d	41	52	54	49	4e	20	53	43	48	55	4c	5a	45	20	20	MARTIN SCHULZE
00000110	20	20	20	20	20	20	20	20	20	20	20	47	45	42	55	45	GEBUE
00000120	48	52	20	57	4f	48	4e	48	45	49	4d	4e	45	54	5a	20	HR WOHNHEIMNETZ
00000130	20	20	20	20	20	20	31	20	20	30	31	30	32	41	4e	53	1 0102ANS
00000140	43	48	4c	55	53	53	20	55	2e	20	31	31	2b	31	32	2e	CHLUSS U. 11+12.
00000150	39	37	20	20	20	20	20	20	20	20	20	20	20	20	20	20	97
00000160	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	

- **CSV: Comma-Separated Values**
- einfache Textdatei, die Tabellen in folgendem Aufbau speichert:
 - jede Zeile der Datei entspricht einer Tabellenzeile
 - Felder (Spalten) sind durch ein Trennsymbol (oft: ein Komma, alternativ: Semikolon, Tabulator, Leerzeichen) getrennt
 - Felder zusätzlich in Anführungszeichen (um Kommata im Feld zu erlauben)

	A	B
1	Eins	Vier
2	Zwei	Fünf
3	Drei	Sechs
4		

Ergebnis des Exports
ins CSV-Format:

"Eins", "Vier"
 "Zwei", "Fünf"
 "Drei", "Sechs"



- **EPS: Encapsulated PostScript**
- im Prinzip eine normale (einseitige) PostScript-Datei, aber mit einer zusätzlichen „Bounding Box“
- EPS-Dateien kann man in andere Dokumente integrieren (wie Bilddateien)

```
%!PS-Adobe-3.0 EPSF-3.0
```

```
%%Title: rob-aktiv.eps
```

```
%%Creator: (ImageMagick)
```

```
%%CreationDate: Thu May 13 09:43:18 2004
```

```
%%BoundingBox: 0 0 168 86
```

```
%LanguageLevel: 1
```

```
%%Pages: 1
```

unten links: (0, 0)

oben rechts: (168, 86)

(Einheit: 1/72 Zoll)

- **CAD**, Computer **A**ided **D**esign
- **GIS**, Geo**i**nformation**s**system
- **DTP**, **D**esktop **P**ublishing

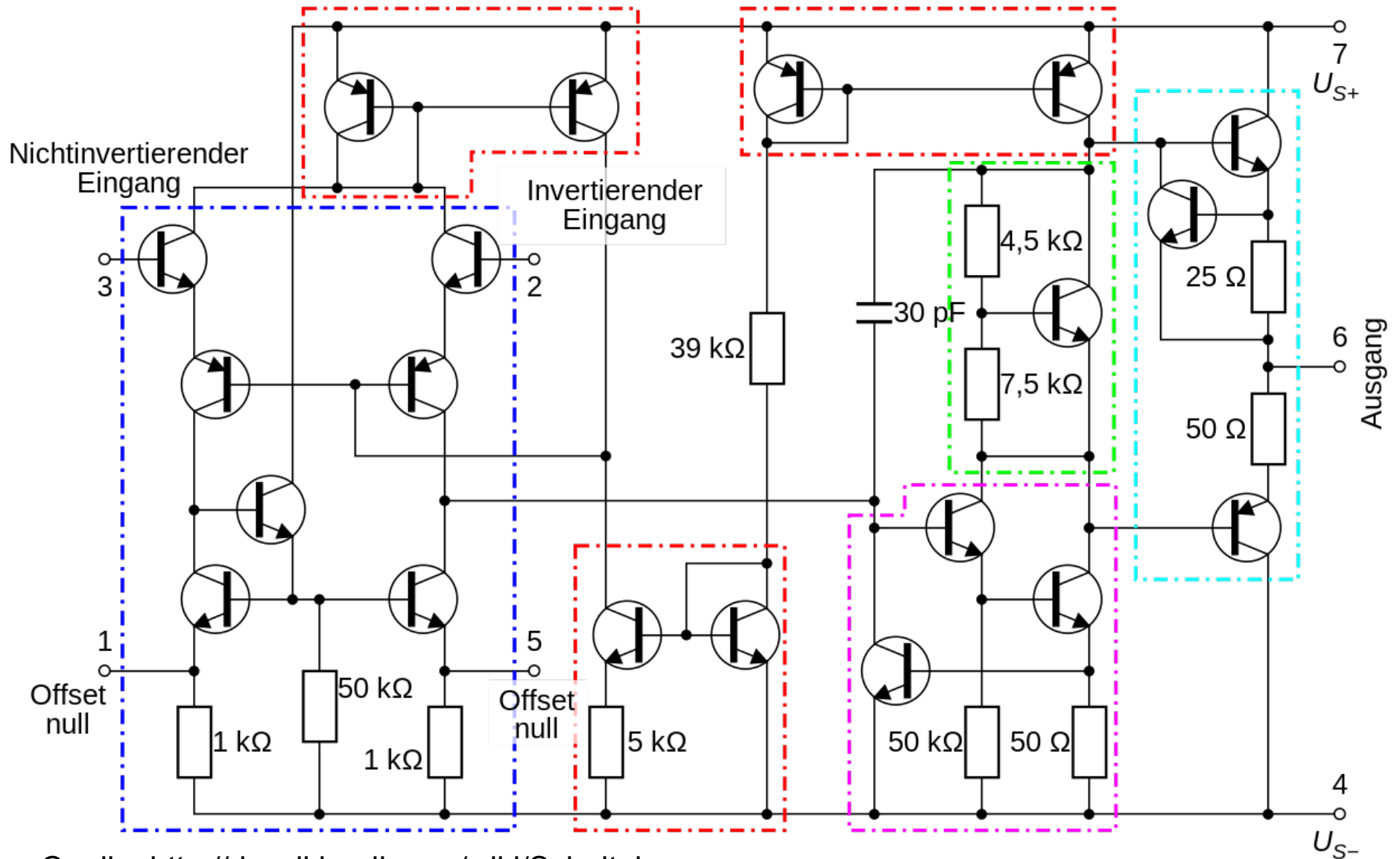
Computer Aided Design

- Mechanische Konstruktion
 - Bauwesen
 - Architektur (CAAD)
 - Holzbau
 - Ingenieurbau
 - Historische Rekonstruktion
 - Städtebau
 - Wasserbau
 - Verkehrswegebau
 - Vermessungswesen
 - Produktdesign

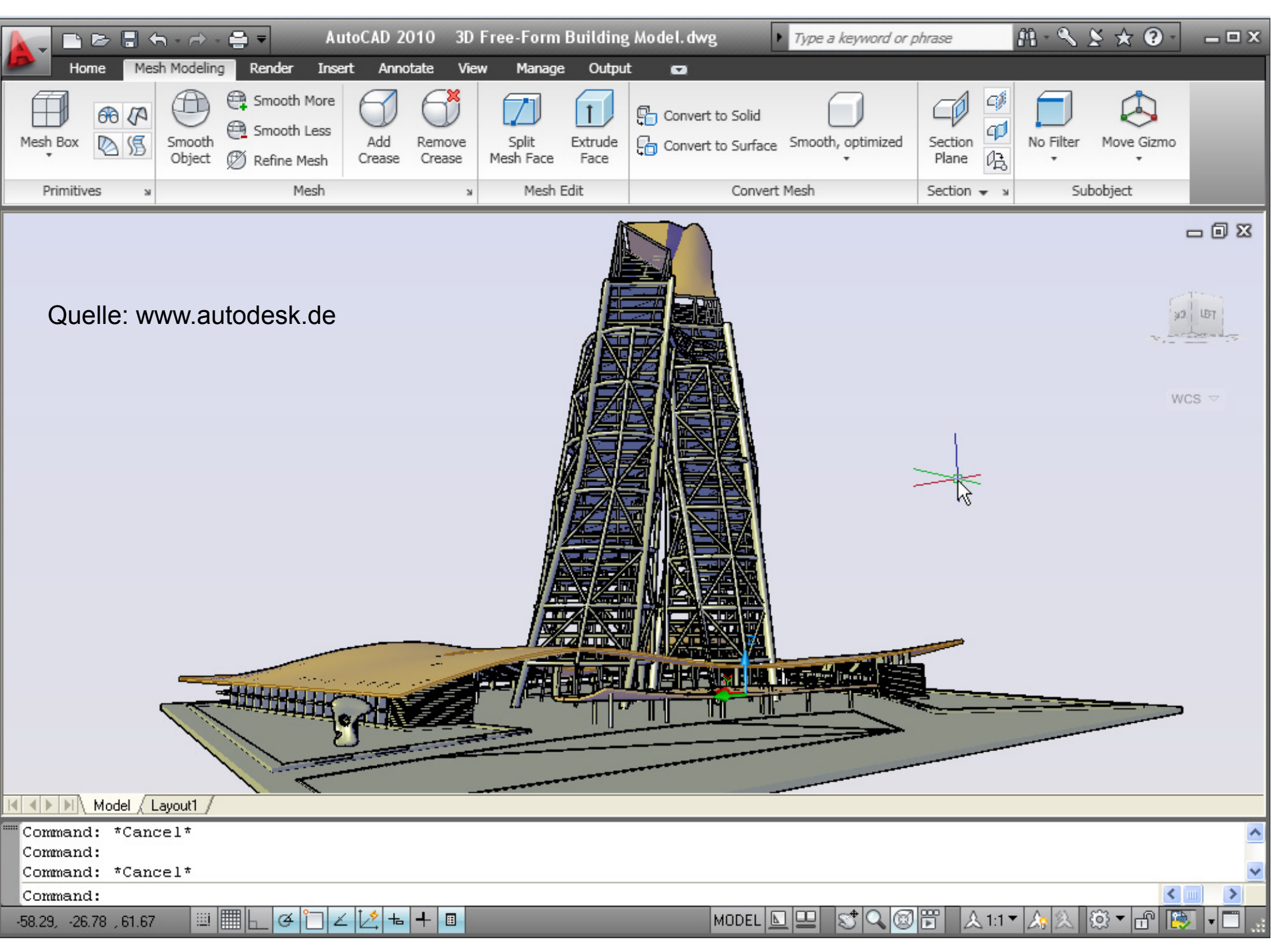
- Maschinenbau
 - Anlagenbau
 - Fahrzeugbau
 - Formen- und Werkzeugbau
 - Antriebstechnik
 - Mechanische Simulation
- Schaltpläne in der Elektrotechnik
- Zahnmedizin
- Schmuck- und Textilindustrie
- Elektronische Schaltungen

Quelle: <http://de.wikipedia.org/wiki/CAD>

- Dateiformate: vektorbasiert
- Verwaltung in objektorientierter Datenbank (nicht in einfachen Vektorgrafikdateien)
- Jedes Objekt besteht aus kleineren Teilobjekten
- Objekte haben Eigenschaften
 - Maschinenteile: Anschlüsse, Befestigungen etc.



Quelle: <http://de.wikipedia.org/wiki/Schaltplan>



Quelle: www.autodesk.de

- GIS: **G**eo**i**nformation**s**system
- Geometriedaten + Sachdaten
- gibt es vektor- und rastergrafikbasiert
- Beispiel: **GPX**, **GPS Exchange Format**
 - XML-basiert, speichert
 - GPS-Informationen (Breiten- und Längengrad, Höhe)
 - Zusatzdaten (Zeitpunkt, Ortsbeschreibung etc.)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1"
  creator="byHand" version="1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1
  http://www.topografix.com/GPX/1/1/gpx.xsd">
```

```
  <wpt lat="39.921055008" lon="3.054223107">
    <ele>12.863281</ele>
    <time>2005-05-16T11:49:06Z</time>
    <name>Cala Sant Vicenç - Mallorca</name>
    <sym>City</sym>
  </wpt>
</gpx>
```

lat = latitude = Breite,
lon = longitude = Länge,
ele = elevation = Höhe
wpt = waypoint = Wegpunkt

GIS (3/3)

Quelle: <http://www.gpsvisualizer.com/examples/>



- Layout-/Satzprogramme (Desktop Publishing)
- ähnlich wie Textverarbeitung, aber rahmenbasiert
- Text fließt von Rahmen zu Rahmen (auch zur nächsten Seite)
- Beispiele: Adobe InDesign, QuarkXPress, Scribus
- Dateiformate proprietär (bei kommerziellen Anwendungen)

Zeitschriftenlayout (2/2)

InDesign Datei Bearbeiten Layout Schrift Objekt Tabelle Ansicht Fenster Hilfe

150 %

X: -271,343 mm B: 57 mm
Y: 80,433 mm H: 63,5 mm

360 350 340 330 320 310 300 290 280 270 260 250 240 230 220 210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40

*120-122_guru-training_hej.indd @ 150 % [Umgewandelt]

Hans-Georg Eßer

In den ersten zwei Teilen der Shell-Einführung [1,2] haben wir Sie mit den Grundlagen der Shell vertraut gemacht. Jetzt stellen wir einige Spezialtools vor, mit denen die Bash besonders leistungsfähig wird – darunter „xargs“.

In der letzten Ausgabe haben wir Ihnen am Ende des Guru-Trainings ein Kommando vorgestellt, mit dem Sie mehrere mit *find* gefundene PDF-Dateien gleichzeitig im Adobe Reader öffnen können – mit dem Versprechen, das genauer zu erklären. Der Befehl war:

```
find /tmp/ -name '*.pdf' -print0 | xargs -0 acroread
```

Er besteht aus zwei Kommandos, die über eine so genannte Pipe miteinander verbunden sind:

- Das erste Kommando lautet *find /tmp/ -name '*.pdf' -print0* und findet alle Dateien mit Endung *.pdf*, die im temporären Ordner */tmp/* oder einem seiner Unterverzeichnisse liegen. Die zusätzliche Option *-print0* ist für den Zweck des anschließenden Adobe-Reader-Starts wichtig, denn Sie bewirkt, dass *find* in der Ausgabe nicht Leerzeichen oder Zeilenumbrüche, sondern ein spezielles Sonderzeichen zwischen die Dateinamen schreibt. Das hat den Vorteil, dass der gesamte Befehl auch dann funktioniert, wenn sich in einigen der Dateinamen Leerzeichen befinden.
- Über das Pipe-Zeichen *|* wird die Ausgabe des ersten Kommandos (eine Liste von Dateinamen) zur Eingabe für das zweite Kommando. Es heißt *xargs -0 acroread*. Das Tool *xargs* ist besonders praktisch: Es wandelt Informationen aus der Standard-eingabe in Argumente um. Die Wirkung können Sie leicht ausprobieren, indem Sie den Befehl *ls | xargs echo* eingeben – *ls* gibt den Inhalt des Wurzelverzeichnisses aus, und *xargs* macht aus dieser Ausgabe Argumente für den Befehl *echo*, der einfach seine Argumente ausgibt (siehe Kasten *Ausgabe als Argumente*). Die Option *-0* (das ist eine Null, nicht der Großbuchstabe „O“) teilt *xargs* mit, dass es nicht normale Leerzeichen als Trenner für die verschiedenen Argumente interpretieren soll, sondern das Sonderzeichen, das *find* über die Option *-print0* als Trenner eingefügt hat.

Die Regel ist: Wenn Sie *find* und *xargs* auf diese Weise kombinieren, müssen Sie entweder *find* mit der Option *-print0* und *xargs* mit der Option *-0* verwenden oder bei beiden Kommandos auf die Optionen verzichten – andernfalls passen Ausgabe und Eingabe nicht zusammen, und *xargs* „versteht“ Sie nicht.

Für Sammler: Umleitungen

Wollen Sie die Ausgabe eines Programms nicht (nur) im Terminalfenster sehen, sondern dauerhaft archivieren oder etwa für den Versand als E-Mail-Anhang in einer Datei sichern, nutzen Sie ein praktisches Feature der Shell: die Ausgabeumleitung. Dazu setzen Sie ganz am Schluss hinter das Kommando ein Großeszeichen (*>*) und einen Dateinamen. Jeder Aufruf mit *> Dateiname* löscht zunächst einen eventuell vorhandenen Inhalt der Datei und schreibt dann die Ausgabe hinein – so können Sie immer nur eine einzige

gender kontinuierlich einleiten, bevor Ausgaben Umleitung mit *&>* in Sie mit *cat* schnell über *essner@netbookkg: .log* ls: Zugriff auf ht möglich: No s ctory -rw-r--r-- 1 ess 2010-02-07 00:2

Probieren Sie auch kaufmännische Und-Verknüpfung. Suchen und Ersetzen Um in einer Datei ko fel“ durch „Birne“ Z diese in einem Editor chen-Ersetzen-Funkti aber auch auf der Sh sed. Dessen Syntax i nungsbedürftig, für schnell gemeistert: H txt, können Sie mit sed -e s/Apfel/E > test2.txt

Spezialbehandlung für Fehler

Beachten Sie, dass *>* und *>>* nur „normale“ Ausgaben eines Befehls erfassen. Es gibt auch spezielle Fehlerausgaben, die aus Linux-Sicht über einen anderen „Kanal“ laufen, die so genannte Standardfehlerausgabe (*stderr*). Entsprechend heißt die normale Ausgabe Standardausgabe (*stdout*). Erzeugt ein Programm neben den regulären Ausgaben auch Fehlermeldungen, bleiben diese weiterhin im Terminalfenster sichtbar und stehen nicht in der Zieldatei. Um auch die Fehler in der Datei zu protokollieren, passen Sie die Befehlszeile an und stellen vor das *>* bzw. *>>* ein kaufmännisches Und (*&*), insgesamt also: *&>* bzw. *&>>*. Liegt im aktuellen Arbeitsverzeichnis z. B. eine Datei *mozilla.ps*, aber keine Datei

AUSGABE ALS ARGUMENTE

Das folgende Kommando gibt den Inhalt des Wurzelverzeichnisses aus, setzt aber noch die Worte „Inhalt Wurzelverzeichnis:“ vor die Ausgabe.

```
$ ls | xargs echo "Inhalt Wurzelverzeichnis:"
```

Inhalt Wurzelverzeichnis: bin boot dev etc home lib lost+found media mnt opt proc root sbin srv sys tmp usr var

Das funktioniert in drei Schritten:

- Der Befehl *ls* / listet den Inhalt des Wurzelverzeichnisses auf, gibt also z. B. *bin boot dev etc home lib lost+found media mnt opt proc root sbin srv sys tmp usr var* aus. Wenn Sie den Befehl alleine aufrufen, erscheinen die Namen direkt in der Shell.
- Über das Pipe-Zeichen reichen Sie die

Der Befehl *xargs echo* liest zunächst die Ausgabe von *ls* ein und setzt sie mehrfach gezeigte Namen. Es baut das Kommando zusammen, „Inhalt Wurzelverzeichnis:“ neuen Argumentliste entsteht der Befehl *echo "Inhalt Wurzelverzeichnis: bin boot dev etc home lib lost+found media mnt opt proc root sbin srv sys tmp usr var"* *xargs* dann aus.

Das Kommando *echo* führt den Umweg, auf dem erhalten hat; es „schließt“ es mit den Argumenten *echo "Inhalt Wurzelverzeichnis: bin boot dev etc home lib lost+found media mnt opt proc root sbin srv sys tmp usr var"*

Archive und Software-Pakete

- Kompressions- und Archivformate
 - Kompatibilität / Plattformen

- (Software-) Paketformate

Installierbare Pakete für ...

- Windows: MSI
- Linux: RPM, DEB
- Mac OS: DMG

- In der Linux-/Unix-Welt: diverse Komprimierer mit eigenen Dateiformaten
 - datei.txt → datei.txt.gz (gzip)
 - datei.txt → datei.txt.bz2 (bzip2)
 - datei.txt → datei.txt.Z (compress)
 - etc.
- In der Windows-Welt: meist Archivformate
 - datei.txt → archiv.zip
(auch wenn nur eine Datei komprimiert wird)

```

esser@netbook:~/tmp$ ls -l
-rw-r--r-- 1 esser esser 14226 2010-07-12 21:07 index.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:03 kopie1.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:03 kopie2.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:03 kopie3.html
esser@netbook:~/tmp$ gzip kopie1.html ; bzip2 kopie2.html
esser@netbook:~/tmp$ compress kopie3.html ; ls -l
-rw-r--r-- 1 esser esser 14226 2010-07-12 21:07 index.html
-rw-r--r-- 1 esser esser  3806 2011-01-14 10:03 kopie1.html.gz
-rw-r--r-- 1 esser esser  3893 2011-01-14 10:03 kopie2.html.bz2
-rw-r--r-- 1 esser esser  6850 2011-01-14 10:03 kopie3.html.Z
esser@netbook:~/tmp$ file kopie*
kopie1.html.gz:  gzip compressed data, was "kopie1.html", from
                  Unix, last modified: Fri Jan 14 10:03:25 2011
kopie2.html.bz2: bzip2 compressed data, block size = 900k
kopie3.html.Z:   compress'd data 16 bits

```

- komprimierte Datei enthält (meist) keine Metadaten
 - Dateiname des Originals:
entsteht durch Weglassen der neuen Dateiendung (.gz, .bz2, .Z)
 - Besitzer, Gruppe, Zugriffsrechte des Originals:
sind identisch in komprimierte Datei übernommen

```

esser@netbook:~/tmp$ ls -l programm
-rwxr-x--- 1 esser users 211 2011-01-14 10:32 programm
esser@netbook:~/tmp$ gzip programm
esser@netbook:~/tmp$ ls -l programm.gz
-rwxr-x--- 1 esser users 40 2011-01-14 10:32 programm.gz
  
```

- komprimierte Datei enthält **meist** keine Metadaten, bei gzip aber Dateiname/Datum:

By default, gzip keeps the original file name and time-stamp in the compressed file. These are used when decompressing the file with the `-N` option. This is useful when the compressed file name was truncated or when the time stamp was not preserved after a file transfer.

`-N --name`

When compressing, always save the original file name and time stamp; this is the default. When decompressing, restore the original file name and time stamp if present. This option is useful on systems which have a limit on file name length or when the time stamp has been lost after a file transfer.

Kompression einzelner Dateien

```
esser@netbook:~$ ls -l
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 test1.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 test2.html

esser@netbook:~$ gzip test1.html

esser@netbook:~$ gzip -n test2.html # (ohne Namen)

esser@netbook:~$ strings test1.html.gz | grep test
test1.html

esser@netbook:~$ strings test2.html.gz | grep test

esser@netbook:~$ mv test1.html.gz FALSCH1.gz
esser@netbook:~$ mv test2.html.gz FALSCH2.gz

esser@netbook:~$ gunzip -N FALSCH1.gz
esser@netbook:~$ gunzip -N FALSCH2.gz

esser@netbook:~$ ls -l
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 FALSCH2
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 test1.html
```

Kompression einzelner Dateien

```
esser@netbook:~/tmp$ zip archiv.zip index.html
adding: index.html (deflated 73%)
```

```
esser@netbook:~/tmp$ ls -l index.html archiv.zip
-rw-r--r-- 1 esser esser 3946 2011-01-14 10:08 archiv.zip
-rw-r--r-- 1 esser esser 14226 2010-07-12 21:07 index.html
```

```
esser@netbook:~/tmp$ unzip -l archiv.zip
```

```
Archive: archiv.zip
```

Length	Date	Time	Name
-----	-----	-----	----
14226	2010-07-12	21:07	index.html
-----			-----
14226			1 file

- speichern (meist komprimiert)
 - mehrere Dateien
 - ganze Verzeichnishierarchien
- besonders populär:
 - ZIP (Windows, Mac OS)
 - tar.gz, tar.bz2 (Linux, Unix)
 - DMG (Mac OS)
- eingeschränkt plattformübergreifend

- Probleme beim Transfer zwischen Betriebssystemen (Windows, Linux, Mac OS)
- Was soll man im Archiv (neben den Dateien selbst) speichern?
 - Timestamps – Erstellung, letzte Änderung, letzter Zugriff (nicht jedes Dateisystem hat alle drei Typen)
 - Zugriffsrechte – bei jedem OS anders geregelt: Besitzer, Gruppe, ACLs, klassische Unix-Dateirechte
 - speziell bei Mac OS: Resource Forks
 - speziell bei Windows: Streams („datei.txt:stream“)

- Unterschiedliche Dateiattribute (der versch. Betriebs- bzw. Dateisysteme) kann ein Archivformat durchaus handhaben
- Aber was tun beim Entpacken einer Datei, die auf einem fremden System gepackt wurde?
 - Im Idealfall: So viele Eigenschaften „mitnehmen“ wie möglich, z. B. Datum der letzten Änderung gibt es auf jedem System
 - Oft: Wegwerfen aller Attribute
- Weiteres Problem: Dateinamenskonventionen

- ZIP-Format
 - populärer Klassiker, schon seit DOS-Zeiten benutzt
 - speichert ganze Unterordner, Archiv kann ergänzt werden
 - mittelmäßige Kompressionsrate, dafür schnell beim Packen und Entpacken
 - für alle Betriebssysteme verfügbar
 - kann mit speziellen Attributen der diversen BS umgehen (darunter auch Linux-Attribute und Mac OS Resource Forks)
 - Packprogramme mit GUI und für die Shell

- Auch Dateien ohne Endung „.zip“ können ZIP-Archive sein, z. B. aktuelle Office-Dateien:
 - .docx, .xlsx, pptx – alle XML-basierten Office-Dateien von Microsoft Office
 - .odt, .ods, .odp – alle XML-basierten Office-Dateien von OpenOffice / LibreOffice
- Solche Dateien kann man untersuchen, indem man sie in *.zip umbenennt (siehe Folien A-130 bis A-139 zu Office-Formaten)

- tar (tape archive)
 - Historie: Unix-Tool, das ganze Verzeichnisse unkomprimiert auf ein Streamer-Band (ein tape) sichert. Nur unter Linux/Unix verbreitet
 - heute: Anlegen von tar-Dateien
 - tar-Kommando bietet Optionen, mit denen sich beliebige Kompressionsprogramme nutzen lassen (-z: gzip, -j: bzip2, -l tool: tool)
 - Nutzung ohne Kompression manchmal nützlich, etwa, beim Packen eines Ordners, der nur bereits komprimierte Dateien enthält

- nur in der Linux-/Unix-Welt üblich
- Archivformate
 - tar (tape archive)
 - ar (archive)
 - cpio (copy in, copy out)
 - pax (portable archive exchange)
- Vorteil:
 - Keine Standardkompression integriert; nach Archivbildung Nutzung eines beliebigen Komprimierers

Beispiel: tar-Archiv (1)

```
esser@netbook:~/tmp$ echo "Hallo Welt" > datei1.txt
```

```
esser@netbook:~/tmp$ echo "Kleiner Test" > datei2.txt
```

```
esser@netbook:~/tmp$ tar cf archiv.tar datei*.txt
```

```
esser@netbook:~/tmp$ cat archiv.tar
```

```
datei1.txt0000644000175000017500000000001311514012427012001
```

```
0ustar  esseresserHallo Welt
```

```
datei2.txt0000644000175000017500000000001511514012436012004
```

```
0ustar  esseresserKleiner Test
```

(cat zeigt nur darstellbare Zeichen an)

Beispiel: tar-Archiv (2)

```
esser@netbook:~/tmp$ hexdump -C archiv.tar
```

```
00000000  64 61 74 65 69 31 2e 74  78 74 00 00 00 00 00 00 |datei1.txt.....|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
*
00000060  00 00 00 00 30 30 30 30  36 34 34 00 30 30 30 31 |....0000644.0001|
00000070  37 35 30 00 30 30 30 31  37 35 30 00 30 30 30 30 |750.0001750.0000|
00000080  30 30 30 30 30 31 33 00  31 31 35 31 34 30 31 32 |0000013.11514012|
00000090  34 32 37 00 30 31 32 30  30 31 00 20 30 00 00 00 |427.012001. 0...|
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
*
00000100  00 75 73 74 61 72 20 20  00 65 73 73 65 72 00 00 |.ustar .esser..|
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
00000120  00 00 00 00 00 00 00 00  00 65 73 73 65 72 00 00 |.....esser..|
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
*
00000200  48 61 6c 6c 6f 20 57 65  6c 74 0a 00 00 00 00 00 |Hallo Welt.....|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
*
00000400  64 61 74 65 69 32 2e 74  78 74 00 00 00 00 00 00 |datei2.txt.....|
[...]
```

Beispiel: tar-Archiv (3)

[...]

00000400	64 61 74 65 69 32 2e 74	78 74 00 00 00 00 00 00	datei2.txt.....
00000410	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	

*

00000460	00 00 00 00 30 30 30 30	36 34 34 00 30 30 30 31	0000644.0001
00000470	37 35 30 00 30 30 30 31	37 35 30 00 30 30 30 30	750.0001750.0000
00000480	30 30 30 30 30 31 35 00	31 31 35 31 34 30 31 32	0000015.11514012
00000490	34 33 36 00 30 31 32 30	30 34 00 20 30 00 00 00	436.012004. 0...
000004a0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	

*

00000500	00 75 73 74 61 72 20 20	00 65 73 73 65 72 00 00	.ustar .esser..
00000510	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000520	00 00 00 00 00 00 00 00	00 65 73 73 65 72 00 00	esser..
00000530	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	

*

00000600	4b 6c 65 69 6e 65 72 20	54 65 73 74 0a 00 00 00	Kleiner Test....
00000610	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	

*

00002800

- Drei Haupt-„Betriebsarten“:
 - **create:** `tar -cf Archivname.tar Dateien`
 - **list:** `tar -tf Archivname.tar`
 - **extract:** `tar -xf Archivname.tar [Dateien]`
 - Option `-v` (verbose): gibt immer Namen der bearbeiteten Dateien aus

- tar.gz (auch: tgz) / tar.bz2:
 - eigentlich kein eigenständiges Format:
Dateien entstehen (eigentlich) in zwei Schritten
 - tar-Archiv erstellen (archiv.tar)
 - dieses Archiv komprimieren; gzip: archiv.tar.gz
 - dank tar-Optionen aus Anwendersicht nur ein Schritt
 - Standardformat für Archive unter Linux/Unix
 - z. B. Quellcode von Software fast immer als tar.gz- oder tar.bz2-Paket erhältlich
 - bzip2: bessere Kompression als gzip, dafür etwas langsamer beim Packen

- DMG (Apple Disk Image, Mac OS)
 - Standardformat unter Mac OS für Software-Downloads
 - Kompression und Passwortschutz möglich
 - DMG-Images werden von Mac OS automatisch „gemountet“ und erlauben dann die Installation der enthaltenen Programme (meist einfaches Kopieren in den Anwendungen-Ordner)
 - technisch: in DMG-Datei steckt ein Disk Image, vergleichbar mit der Raw-Kopie einer Plattenpartition

- Die meisten Linux-Distributionen setzen eine Paketverwaltung ein:
 - Installation von Paketen aus sog. **Repositories** (Repos, Paketquellen)
 - Bequemes Entfernen, Aktualisieren von Paketen
 - Paketverwaltung löst automatisch **Abhängigkeiten** und **Konflikte** auf
- Es gibt viele Paketsysteme. Zwei sind populär:
 - RPM (Red Hat Package Manager)
 - DEB (Debian Package Manager) & APT (Advanced Package Tool)

- Vorteile der Paketverwaltung gegenüber „install.exe“ aus Windows-Welt:
 - automatische „Masseninstallation“ von vielen Anwendungen möglich
 - automatische Aktualisierung aller Programme, für die es ein Update gibt
 - automatische Aktualisierung des ganzen Betriebssystems auf eine neuere Version

- Ein **RPM-Paket** besteht aus ...
 - einem **Archiv** mit den zu installierenden Dateien
 - **Metadaten** (Paketname, Version, Datum der Erstellung, für welche Linux-Version, Kontakt zum Paketbauer, Abhängigkeiten, Konflikte)
 - Skripte, die vor/nach Installation/Deinstallation ausgeführt werden
- Ein **RPM-Repository** listet eine Sammlung von RPM-Paketen und deren Download-Adressen auf (etwas vereinfacht...)

- Manche Programme sind in mehrere RPM-Pakete unterteilt, von denen evtl. nur eine Auswahl benötigt wird. Beispiel:
 - prog-1.2.3.i586.rpm
 - prog-1.2.3.x86_64.rpm
 - prog-lib-1.2.3.i586.rpm
 - prog-lib-1.2.3.x86_64.rpm
 - prog-debug-1.2.3.i586.rpm
 - prog-debug-1.2.3.x86_64.rpm
 - prog-data-1.2.3.noarch.rpm

- RPM-Pakete mit Kommando rpm installieren:
`rpm -i paketname.rpm`
- Update eines schon installierten Pakets:
`rpm -U paketname.rpm`
- Entfernen eines Pakets:
`rpm -e paketname` (ohne „.rpm“)
- Überprüfen eines installierten Pakets:
`rpm -V paketname` (ohne „.rpm“)

- Problem:
 - Es gibt viele verschiedene Linux-Distributionen (OpenSuse, Fedora, Red Hat, Mandriva, ...), die alle RPM-Pakete verwenden.
 - Pakete verschiedener Distributionen sind meist inkompatibel; in der Regel auch Pakete verschiedener Versionen (z. B. OpenSuse 11.2 / 11.3)
- Lösung: Repositories
 - Jeder Distributor verwaltet eines oder mehrere Repositories für jede Version seiner Distribution; mehr dazu später

- Problem: Abhängigkeiten
 - Ein Programm X benötigt eine Bibliothek Y (shared library, wie DLL-Datei bei Windows)
 - Das Paket, das X enthält, enthält kein Y
 - Der Paketmanager (RPM) weiß nicht, in welchem Paket sich die Bibliothek Y befindet – wenn es überhaupt eines für diese Linux-Version gibt
 - Installation des X-Pakets schlägt fehl (immerhin), weil RPM erkennt, dass die Bibliothek fehlt
- Lösung: auch Repositories (→ später)

- (Kein) Problem: Konflikte
 - Im RPM-Paket X findet sich der Hinweis, dass es nicht parallel zu Paket Y installiert sein darf
 - Y ist installiert, Anwender will X installieren
 - RPM verweigert Installation
 - Problem oder Feature?
- Lösung:
 - Es gibt die Option `--force`, die das Ignorieren der Konflikte erzwingt (fast immer eine schlechte Idee)

- Installation am besten nur über Repositories
- Je nach Distribution verschiedene Tools für Zugriff auf die Repos:
 - OpenSuse: Zypper (zypper install paketname)
 - Fedora/Red Hat: YUM (yum install paketname)
 - Mandriva: URPMI (urpmi paketname)
 - usw.
- Zugriff auf verschiedene Medien möglich

- Repositories auf verschiedenen Medien:
 - HTTP
 - FTP
 - lokales Verzeichnis
 - CD/DVD
 - u. a.
- Repositories nach Inhalten aufgeteilt:
 - Pakete
 - Updates
 - Quellpakete

- Neben den offiziellen Repos vom Distributor gibt es oft viele Zusatz-Repos von Drittanbietern, z. B.
 - für spezielle Multimedia-Player und -Codecs
 - für proprietäre Hardware-Treiber (etwa Grafikkarten von Nvidia oder ATI/AMD)
 - von Software-Entwicklern, die ihre Programme nicht in den „regulären“ Repos unterbringen konnten
- Wie schützt man sich vor „falschen“ Repos?

- Signierte Pakete / Repo-Keys:
 - Public-Key-Kryptographie (GPG, PGP etc.)
 - Jedes Repo bietet ein Paket mit öffentlichen Schlüsseln dieses Repos an.
Wollen Sie das Repo nutzen? → Keys installieren
 - Jedes Paket ist mit dem privaten Key des Repos signiert (diesen Key kennt nur der Repo-Betreiber)
 - Bei Installation eines Pakets aus dem Repo wird die Signatur mit dem vorhandenen public key geprüft.
Gibt es keinen Key oder einen Fehler, erscheint eine Warnung

- Gleicher Ansatz wie bei RPM:
 - Pakete über Tool dpkg installieren, updaten, löschen, Status prüfen etc.
 - Pakete enthalten Archive, Metadaten (inkl. Abhängigkeiten und Konflikten), Installations- und Deinstallationsskripte
 - Admin-Tool dpkg bietet ähnliche Funktionen wie RPM-Tool rpm
- Andere Liste von Distributionen: Debian, Ubuntu, Knoppix, ...

- DEB-Pakete nicht auf RPM-Systemen installierbar, und
- RPM-Pakete nicht auf DEB-Systemen installierbar („zwei Welten“)
- aber: Programm alien konvertiert zwischen DEB- und RPM-Formaten
- Wie in der RPM-Welt: Ein DEB-Paket zu haben, heißt nicht, es installieren zu können
- Lösung aller Probleme auch hier: Repos

- Anders als in der RPM-Welt gibt es ein einheitliches Tool für die Verwaltung von Repos: APT (Advanced Package Manager)
- Zwar müssen auch APT-Anwender für jede Distribution (und jede Version davon) separate Repos nutzen, aber die APT-Kommandos sind immer gleich
 - `apt-get install paketname`
 - `apt-cache search paketname`
 - `apt-get remove paketname`

Verwaltung der Repos in /etc/apt/resources.list

```
deb cdrom:[Ubuntu-Netbook 10.04 _Lucid Lynx_ - Release i386
(20100429.4)]/ lucid main restricted
deb http://de.archive.ubuntu.com/ubuntu/ lucid main restricted
deb-src http://de.archive.ubuntu.com/ubuntu/ lucid main restricted

## Bug Fixes, Updates
deb http://de.archive.ubuntu.com/ubuntu/ lucid-updates main restricted

## Universe Repos
deb http://de.archive.ubuntu.com/ubuntu/ lucid universe
deb http://de.archive.ubuntu.com/ubuntu/ lucid-updates universe
deb http://de.archive.ubuntu.com/ubuntu/ lucid multiverse
deb http://de.archive.ubuntu.com/ubuntu/ lucid-updates multiverse

## Fremdanbieter
deb http://download.virtualbox.org/virtualbox/debian lucid non-free
deb http://ppa.launchpad.net/d.filoni/dillo/ubuntu lucid main
```

- Aus Sicht eines Software-Anbieters:
 - Chaos – Debian, RPM, weitere Paketformate, diverse inkompatible Distributionen und deren Versionen
 - Wer eigene Repos anbieten will, muss diese für alle zu unterstützenden Distributionen separat pflegen
 - Beim RPM-Format heißt das auch: Nutzung verschiedener Tools, mit denen man Repos erstellt, denn Suse != Fedora != Mandriva...
 - Beim DEB-Format: immerhin einheitliche Tool-Welt (APT)

Beispiel: RPM-Paket „bauen“

- Um ein RPM-Paket zu bauen, ist ein sog. Specfile nötig (specification file)
- Beispiel für ein Specfile
 - Programm „Bomberclone“
 - Quelle: T. Schürmann, LinuxUser 07/2006, S. 52 ff., <http://www.linux-user.de/ausgabe/2006/07/052-rpm/>

- Specfile bomberclone.spec:

```
# Specfile fuer BomberClone
Summary: Bomberman-Klon
Name: bomberclone
Version: 0.11.6.2
Release: 1
Copyright: GPL
Group: Games/Action
Source: bomberclone-0.11.6.2.tar.gz
URL: http://www.bomberclone.de
Distribution: Suse Linux 10.0
Packager: Tim Schuermann <tschuermann@linux-user.de>
```

%description

Ein Bomberman-Klon, bei dem sich mehrere Spieler mit kleinen Bomben heftig unter Druck setzen. Mehrspielerpartien in einem Netzwerk sind ebenfalls möglich.

%prep**%setup**

./configure

%build

make

%install

make install

%files

/usr/local/bin/bomberclone

/usr/local/share/games/bomberclone

%doc /usr/local/doc/bomberclone/README

%doc AUTHORS TODO INSTALL NEWS COPYING ChangeLog

%post

ldconfig

- Paket bauen mit
`rpmbuild -bb bomberclone.spec`

- MSI-Dateien
 - historisch: „Microsoft Installer“; für Windows-Installer
 - enthalten keine eigene Installationsroutine, sondern das Software-Paket und eine „Anleitung“ für den Windows-Installer
 - komplexe Struktur, MSI-Dateien enthalten Datenbanktabellen
 - MS bietet Tool, das aus XML-Dateien eine MSI-Datei erstellt (Vorgang deutlich komplexer als beim Bauen von RPM- / DEB-Paketen)

- MSI-Dateien / Windows-Installer erlauben
 - Roll-Back bei Installationsfehlern: Alle schon vorgenommenen Änderungen rückgängig machen (z.B. auch Registry-Edits)
 - Einrichten der Deinstallationsroutine (Kontrollzentrum, Softwareverwaltung)
 - Aufteilung der Software in Komponenten, Anwender kann bei Installation auswählen, welche Teile installiert werden
 - ...

- MSI-Dateien / Windows-Installer erlauben
 - On-Demand-Installation und „Advertise“-Feature:
 - Programm enthält Menüpunkte für *nicht* installierte Komponenten
 - bei Aufruf dynamische Nachinstallation der benötigten Komponenten über den Windows-Installer – dafür keine neue Eingabe des Administrator-Passworts nötig

- kein spezielles Paketformat für Mac OS
- Software meist in DMG-Image
- Anwendung besteht häufig aus Verzeichnis
 - z. B.: Eintrag „Safari“ im Mac-OS-Programmordner ist in Wirklichkeit ein Verzeichnis „Safari.app“ (→ nächste Folie)
- Einige Programme bringen Installer mit (vgl. setup.exe/Windows)
- Neu: Mac OS AppStore (wie für iPhone & Co.)

```
imac27:Applications esser$ ls -l Safari.app/
drwxr-xr-x  12 root  wheel  408 Nov 18 21:25 Contents
```

```
imac27:Applications esser$ ls -l Safari.app/Contents/
lrwxr-xr-x    1 root  wheel      28 Nov 18 21:23 CodeResources ->
_CodeSignature/CodeResources
-rw-r--r--    1 root  wheel  14809 Oct 15 02:40 Info.plist
drwxr-xr-x    3 root  wheel   102 Nov 18 21:25 MacOS
-rw-r--r--    1 root  wheel     8 Oct 15 02:40 PkgInfo
drwxr-xr-x  475 root  wheel  16150 Nov 18 21:25 Resources
-rwxr-xr-x    1 root  wheel 214752 Oct 15 02:40 Safari Webpage Preview
Fetcher
drwxr-xr-x    3 root  wheel   102 Nov 18 21:27 SafariSyncClient.app
drwxr-xr-x    3 root  wheel   102 Apr 20 2010 WebApplicationCore.bundle
drwxr-xr-x    3 root  wheel   102 Nov 18 21:25 _CodeSignature
-rw-r--r--    1 root  wheel   458 Oct 15 02:42 version.plist
```

```
imac27:Applications esser$ ls -l Safari.app/Contents/MacOS/
-rwxr-xr-x  1 root  wheel 13518992 Oct 15 02:42 Safari
```

IT-Infrastruktur

SS 2015

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz B:

- PC als Arbeitsplatz

v1.1, 2015/03/05

Vorlesungsübersicht

Seminar

Wiss. Arbeiten

Datenformate und Wandlung

PC als Arbeitsplatz

Ergonomie und Arbeitsschutz

Rechnerstrukturen

Zentrale / verteilte IT-Infrastrukturen

Folien B

- Office-Anwendungen
 - Textverarbeitung, Serienbrief-Funktion
 - Tabellenkalkulation
 - Präsentationssoftware
 - Textsatz mit LaTeX
- Internet
 - Browser, SaaS (Software as a Service)
 - Mail, Usenet-News, FTP
 - Remote Access: RDP, VNC, SSH

- Standardapplikationen, z. B.
 - Statistik
 - Numerik
 - Computeralgebra
 - Geoinformationssysteme
- Diagramme und Schemaskizzen
- einfache Bildbearbeitung (Retusche, Veränderung der Auflösung)
- Versionierung

- Auswertung von erhobenen Daten (Messungen, Umfrage-Ergebnisse, Verkaufszahlen etc.)
- Data Mining (Wissensextraktion aus Daten)
- Mehrere Anbieter
 - SPSS Statistics (IBM)
 - PSPP (freier SPSS-Klon)
 - Programmiersprache S
(Implementationen: R, S-plus)
 - SAS (Business Intelligence Software)

Exkurs: LaTeX

- Seminararbeit evtl. mit LaTeX erstellen
- auch gute Vorbereitung für Bachelorarbeit
- hier eine Einführung in die grundlegenden Konzepte
 - LaTeX basiert auf TeX und ist ein Makropaket, das TeX leichter benutzbar macht
 - zentrale Idee: Trennung zwischen Layout und Inhalt
 - Autor konzentriert sich auf Inhalt, LaTeX übernimmt das Layout

- LaTeX ist ein Dokumenten-*Compiler*.
- Text in einem Editor Ihrer Wahl erstellen
- Als Kodierung UTF8 einstellen
- speichern als *.tex-Datei
- übersetzen mit `pdflatex`, erzeugt *.pdf-Datei



- Wenn LaTeX beim Setzen auf einen Fehler trifft, stoppt es den Vorgang und fragt den Benutzer
 - Wenn die Fehlermeldung unklar ist, einfach [Eingabe] drücken → LaTeX versucht, den Fehler selbst zu beheben
 - Aussehen der PDF-Datei deutet dann meist auf den Fehler hin
 - Wenn viele Fehler auftreten: „S“ eingeben (scrollmode)
 - Wie dauerhaftes Drücken von [Eingabe]

- Sonderzeichen: `\$ # ^ _` muss man „escapen“, um sie benutzen zu können
 - `\$` erzeugt `$`
 - `\#` erzeugt `#`
 - `\textbackslash{}` erzeugt `\`
 - `\textasciitilde{}` erzeugt `~` (oder: `\verb#\~#`)
 - `\^{}` erzeugt `^` (oder: `\verb#\^#`)
 - `\_` erzeugt `_`
- Befehle i. d. R. mit `{}` abschließen: `\today{}`

Beispiel für Weglassen der Klammern {}:

Heute ist der 35. Mai 2012. Oder:
 Heute ist der 35. Mai 2012. Falsch
 ist: Am 35. Mai 2012 regnet es.
 Richtig: Am 35. Mai 2012 scheint
 die Sonne. Oder: Am 35. Mai 2012
 schneit es.

Heute ist der `\today`.
 Oder: Heute ist der `\today` .
 Falsch ist:
 Am `\today` regnet es.
 Richtig:
 Am `\today{}` scheint die Sonne.
 Oder: Am `\today\` schneit es.

(Quelle: LaTeX2e-Kurzbeschreibung, Version 3.0 von Marco
 Daniel, Patrick Gundlach, Walter Schmidt, Jörg Knappen,
 Hubert Partl und Irene Hyna)

- Leerzeichen / Leerzeilen (1)
 - Leerzeichen, Tabulatoren und Zeilenumbrüche werden alle als (ein) Leerzeichen interpretiert
 - auch mehrere davon (außer Umbrüche) werden zu einem Leerzeichen
 - Absatzumbruch durch Einfügen einer (oder mehrerer) Leerzeilen
 - keine Befehle wie `\newline` oder `\\` zum Erzwingen des Umbruchs verwenden
 - neue Seite: `\pagebreak`

- Leerzeichen / Leerzeilen (2)
 - Folgende Texte erzeugen das gleiche Dokument:

Abc abc abc

Abc Abc Bcad

Abcda abc afb

Abc abc abc

Abc Abc Bcad

Abcda abc afb

- Befehle und Gruppen
 - LaTeX-Kommandos sind entweder einzelne **Befehle** wie `\today{}` oder
 - Gruppen, die mit `\begin{gruppe}` begonnen und mit `\end{gruppe}` beendet werden

`\begin{center}`

Abgesetzter zentrierter Text

`\end{center}`

- Dokumentstruktur

- einheitlicher Aufbau:

```
\documentclass{article}      % oder andere Klasse
% Präambel
% Befehle wie \usepackage{...} für LaTeX-Pakete
\begin{document}
% Dokument-Inhalt
Mein erstes Beispiel-Dokument.
\end{document}
```

- **Klasse:** definiert globale Dokumenten-Eigenschaften (Layout)
 - **Pakete:** ergänzen Zusatzfunktionen

- Aufzählungen (auch geschachtelt)

- Bullet-Listen:

```
\begin{itemize}  
\item Absatz 1  
\item Absatz 2  
\end{itemize}
```



- Absatz 1
- Absatz 2

- Nummeriert:

```
\begin{enumerate}  
\item Absatz 1  
\item Absatz 2  
\end{enumerate}
```



1. Absatz 1
2. Absatz 2

- Einfache Auszeichnungen
 - `\textbf{Text}`: **fett**
 - `\emph{Text}`: hervorgehoben (*kursiv*)
 - `\underline{Text}`: unterstrichen
- Bilder und Tabellen: über `figure`-, `table`- und `tabular`-Umgebungen sowie den Befehl `\includegraphics{bilddatei}` (ohne Endung)
- Bild-/Tabellen-Unterschrift mit `\caption{Text.}`
- Labels: `\label{label}`, Verweise: `\ref{label}`, `\pageref{label}` (Seitenzahl)

- Literaturverwaltung über separates Tool (bibtex oder biber)
- Quellen in *.bib-Dateien mit eigener Syntax verwalten
- Typische Sequenz, um Dokument dokument.tex zu bauen:

```
pdflatex dokument  
bibtex dokument  
pdflatex dokument  
pdflatex dokument
```

- Syntax in den *.bib-Dateien

```
@article{CS03,
  author = "Carrier, Brian and Spafford, Eugene H.",
  title = "Getting physical with the digital
          investigation process",
  journal = "International Journal of Digital Evidence",
  volume = 2,
  year = 2003
}
```

- CS03 ist ein BibTeX-Label → `\cite{CS03}`
- Beispiele in `literatur/literatur.bib`

- BibTeX-Tipps

- für Bücher: Google-Suche nach „ISBN to Bibtex“
→ <http://manas.tungare.name/software/isbn-to-bibtex/> (oder andere Konverter-Seiten)
- für Artikel: z. B. auf der ACM-DL-Seite (<http://dl.acm.org>), Beitrag suchen, Detailseite aufrufen, BibTeX-Export

Tools and Resources



[Save to Binder](#)



Export Formats:

[BibTeX](#) [EndNote](#) [ACM Ref](#)

Share:



- Ausprobieren
- Mit funktionierenden Beispielen starten (z. B. mit der Vorlage; siehe auch folgende Folien)
- auf den Inhalt konzentrieren / nicht durch Layout-Fragen ablenken lassen
- ggf. nach speziellen Paketen suchen; LaTeX bringt umfangreiche Dokumentation mit:
`texdoc paketname` (Mac, Linux)

- für mehr Komfort: LaTeX-IDE einsetzen
 - TeXShop (Mac),
<http://pages.uoregon.edu/koch/texshop/>
 - TeXstudio (Windows, Mac, Linux u. a.),
<http://texstudio.sourceforge.net/>
 - Features:
 - Aufrufe von pdflatex, bibtex/biber und anderen Tools über Tastenkombination
 - parallele Ansicht von Editor und Vorschau
 - SyncTeX
 - ggf. Debugging-Hilfe

TeXShop

Ablage Bearbeiten Quelltext Makros Setzen Vorschau Fenster Hilfe

fom-seminar.tex

Setzen LaTeX Makros Markierungen Vorlagen

1 %-----
2 % Define document and include general packages
3 %-----
4 \documentclass[12pt,oneside,titlepage,listof=totoc,bibliography=totoc]{scrartcl}
5 \usepackage[utf8]{inputenc}
6 \usepackage[ngerman]{babel}
7 \usepackage[babel,german=quotes]{csquotes}
8 \usepackage[T1]{fontenc}
9 \usepackage{fancyhdr}
10 \usepackage{fancybox}
11 \usepackage[a4paper, left=4cm, right=2cm, top=2.8cm, bottom=2.3cm]{geometry}
12 \usepackage{graphicx}
13 \usepackage{color}tbl
14 \usepackage{array}
15 \usepackage{float} %Positionierung von Abb. und Tabellen mit [H] erzwingen
16 \usepackage{footnote}
17 \usepackage{caption}
18 \usepackage{mdwlist}
19 \usepackage{amssymb}
20 \usepackage{mathptmx}
21 \usepackage{amsmath}
22 \usepackage[table]{xcolor}
23 \usepackage{marvosym} % Verwendung von Symbolen, z.B. perfekt
24 \usepackage{colorlinks=true,linkcolor=black}{hyperref}
25 \definecolor{darkblack}{rgb}{0,0,0}
26 \hypersetup{pdfTeX=true, colorlinks=true, breaklinks=true, linkcolor=darkblack, urlcolor=darkblack}
27 \usepackage{times}
28 \fontfamily{ptm}\selectfont
29
30 %Pakete für Tabellen
31 \usepackage{epstopdf}
32 \usepackage{nicefrac} % Brüche
33 \usepackage{multirow}
34 \usepackage{rotating} % vertikal schreiben
35 \usepackage{colortbl}
36 \usepackage{mdwlist}
37
38 \definecolor{dunkelgrau}{rgb}{0.8,0.8,0.8}
39 \definecolor{hellgrau}{rgb}{0.8,0.7,0.99}
40
41 % sauber formatierter Quelltext
42 \usepackage{listings}
43 \lstset{numbers=left,
44 > numberstyle=\tiny,
45 > numbersep=5pt,
46 > breaklines=true,
47 > showstringspaces=false,
48 > frame=1,
49 > xleftmargin=5pt,
50 > xrightmargin=5pt,
51 > basicstyle=\ttfamily\scriptsize,
52 > stepnumber=1,
53 > keywordstyle=\color{blue}, % keyword style
54 > commentstyle=\color{dkgreen}, % comment style
55 > stringstyle=\color{mauve} % string literal style
56 }
57
58 % Bibl~~at~~ex
59 \usepackage{

fom-seminar.pdf

Größe 128 Seite 2 von 5

Inhaltsverzeichnis

1 Eine Section 1
1.1 Eine Subsection 2
1.1.1 Eine Subsubsection 2
1.1.2 Noch eine Subsubsection 3
2 eine weitere Section 3
3
3
ne Section
Tilde, \ ist der Backslash. ^ oder ^ ist das Dach. \$ ist das Dollar-Zeichen. # ist die
eine Section. Hier lässt sich auch gut Blindtext integrieren. Lorem ipsum dolor sit
onsectetur adipiscing elit. In egestas tellus in libero consectetur, non finibus odio
it. Etiam iaculis, ex sed interdum tristique, est eros dignissim risus, vel consectetur
r in urna. Nullam tempus sed elit vel laoreet. Quisque non enim lacus. Class aptent
aciosqu ad litora torquent per conubia nostra, per inceptos himenaeos.
kennzeichnet man durch eine Leerzeile. Ut malesuada commodo nunc ac ultr
congue malesuada justo sed condimentum. Morbi ornare, nisi nec sollicitudin
rper, libero dolor mattis enim, nec dictum quam dolor et felis. Curabitur iaculis
e diam scelerisque, nec ullamcorper elit molestie. Aliquam aliquet sit amet mauris
estibulum. Cras commodo a lectus vel vestibulum. In justo quam, auctor in rutrum
cipit vitae leo. Nunc viverra dapibus lectus. In massa quam, ullamcorper nec mas
onsequat sollicitudin odio. Interdum et malesuada fames ac ante ipsum primis in
faucibus. Suspendisse pretium et lorem a molestie. Cras nisl nibh, ultricies ac erat porta,
gravida egestas arcu. Maecenas ac augue volutpat, pharetra augue in, dignissim lectus.
Maecenas nec dictum neque, eu tempus ante.
Text kann man auch fett und kursiv auszeichnen, z. B.: Aliquam erat volutpat. Donec
condimentum purus sed ligula feugiat, eu porttitor nisi sodales. Vivamus rutrum
felis id erat venenatis varius. Nunc volutpat vulputate tortor, ut porta dolor convallis ac.
Nullam posuere ultricies porttitor. Ut laoreet ultrices vestibulum. Praesent venenatis porta

(TeXShop, OS X)

05.03.2015

IT-Infrastruktur, SS 2015, Hans-Georg Eßer

Folie B-23

- Datei `fom-seminar.zip` (im Campus-System) herunterladen und entpacken; erzeugt Ordner `fom-seminar/`.
- Darin liegt eine Datei `fom-seminar.tex` – das ist das Hauptdokument
- Literaturverzeichnis:
`literatur/literatur.bib`
- Übersetzen mit `./compile.sh` (Linux, OS X) bzw. `compile.bat` (Windows)

- Anpassungen:

```

112 % Autor
113 \newcommand{\myAutor}{Max Mustermann}
114 % Titel der Arbeit
115 \newcommand{\myTitel}{Meine \LaTeX{}-Seminararbeit}
116 % Betreuer
117 \newcommand{\myBetreuer}{Dipl.-Math. Dipl.-Inform. Hans-Georg Eßer}
118 % Matrikelnummer
119 \newcommand{\myMatrikelNr}{123456}
120 % Ort
121 \newcommand{\myOrt}{Nürnberg}
122 % Datum der Abgabe
123 \newcommand{\myAbgabeDatum}{15. Januar 2015}
124 % Semesterzahl
125 \newcommand{\mySemesterZahl}{4}
126 % Name der Hochschule
127 \newcommand{\myHochschulName}{FOM Hochschule}
128 % Studiengang
129 \newcommand{\myStudiengang}{Wirtschaftsinformatik}
130 % Art der Arbeit
131 \newcommand{\myThesisArt}{Seminararbeit}

```

FOM Hochschule



Berufsbegleitender Studiengang
Wirtschaftsinformatik, 4. Semester

Seminararbeit
im Fach „IT-Infrastruktur“

über das Thema

Meine $\text{\LaTeX}$ -Seminararbeit

Betreuer: Dipl.-Math. Dipl.-Inform. Hans-Georg Eßer

Autor: Max Mustermann
Matrikelnr.: 123456

Abgabe: 15. Januar 2015

1

Inhaltsverzeichnis

1 Eine Section	1
1.1 Eine Subsection	2
1.1.1 Eine Subsubsection	2
1.1.2 Noch eine Subsubsection	3
2 Und eine weitere Section	3
Literaturverzeichnis	3

1 Eine Section

Dies ist eine Section. Hier lässt sich auch gut Blindtext integrieren. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In egestas tellus in libero consectetur, non finibus odio hendrerit. Etiam iaculis, ex sed interdum tristique, est eros dignissim risus, vel consectetur ex dolor in urna. Nullam tempus sed elit vel laoreet. Quisque non enim lacus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Absätze kennzeichnet man durch eine Leerzeile. Ut malesuada commodo nunc ac ultricies. In congue malesuada justo sed condimentum. Morbi ornare, nisi nec sollicitudin ullamcorper, libero dolor mattis enim, nec dictum quam dolor et felis. Curabitur iaculis nisi vitae diam scelerisque, nec ullamcorper elit molestie. Aliquam aliquet sit amet mauris luctus vestibulum. Cras commodo a lectus vel vestibulum. In justo quam, auctor in rutrum vel, suscipit vitae leo. Nunc viverra dapibus lectus. In massa quam, ullamcorper nec massa at, consequat sollicitudin odio. Interdum et malesuada fames ac ante ipsum primis in faucibus. Suspendisse pretium et lorem a molestie. Cras nisl nibh, ultricies ac erat porta, gravida egestas arcu. Maecenas ac augue volutpat, pharetra augue in, dignissim lectus. Maecenas nec dictum neque, eu tempus ante.

Text kann man auch **fett** und *kursiv* auszeichnen, z. B.: **Aliquam erat volutpat. Donec condimentum purus sed ligula feugiat, eu porttitor nisi sodales. Vivamus rutrum felis id erat venenatis varius.** Nunc volutpat vulputate tortor, ut porta dolor convallis ac. Nullam posuere ultricies porttitor. Ut laoreet ultrices vestibulum. Praesent venenatis porta bibendum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. *Nulla pellentesque libero nec pretium tempus. Proin at purus risus. Vestibulum ex nibh, hendrerit sed lectus eget, dictum feugiat nunc.*

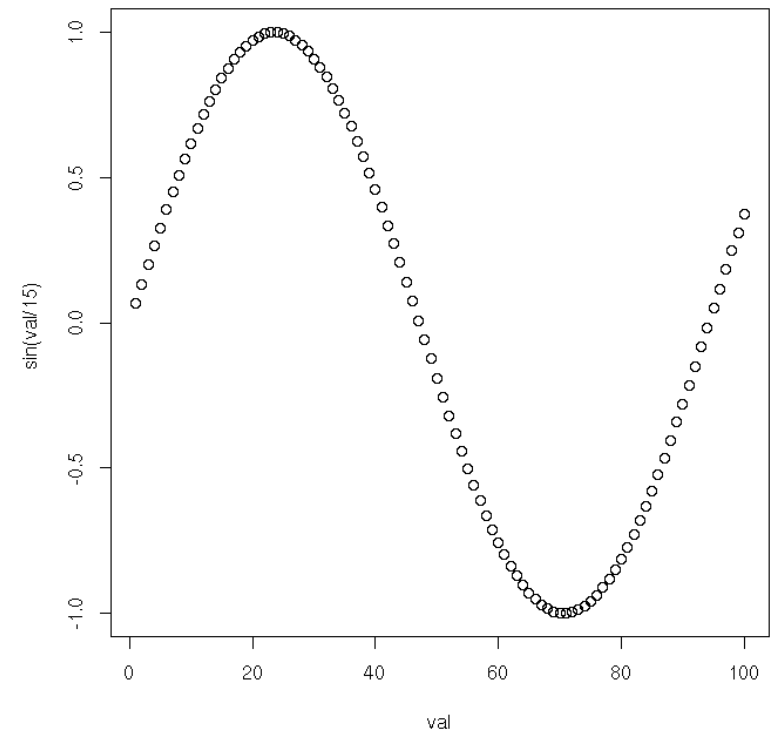
Das ist ein Zitat: [2]. Wenn eine Seitenangabe nötig ist, geht das folgendermaßen [2, S. 123]. Man kann auch mehrere Quellen gleichzeitig [2, 1] zitieren.

Statistik: Programmiersprache R

- R ist Programmiersprache
- interaktiver Modus oder R-Skripte ausführen
- IDE: R Studio
- nur mit Statistik-Grundkenntnissen sinnvoll nutzbar
- R ist über Zusatzpakete erweiterbar

R: Interessante Sprach-Features

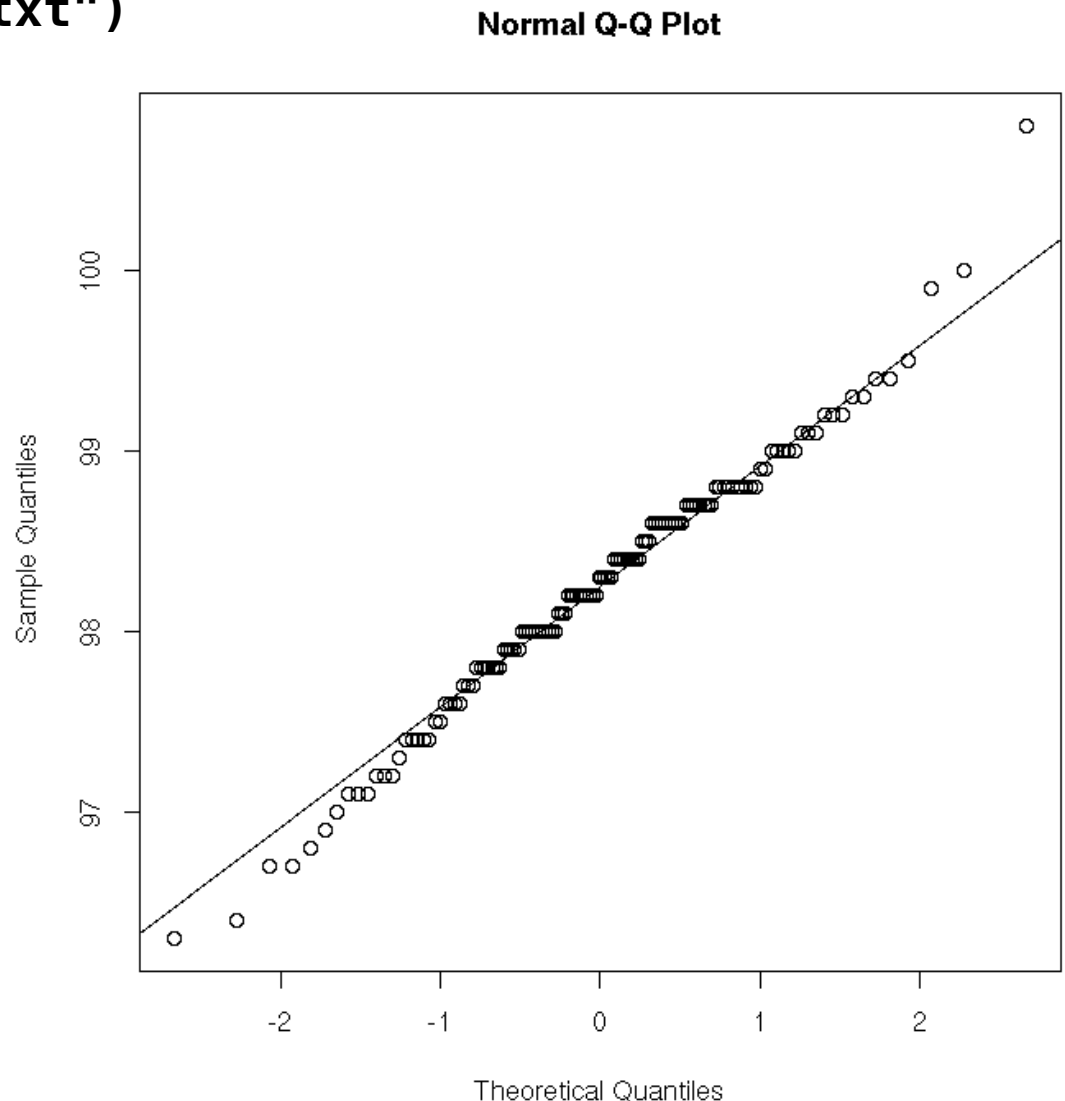
```
> x = 1:20
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> y = x+0.5
> y
[1] 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5 10.5 11.5 12.5
[13] 13.5 14.5 15.5 16.5 17.5 18.5 19.5 20.5
> index = c(1,2,3,11,12,13)
> x[index]
[1] 1 2 3 11 12 13
> y[index]
[1] 1.5 2.5 3.5 11.5 12.5 13.5
> val = 1:100; plot(val,sin(val/15))
```



```
> temp = read.table("normtemp.txt")
> names(temp)
[1] "V1" "V2" "V3"
> degreesF = temp$V1
> qqnorm(degreesF)
> qqline(degreesF)
> plot(density(degreesF))
> qqnorm(degreesF)
> qqline(degreesF)
```

normtemp.txt:

96.3	1	70
96.7	1	71
96.9	1	74
97.0	1	80
97.1	1	73
...		



```
> plot(density(degreesF))
> shapiro.test(degreesF)
```

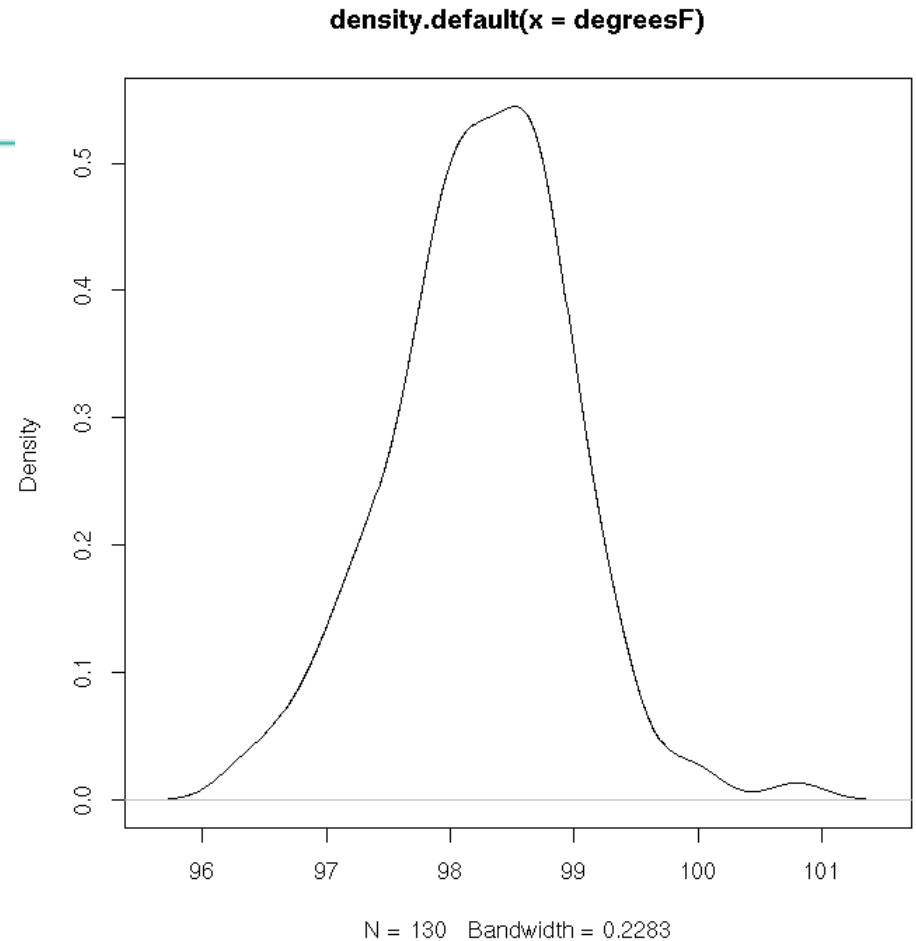
Shapiro-Wilk normality test

```
data: degreesF
W = 0.9866, p-value = 0.2332
```

```
> t.test(degreesF, mu=98.6,
  alternative="two.sided")
```

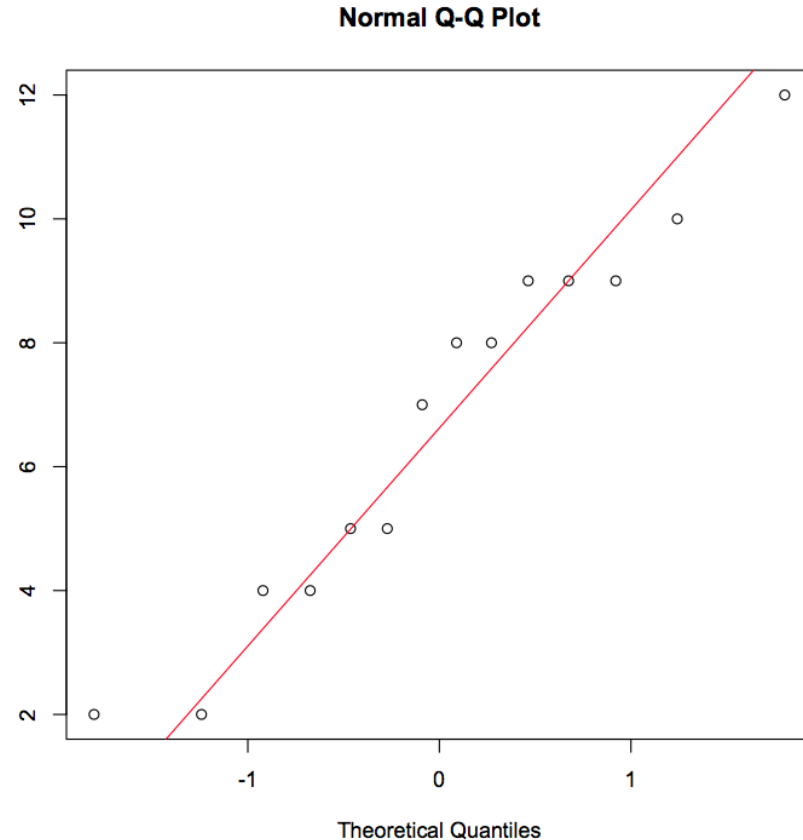
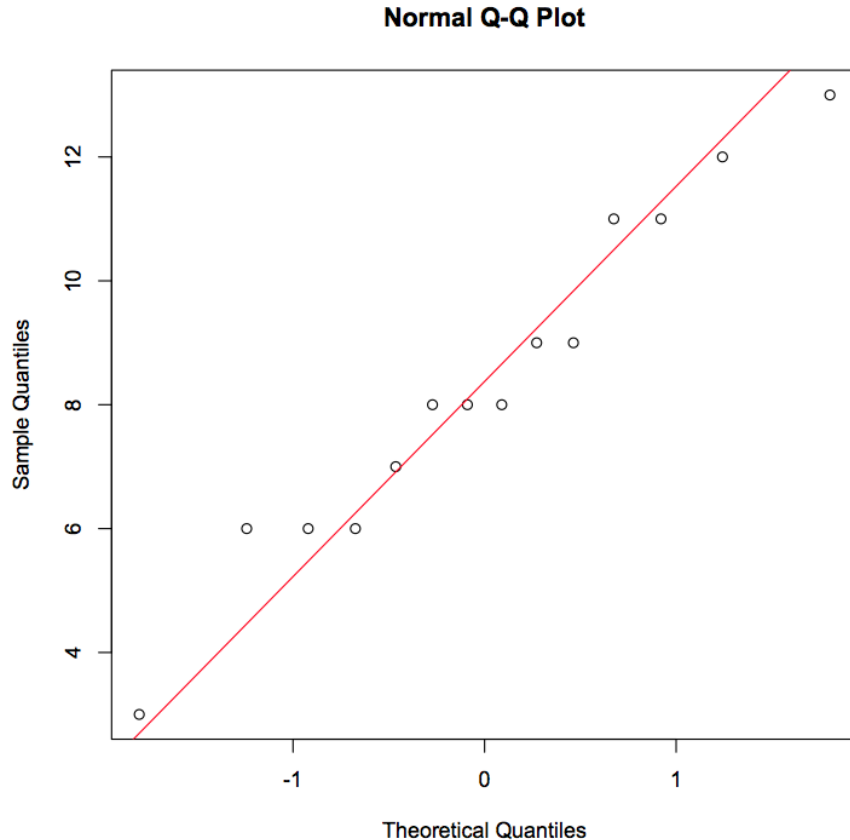
One Sample t-test

```
data: degreesF
t = -5.4548, df = 129, p-value = 2.411e-07
alternative hypothesis: true mean is not equal to 98.6
95 percent confidence interval:
 98.12200 98.37646
sample estimates:
mean of x
 98.24923
```



R: Beispiele (3)

```
> x <- c(8,11,3,6,6,7, 9,11,12,9,8,8,6,13) # Test vorher
> y <- c(2,12,4,4,2,9,10, 7, 5,5,8,8,9, 9) # Test nachher
> qqnorm(x); qqline(x, col = 2);           # linker Plot (x)
> qqnorm(y); qqline(y, col = 2);           # rechter Plot (y)
```



Daten sind nicht
normalverteilt

```
> x <- c(8,11,3,6,6,7, 9,11,12,9,8,8,6,13) # Test vorher  
> y <- c(2,12,4,4,2,9,10, 7, 5,5,8,8,9, 9) # Test nachher  
> wilcox.test(x, y, alternative = "greater", paired = TRUE,  
  correct = TRUE)
```

Wilcoxon signed rank test with continuity correction

data: x and y

V = 61.5, p-value = 0.04129

alternative hypothesis: true location shift is greater than 0

p-value ist kleiner als 0,05 → Null-Hypothese
(„keine signifikante Verbesserung“) ablehnen

The screenshot shows the RStudio IDE interface. The main editor displays an R script with a function `count` that processes a data frame `df` and a weighting variable `wt_var`. The script includes logic for handling missing values, creating interactions, and applying weights. The environment pane on the right shows the objects `df`, `df2`, and `labels` with their dimensions. The traceback pane shows the call stack for the `count` function. The console pane at the bottom shows the execution of `printSummary(mtcars)` and the current state of the `count` function.

```

38 count <- function(df, vars = NULL, wt_var = NULL) {
39   if (is.vector(df)) {
40     df <- data.frame(x = df)
41   }
42
43   if (!is.null(vars)) {
44     vars <- as.quoted(vars)
45     df2 <- quickdf(eval.quoted(vars, df))
46   } else {
47     df2 <- df
48   }
49
50   id <- ninteraction(df2, drop = TRUE)
51   u_id <- !duplicated(id)
52   labels <- df2[u_id, , drop = FALSE]
53   labels <- labels[order(id[u_id]), , drop = FALSE]
54
55   if (is.null(wt_var) && "freq" %in% names(df)) {
56     message("Using freq as weighting variable")
57     wt_var <- "freq"
58   }
59
60   if (!is.null(wt_var)) {
61     wt_var <- as.quoted(wt_var)
62     if (length(wt_var) > 1) {
63       stop("wt_var must be a single variable", call. = FALSE)
64     }
65
66     wt <- eval.quoted(wt_var, df)[[1]]
67     freq <- vaggregate(wt, id, sum, .default = 0)

```

Environment

Object	Class	Dimensions
df	data.frame	32 obs. of 11 variables
df2	data.frame	32 obs. of 11 variables
labels	data.frame	32 obs. of 11 variables

Values

Variable	Class	Dimensions	Values
id	atomic	[1:32]	1 2 3 4 5 6 7 8 9 10 ...
u_id	logical	[1:32]	TRUE TRUE TRUE TRUE TRUE TRUE ...
vars	NULL	(empty)	
wt_var	NULL		

Traceback

```

count(df) at count.r:53
printSummary(mtcars) at utils.R:4

```

Console

```

> printSummary(mtcars)
Called from: eval(expr, envir, enclos)
Browse[1]> n
debug at /home/jjallaire/plyr/R/count.r#53: labels <- labels[order(id[u_id]), , drop = FALSE]
Browse[2]> |

```

Quelle: <http://rstudioblog.files.wordpress.com/2013/09/rstudiodebugger.png>

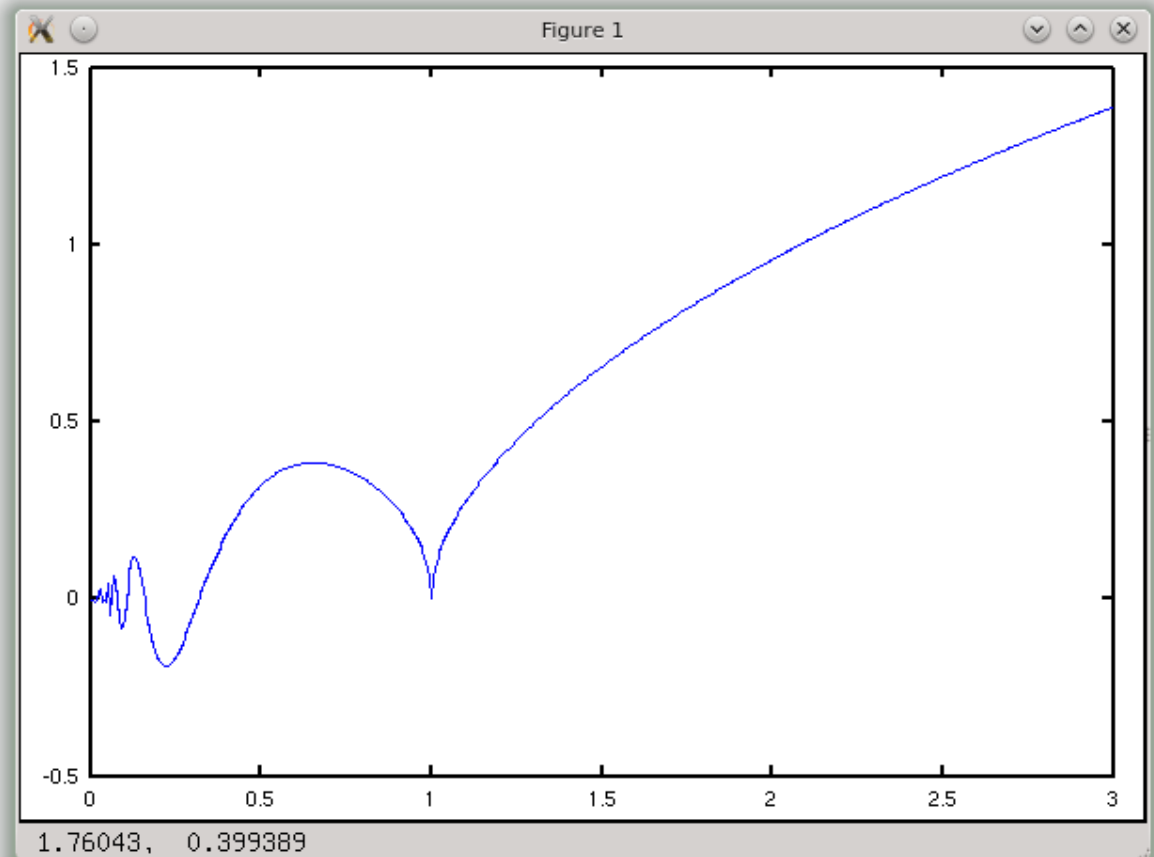
Numerik: MatLab, GNU Octave

Datenanalyse, Algorithmen-Entwicklung und Modellerstellung

- Interpolation und Regression
- Differenzierung und Integration
- Lineare Gleichungssysteme
- Fourier-Analyse
- Eigenwerte und Singulärwerte
- Gewöhnliche Differenzialgleichungen
- Dünnbesetzte Matrizen

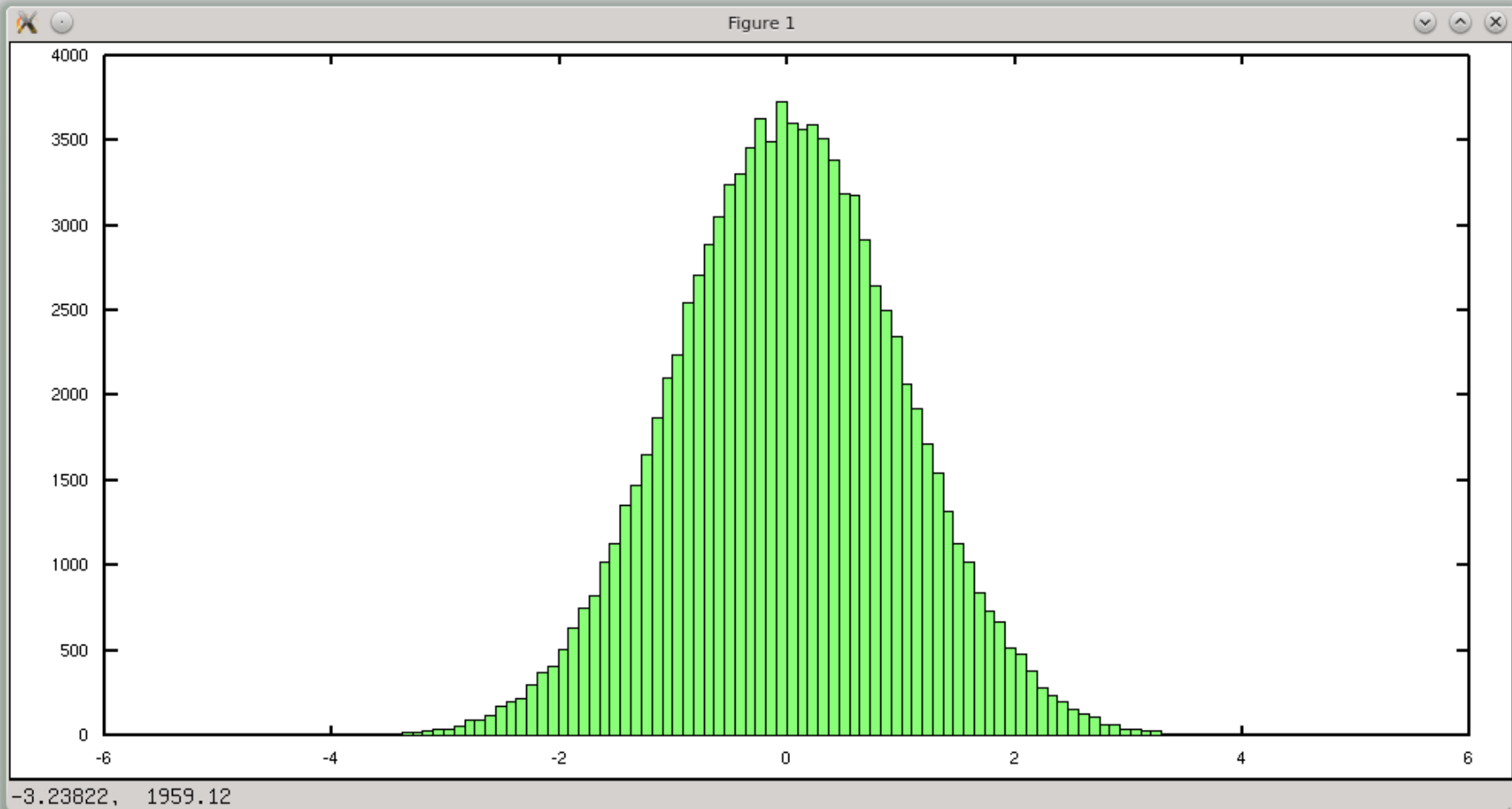
```
octave:1> function y = f(x)
>   y = x .* sin (1./x) .* sqrt (abs (1-x));
> endfunction
octave:2> x = 0:0.005:3;
octave:3> plot (x, f(x));
octave:4> quad ("f", 0, 3)
ABNORMAL RETURN FROM DQAGP
ans = 1.9819
```

$$f(x) = x \sin\left(\frac{1}{x}\right) \sqrt{|1-x|}$$



Octave, Beispiel: Histogramm

```
octave:1> hist(randn(100000, 1), 100)
```



```
octave:48> A = [ 1 1 1; 0 1 1; 0 0 1]
```

```
A =
```

```

1  1  1
0  1  1
0  0  1
```

```
octave:49> inv (A)
```

```
ans =
```

```

1  -1  -0
0   1  -1
0   0   1
```

```
octave:50> A * inv(A)
```

```
ans =
```

```

1  0  0
0  1  0
0  0  1
```

Computer-Algebra-Systeme

- anders als numerische Software führen CAS *symbolische* Berechnungen durch
- mathematische Ausdrücke, Matrizen etc. können Variablen enthalten, z. B.
 - Matrix mit Variable invertieren
 - symbolisches Differenzieren und Integrieren
- Software
 - Maple (kommerziell), Axiom (freie Software)
 - Wolfram Alpha (Webseite)

Matrix-Invertierung mit Wolfram Alpha



invert matrix((1,1,1),(0,1,x),(0,0,x+1))



Examples Random

Assuming "invert" is referring to linear algebra | Use "invert matrix" as a [math function](#) instead

Input:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & x \\ 0 & 0 & x+1 \end{pmatrix}^{-1} \quad (\text{matrix inverse})$$

Result:

$$\begin{pmatrix} 1 & -1 & \frac{x-1}{x+1} \\ 0 & 1 & -\frac{x}{x+1} \\ 0 & 0 & \frac{1}{x+1} \end{pmatrix}$$



integrate f(x) = (x^2+2x sin(a x))/a^x



Examples ↗ Random

Indefinite integral:

Step-by-step solution

$$\int \frac{x^2 + 2 x \sin(ax)}{a^x} dx = \left(a^{-x} \left(-\left(a^2 + \log^2(a) \right)^2 \left(x^2 \log^2(a) + 2 x \log(a) + 2 \right) - \right. \right. \\ \left. \left. 2 \log^3(a) \left(a^2 x \log(a) - a^2 + x \log^3(a) + \log^2(a) \right) \sin(ax) - \right. \right. \\ \left. \left. 2 a \log^3(a) \left(a^2 x + x \log^2(a) + 2 \log(a) \right) \cos(ax) \right) \right) / \\ \left(\log^3(a) \left(a^2 + \log^2(a) \right)^2 \right) + \text{constant}$$

$\log(x)$ is the natural logarithm

Geo-Informationssysteme (GIS)

- Erfassung, Bearbeitung, Organisation, Analyse und Präsentation räumlicher Daten
- verwenden eigene Auszeichnungssprachen, um Daten zu speichern; z. B.

- KML (Keyhole Markup Language), verwendet von Google Earth

- GPS Exchange Format (GPX)

(Quelle: Wikipedia)

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <Placemark>
    <name>Zürich</name>
    <description>Zürich</description>
    <Point>
      <coordinates>8.55,47.3666667,0
      </coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

Google Search Gallery  +Hans-Georg

Maps Gallery What's Hot Explore Add your maps

tourist attractions in munich germany

Bayerische Staatskanzlei

Bavaria's governor keeps his office in the humongous Bayerische Staatskanzlei which takes up the entire eastern flank of the Hofgarten. It's a strikingly modern glass palace built around the restored centre section of the Army Museum that for years stood as a ruined anti-war memorial.

Details from Google Maps
 Franz-Josef-Strauß-Ring 1, 80539 Munich, Germany
www.bayern.de
 089 21650
 4.1 ★★★★★ Google+ page

NEUHAUSEN-NYPHENBURG
 BOGENHAUSEN
 STADT-LEHEL
 SCHWANTHALERHÖHE

Map data ©2014 GeoBasis-DE/BKG (©2009), Google

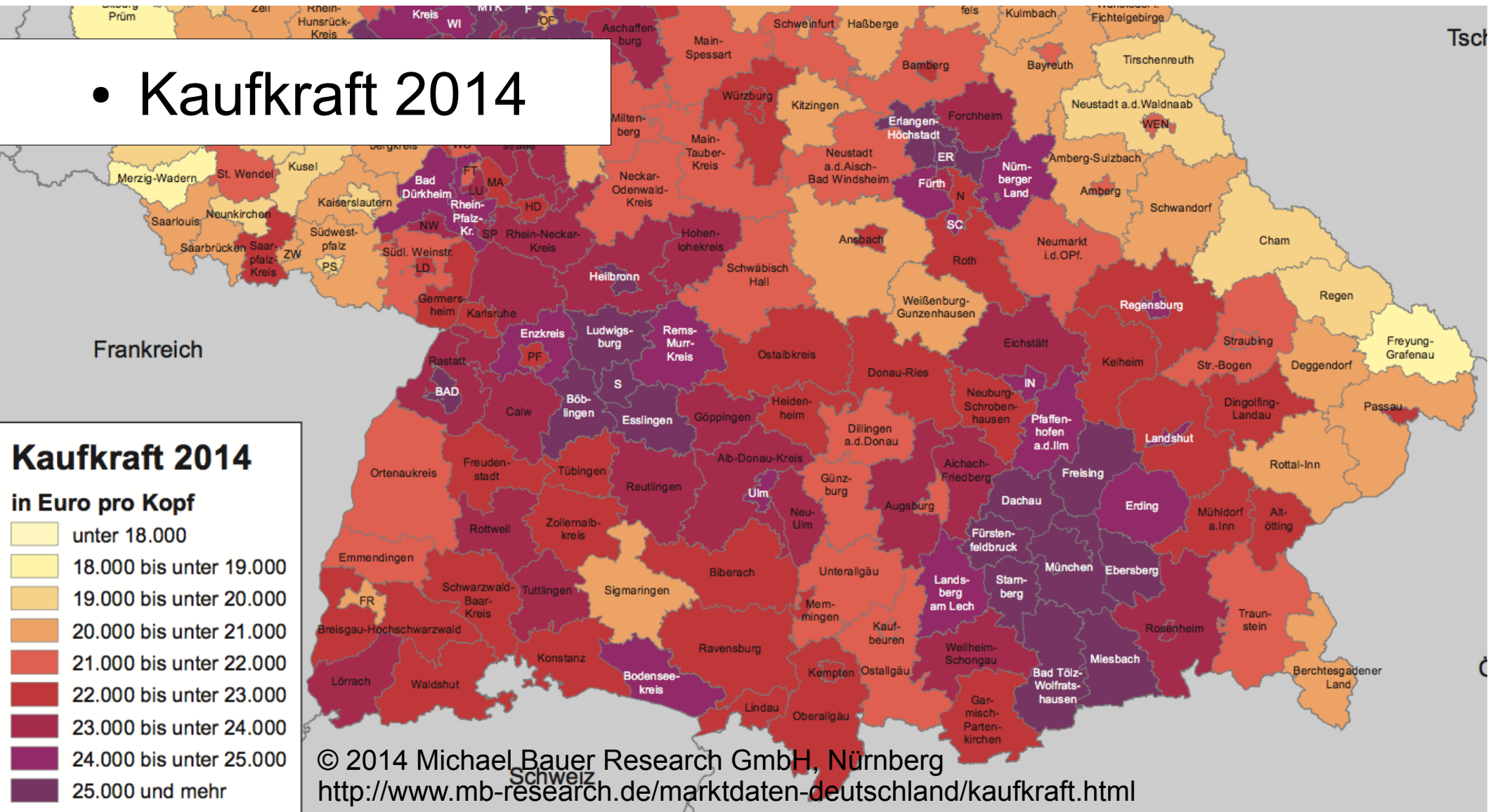
• Nielsen-Gebiete

- Gebiet 1: Hamburg
Bremen
Schleswig-Holstein
Niedersachsen
- Gebiet 2: Nordrhein-Westfalen
- Gebiet 3a: Hessen
Rheinland-Pfalz
Saarland
- Gebiet 3b: Baden-Württemberg
- Gebiet 4: Bayern
- Gebiet 5+6: Berlin
Mecklenburg-Vorpommern
Brandenburg
Sachsen-Anhalt
- Gebiet 7: Thüringen
Sachsen



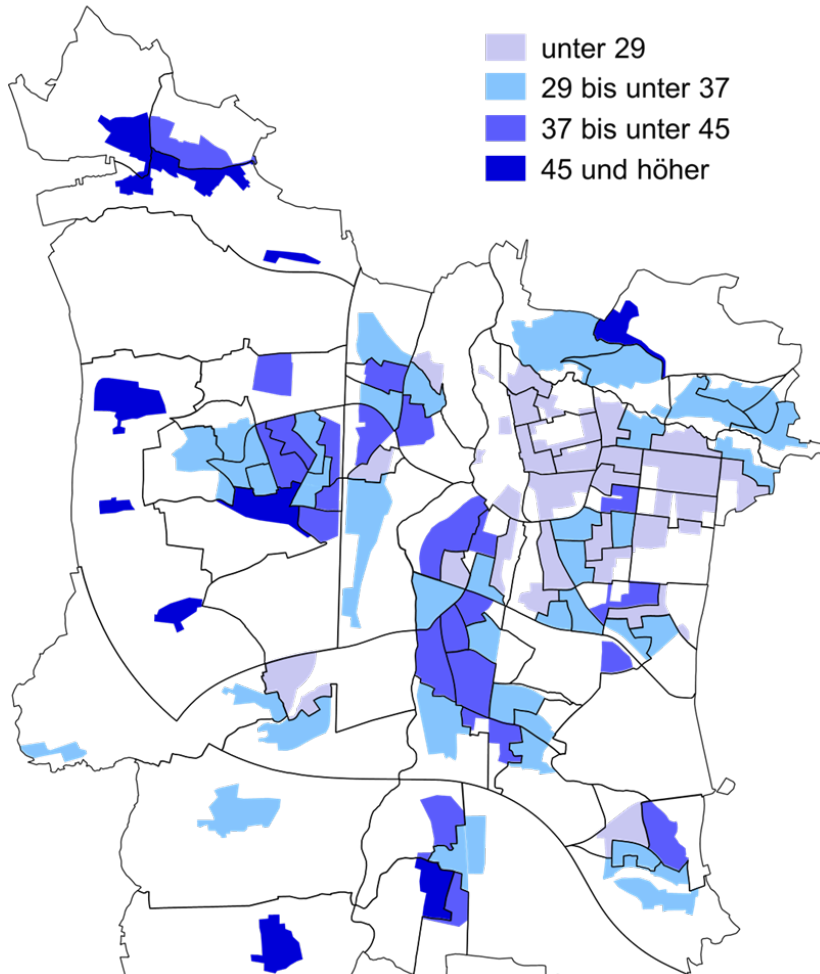
Bild: <http://www.makz.de/nielsengebiete.html>

- Kaufkraft 2014



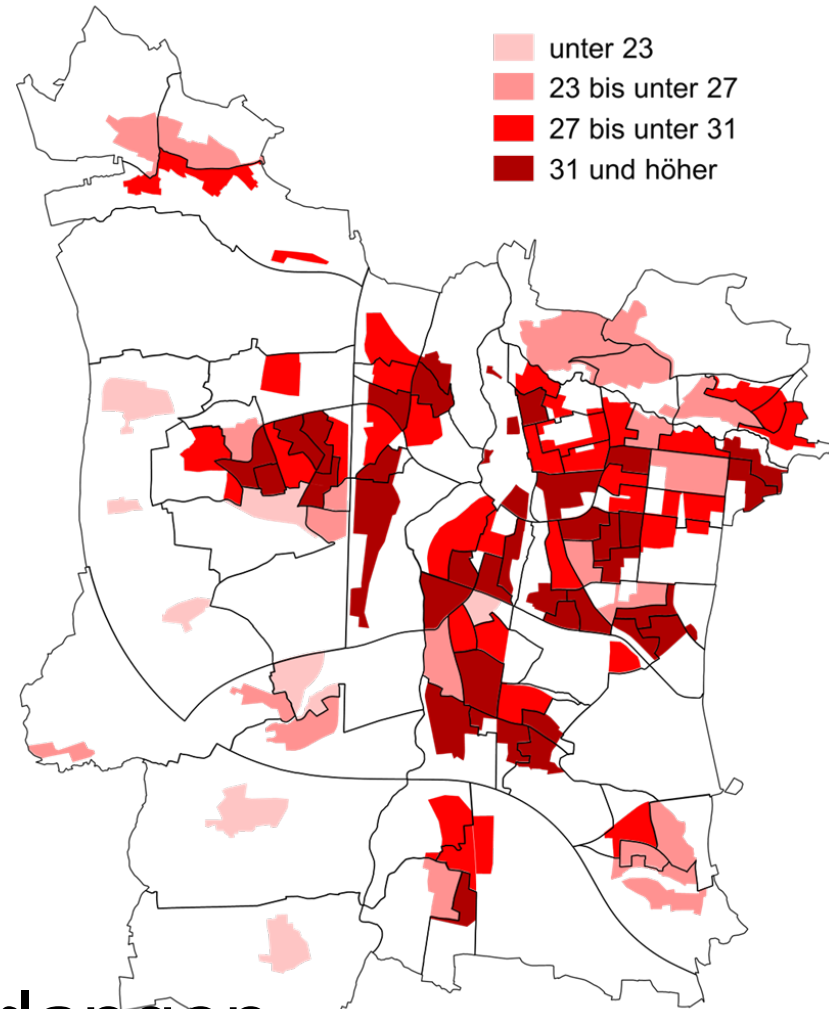
Stimmenanteile der CSU

Stadtratswahl 2014 - Stimmenanteile in Prozent



Stimmenanteile der SPD

Stadtratswahl 2014 - Stimmenanteile in Prozent



- Stadtratswahl 2014, Erlangen

Bilder: <http://www.erlangen.de/tabid-1785/>

Versionierung

- Aufgabe: Änderungen an Dokumenten (oft: an Quellcode-Dokumenten) verwalten
- Eine Auswahl von Tools
 - 1972: Source Code Control System (SCCS)
 - 1982: Revision Control System (rcs)
 - 1990: Concurrent Versions System (cvs)
 - 2000: Subversion (svn)
 - 2005: Git
 - 2005: Mercurial (hg)

- Naiver Ansatz: Versionen in separaten Ordnern speichern
- Vergleich der Versionen mit `diff`

```
$ find . -name '*.tex' -exec ls -l {} \;
-rw-r--r--@ 1 ... 823127 31 Okt 2013 ./2013-10-31/diss-hgesser-ulix.tex
-rw-r--r--@ 1 ... 896574 7 Jan 2014 ./2014-01-07/diss-hgesser-ulix.tex
-rw-r--r--@ 1 ... 1005074 2 Mai 2014 ./2014-05-02/diss-hgesser-ulix.tex
-rw-r--r--@ 1 ... 1005063 3 Mai 2014 ./2014-05-03/diss-hgesser-ulix.tex
-rw-r--r--@ 1 ... 1594695 25 Aug 22:26 ./2014-08-25/diss-hgesser-ulix.tex
$ diff 2014-05-02/diss-hgesser-ulix.tex 2014-05-03/diss-hgesser-ulix.tex
12312d12311
< char *buf;
$ _
```

- Nachteile des naiven Ansatzes
 - alle Dateien mehrfach vollständig gespeichert
 - keine Zusammenfassung der Änderungen von einer Version zur nächsten
 - keine automatische Zuteilung von Versions- bzw. Revisionsnummern
- Alternative: spezialisierte Versionsverwaltungen
 - speichern jeweils nur die Änderungen (Delta)
 - legen Versionsnummern und Kommentare an
 - multi-user-tauglich

- Alle Versionsverwaltungen unterstützen (mindestens) diese drei Operationen:
 - **ADD:** Eine Datei (oder einen ganzen Ordner) hinzufügen (= unter die Verwaltung stellen)
 - **CHECK-IN / COMMIT:** Die aktuelle Arbeitsfassung „einchecken“ (sichert die Datei unter einer neuen Revisionsnummer ins Archiv)
 - **CHECK-OUT:** Die letzte oder eine ältere Revision „aus-checken“, also aus dem Archiv wiederherstellen

- hier nur lokaler Einsatz (kein Repository im Netzwerk)
- Verzeichnis für die Nutzung von Mercurial vorbereiten:

```
hg init
```

- Datei hinzufügen:

```
hg add Dateiname
```

- alle aktualisierten Dateien einchecken:

```
hg commit
```

öffnet Editor-Fenster, Eingabe einer Zusammenfassung

```
Erste Revision; nur Datei test.tex
```

```
HG: Bitte gib eine Versions-Meldung ein. Zeilen beginnend mit  
    'HG:' werden
```

```
HG: entfernt. Leere Versionsmeldung wird das Übernehmen  
    abbrechen.
```

```
HG: --
```

```
HG: Benutzer: Hans-Georg Esser <h.g.esser@gmx.de>
```

```
HG: Zweig 'default'
```

```
HG: Hinzugefügt test.tex
```

- Neue Dateien im Ordner entdecken:

```
$ hg status  
? literatur.bib
```

- Andere Revision aus-checken:

```
$ hg co 0  
1 Dateien aktualisiert, 0 Dateien  
zusammengeführt, 0 Dateien entfernt,  
0 Dateien ungelöst
```

(hat im Arbeitsverzeichnis Version 1 durch die
ältere Version 0 ersetzt)

- Verlauf der Änderungen anzeigen:

```
$ hg annotate -r0 test.tex
0: \documentclass{article}
0: \begin{document}
0: Hello World
0: \end{document}
$ hg annotate -r1 test.tex
0: \documentclass{article}
0: \begin{document}
0: Hello World
1: One more line
0: \end{document}
$ hg annotate -r2 test.tex
0: \documentclass{article}
0: \begin{document}
2: Old two lines deleted, one new line
0: \end{document}
```

- Vergleich zweier Versionen:

```
$ hg diff -r0:2 test.tex
diff -r 34972b9741f6 -r 3e988791a22a test.tex
--- a/test.tex Mon Dec 01 11:37:11 2014 +0100
+++ b/test.tex Mon Dec 01 11:54:00 2014 +0100
@@ -1,5 +1,5 @@
  \documentclass{article}
  \begin{document}
-Hello World
+Old two lines deleted, one new line
  \end{document}
```


- Protokoll:

```
$ hg log
```

```
Änderung:      2:3e988791a22a
```

```
Marke:         tip
```

```
Nutzer:        Hans-Georg Esser <h.g.esser@gmx.de>
```

```
Datum:         Mon Dec 01 11:54:00 2014 +0100
```

```
Zusammenfassung: Zwei Zeilen geloescht, eine neu
```

```
Änderung:      1:8f2ae3855183
```

```
Nutzer:        Hans-Georg Esser <h.g.esser@gmx.de>
```

```
Datum:         Mon Dec 01 11:49:28 2014 +0100
```

```
Zusammenfassung: Aenderung, eine neue Zeile in test.tex
```

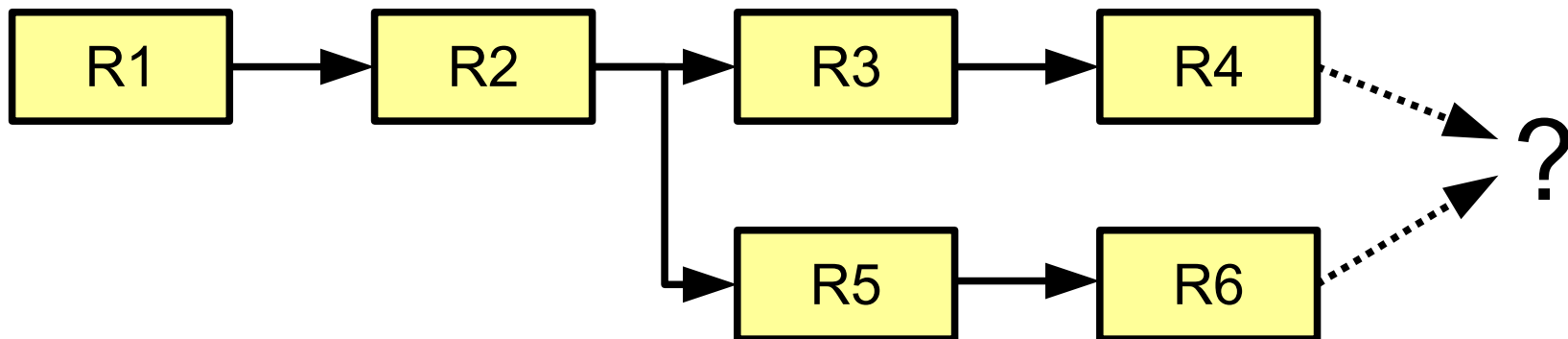
```
Änderung:      0:34972b9741f6
```

```
Nutzer:        Hans-Georg Esser <h.g.esser@gmx.de>
```

```
Datum:         Mon Dec 01 11:37:11 2014 +0100
```

```
Zusammenfassung: Erste Revision; nur Datei test.tex
```

- Durch Ändern einer älteren Version und Wieder-Ein-Checken entsteht ein paralleler Zweig – und damit ein Baum



- Entwicklung kann nun parallel weiter laufen
- Zweige können auch wieder vereinigt werden

- Weiterentwicklung von Code in mehreren Zweigen
- Zweige sollen zusammengeführt werden
 - **Merge-Operation**
 - Änderungen eines Zweigs in anderen Zweig übernehmen

Rev. 0:

```
int main () {
    // Block 1
    int i = 0;
    for (; i < 10; i++) {
        printf (i);
    }

    // Block 2
    char c = 'a';
    for (; c <= 'z'; c++) {
        printf (c);
    }
}
```

Rev. 1:

```
int main () {
    // Block 1
    int i = 0;
    for (; i < 10; i++) {
        printf ("%d\n", i);    // format code fehlt!
    }

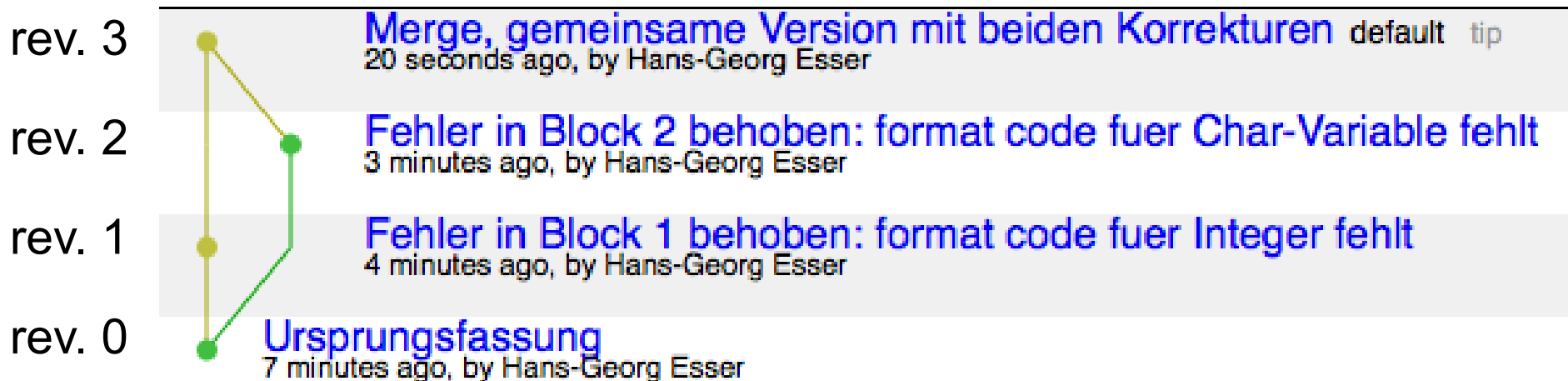
    // Block 2
    char c = 'a';
    for (; c <= 'z'; c++) {
        printf (c);
    }
}
```

Rev. 2:

```
int main () {
    // Block 1
    int i = 0;
    for (; i < 10; i++) {
        printf (i);
    }

    // Block 2
    char c = 'a';
    for (; c <= 'z'; c++) {
        printf ("%c \n", c);    // %c!
    }
}
```

```
$ hg merge -r1           # arbeite gerade mit rev. 2
Führe code.c zusammen
0 Dateien aktualisiert, 1 Dateien zusammengeführt, 0
Dateien entfernt, 0 Dateien ungelöst
(Zusammenführen von Zweigen, vergiss nicht 'hg commit'
auszuführen)
$ hg commit
```



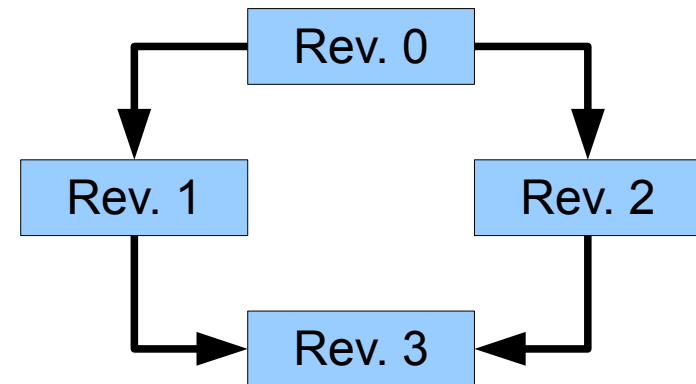
- Ergebnis:

Rev. 3:

```
int main () {  
    // Block 1  
    int i = 0;  
    for (; i < 10; i++) {  
        printf ("%d\n", i);    // format code fehlt!  
    }  
  
    // Block 2  
    char c = 'a';  
    for (; c <= 'z'; c++) {  
        printf ("%c \n", c);    // %c!  
    }  
}
```

Änderung
aus Rev. 1

Änderung
aus Rev. 2

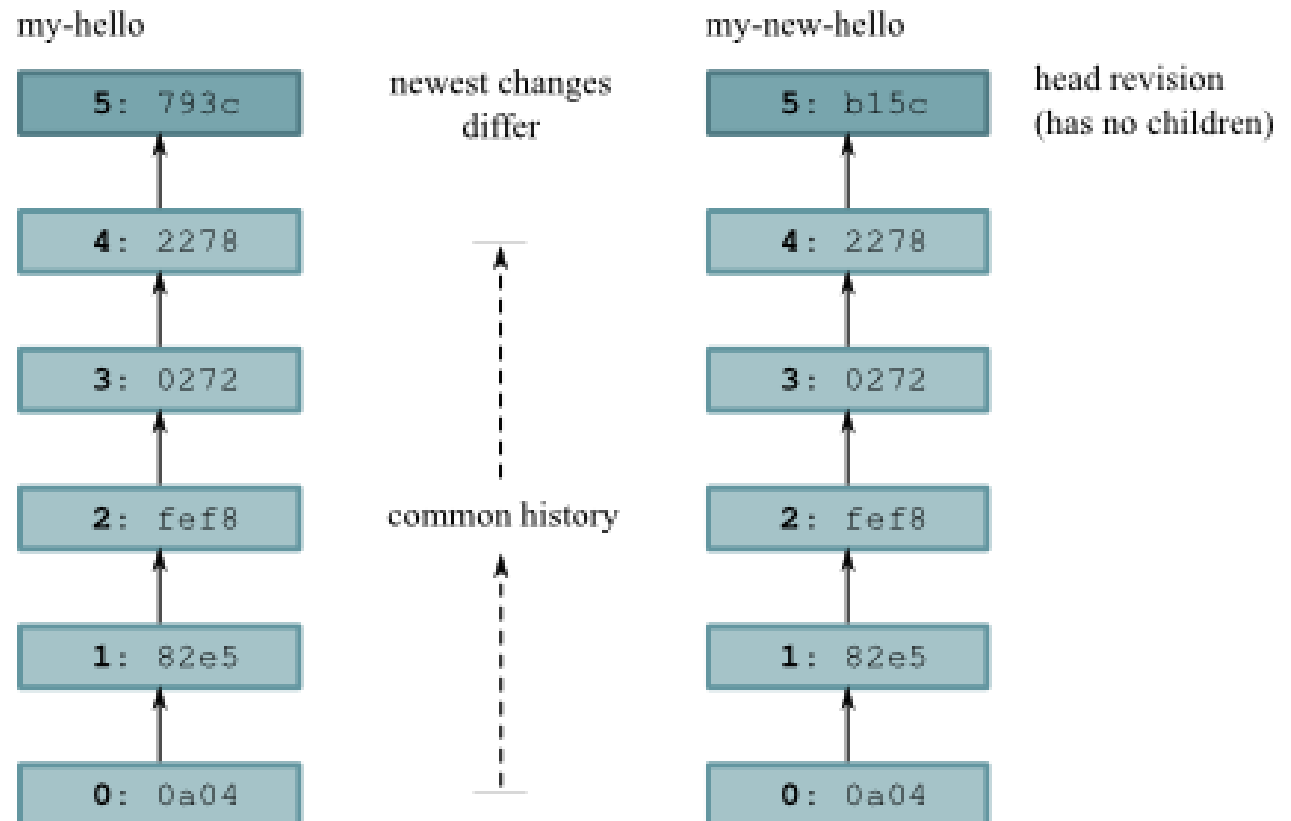


- Merge-Operation erfolgreich, wenn Änderungen in Zweigen verschiedene Bereiche der Datei (oder versch. Dateien) betreffen
- Bei Änderungen derselben Bereiche Konflikt
→ kein automatischer Merge möglich
- Lösen der Konflikte über Tools wie KDiff3
→ KDiff3-Demo

- Klonen (hg clone) erzeugt unabhängige Kopie eines Repositorys
- Bsp.:

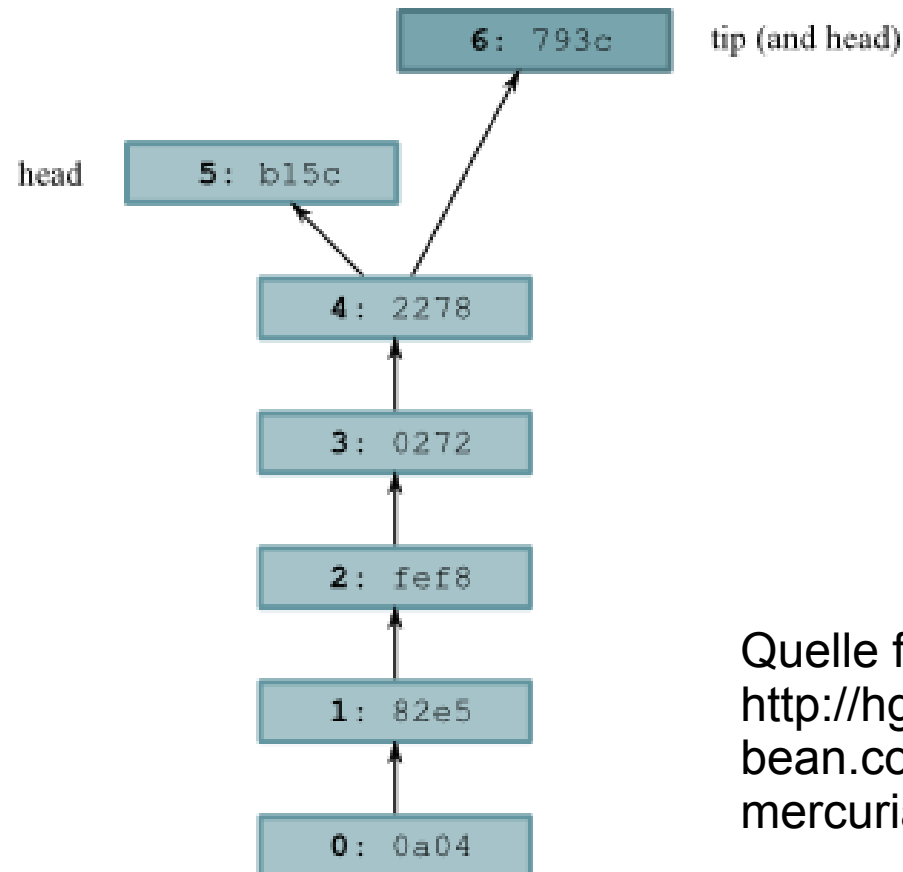
```
$ hg clone \
  my-hello \
  my-new-hello
```

Quelle für Beispiel und Bilder:
<http://hgbook.red-bean.com/read/a-tour-of-mercurial-merging-work.html>



- Pull-Operation (hg pull) zieht aktuelle Fassung aus anderem Repository
- Bsp.:

```
$ hg pull \
  ../my-hello
```



Quelle für Beispiel und Bilder:
<http://hgbook.red-bean.com/read/a-tour-of-mercurial-merging-work.html>

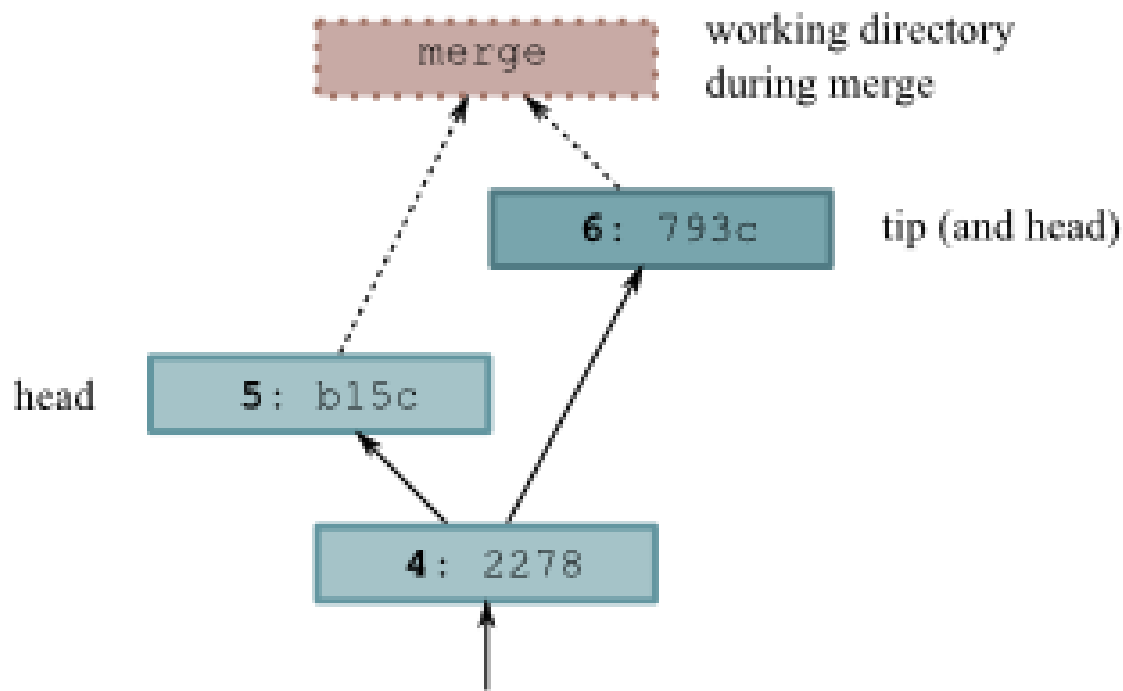
- Zwei Heads (Revisionen ohne Kinder)

```
$ hg heads
changeset: 6:b6fed4f21233
tag:       tip
parent:    4:2278160e78d4
user:      Bryan O'Sullivan <bos@serpentine.com>
date:      Tue May 05 06:55:53 2009 +0000
summary:   Added an extra line of output
changeset: 5:5218ee8aecf3
user:      Bryan O'Sullivan <bos@serpentine.com>
date:      Tue May 05 06:55:55 2009 +0000
summary:   A new hello for a new day.
```

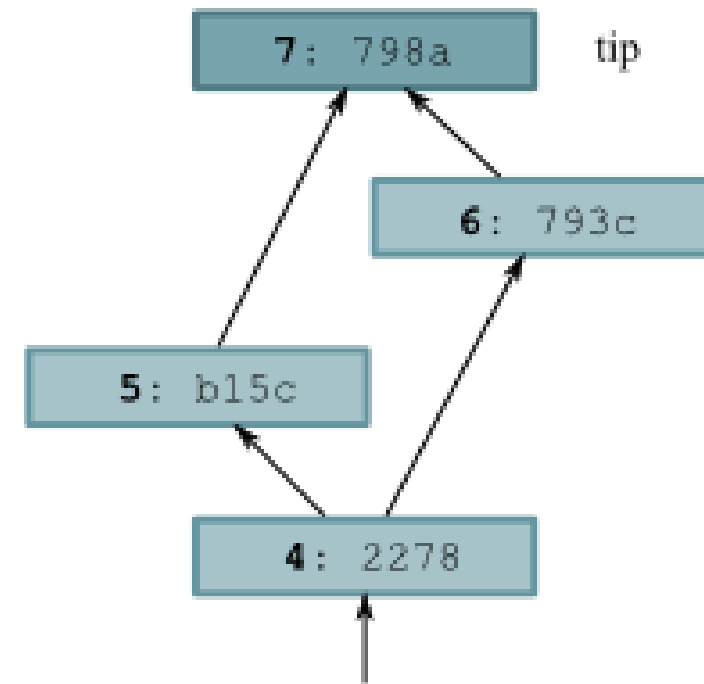
- hg merge vereinigt die beiden heads

- hg merge und hg commit

Working directory during merge



Repository after merge committed



IT-Infrastruktur

SS 2015

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz D:

v1.1, 2015/03/05

Rechnerstrukturen

- Pipelining
- Multi-Core-, Multi-Prozessor-Systeme

Vorlesungsübersicht

Seminar

Wiss. Arbeiten

Datenformate und Wandlung

PC als Arbeitsplatz

Ergonomie und Arbeitsschutz

Rechnerstrukturen

Zentrale / verteilte IT-Infrastrukturen

Folien D

- Prinzipien und Methoden für Analyse, Implementierung, Bewertung und Klassifikation von Rechnerarchitekturen
- Architekturprinzipien und Merkmale moderner RISC- und CISC- (Mikro-) Prozessoren wie
 - Befehlssätze
 - Superskalarität
 - Pipelining
 - Cache-Organisation
- Organisationsprinzipien von Multiprozessor-Systemen und wichtige Architekturmodelle

1. Grundlagen

- Begriffsbestimmung, Abgrenzung
- Historische Entwicklung

2. Instruction Set Architecture (ISA)

- Registerstruktur, Adressierungsarten
- Maschinenbefehlssätze, Spezialbefehle
- Stack-Maschinen

3. Leistungsbewertung und Leistungsmessung

4. Pipelining

- Klassische Fünf-Stufen-Pipeline
- Pipeline-Hemmnisse
- Superskalarität, *out of order execution*
- Spekulative Befehlsausführung, Sprungvorhersage

5. Speichersysteme

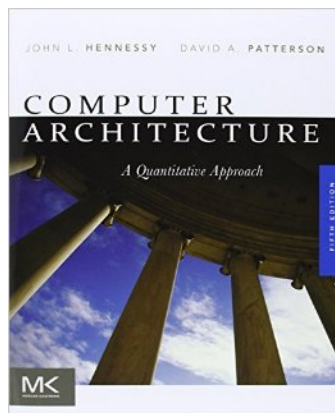
- Speichertypen, Caches

6. Sprungvorhersage

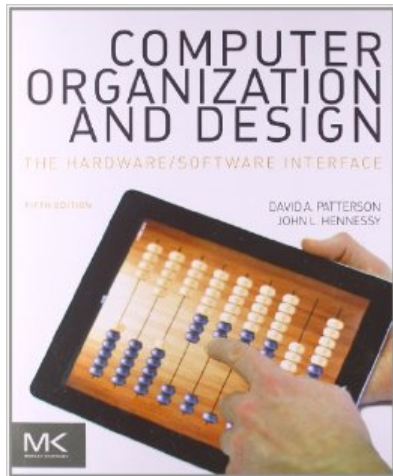
7. Mehrprozessorsysteme



Rechneraufbau und Rechnerarchitektur Axel Böttcher (Hochschule München) ISBN: 3540209794 (2007)



Computer Architecture: A Quantitative Approach 5th ed. John L. Hennessy, David A. Patterson ISBN: 012383872X (2011)



Computer Organization and Design, The Hardware/Software Interface 5th ed.

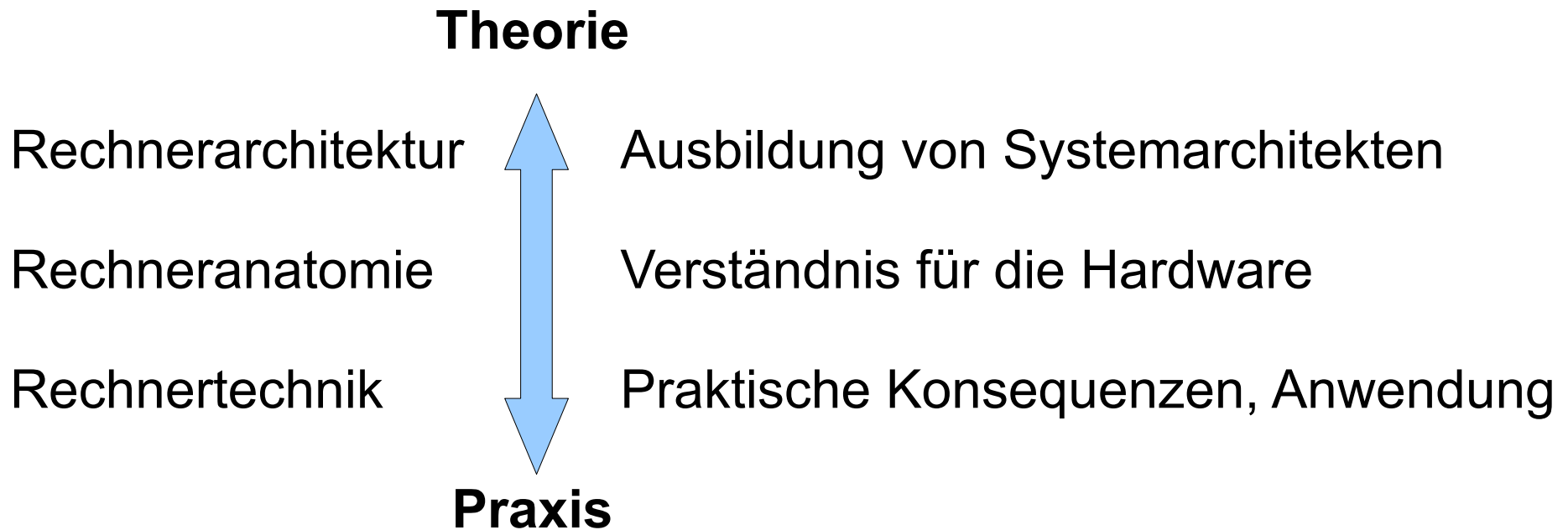
David A. Patterson, John L. Hennessy
ISBN: 0124077269 (2013)



Rechnerarchitektur: Von der digitalen Logik zum Parallelrechner Andrew S. Tanenbaum ISBN: 3868942386 (2014)

- Einen Großteil der Folien habe ich von
Prof. Dr. Axel Böttcher
(FH München) übernommen, der diese
Vorlesung 2009 gehalten hat und mir sein
Material freundlicherweise zur Verfügung
gestellt hat.

1. Grundlagen



Rechnerarchitektur umfasst

- die Analyse
 - den Entwurf
 - die Bewertung
 - die Synthese
- von Rechnern und Rechnerkomponenten.

Dazu:

- strukturelle
- organisatorische
- implementierungstechnische

Aspekte berücksichtigen und auf der

- globalen Systemebene
- Maschinenbefehlssatzebene
- Mikroarchitekturebene

untersuchen

Zwischen den beteiligten Ebenen und den verschiedenen Teilaspekten der Rechnerarchitektur sind Rückkopplungen möglich.

Auf allen Ebenen umfangreiche Wechselwirkungen mit anderen Disziplinen der

- Informatik,
- Ingenieur- und Naturwissenschaften
- Mathematik

Verschiedene Betrachtungsebenen

Globale Systemebene (Prozessoren, Busse, Speicher)

- *Für:* System-Architekt für Chip/Motherboard, Vertrieb
- *Sicht auf:* ganzes System
- *Elemente:* Welche Hauptelemente besitzt das System, und wie sind diese miteinander verbunden.

Maschinenbefehlssatzebene

- *Für:* Compilerbauer, Assembler-Programmierer, Vertrieb
- *Sicht auf:* Prozessor-Funktionen
- *Elemente:* Satz von Befehlen (Maschineninstruktionen), den der Prozessor beherrscht.

Verschiedene Betrachtungsebenen

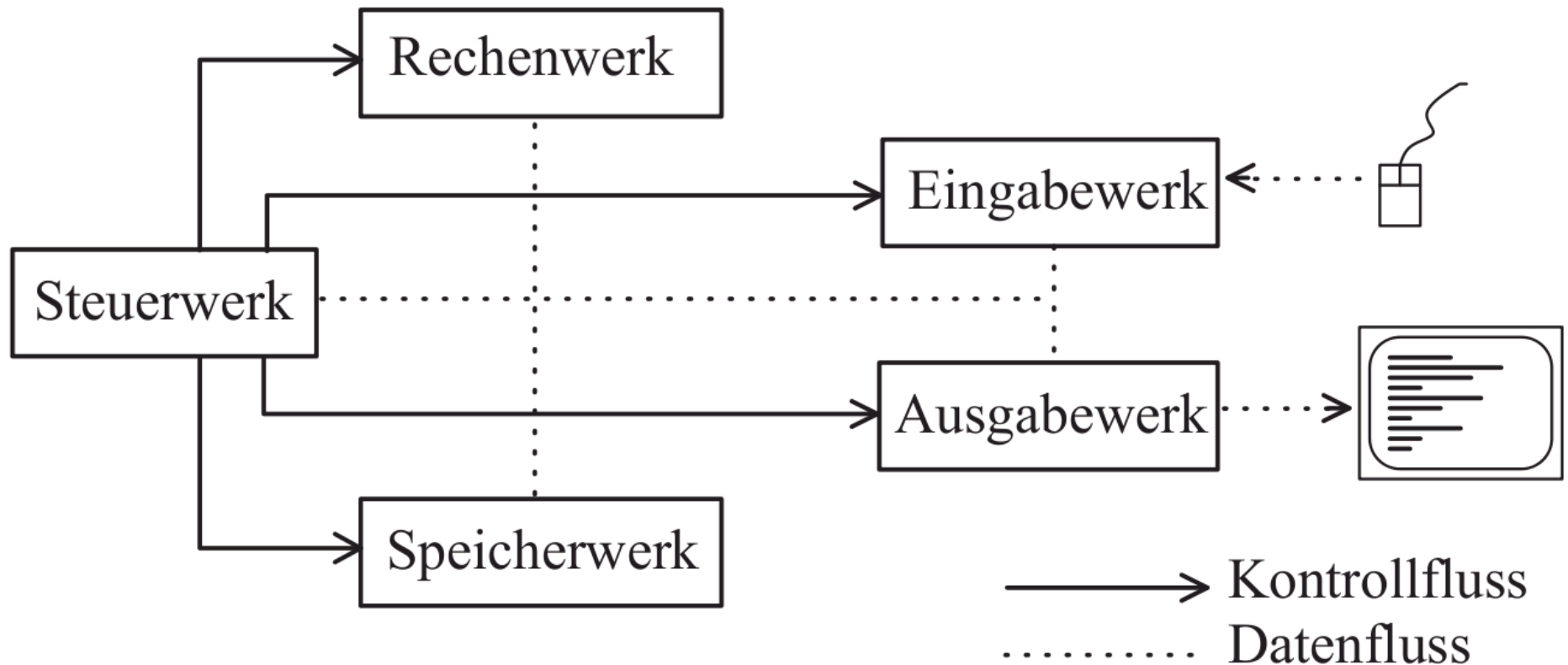
Register-Transfer-Ebene (Mikroprogramm-Ebene)

- *Für:* CPU-Entwickler, Programmierer, Mikroprogrammierer (Informatik-Grundwissen)
- *Sicht auf:* Infrastruktur auf dem Chip
- *Elemente:* Register; wie diese mit ihrer Umgebung verbunden sind und welche Verknüpfungen vorgesehen sind, um den Befehlssatz zu realisieren.

Teile der Mikroarchitekturebene

- Logikebene
 - *Für:* Chip-Designer
 - *Sicht auf:* Schaltung
 - *Elemente:* Gatter und Flipflops und wie diese verbunden sind.
- Hardware-Realisierungs-Ebene
 - *Für:* Designer
 - *Sicht auf:* Chiplayout
 - *Elemente:* Bauteile (z.B. Transistoren, Kondensatoren etc.) und wie diese verbunden sind, um Gatter oder Flipflops zu realisieren.

- Bestandteile eines Universalrechners (Computer) nach John v. Neumann



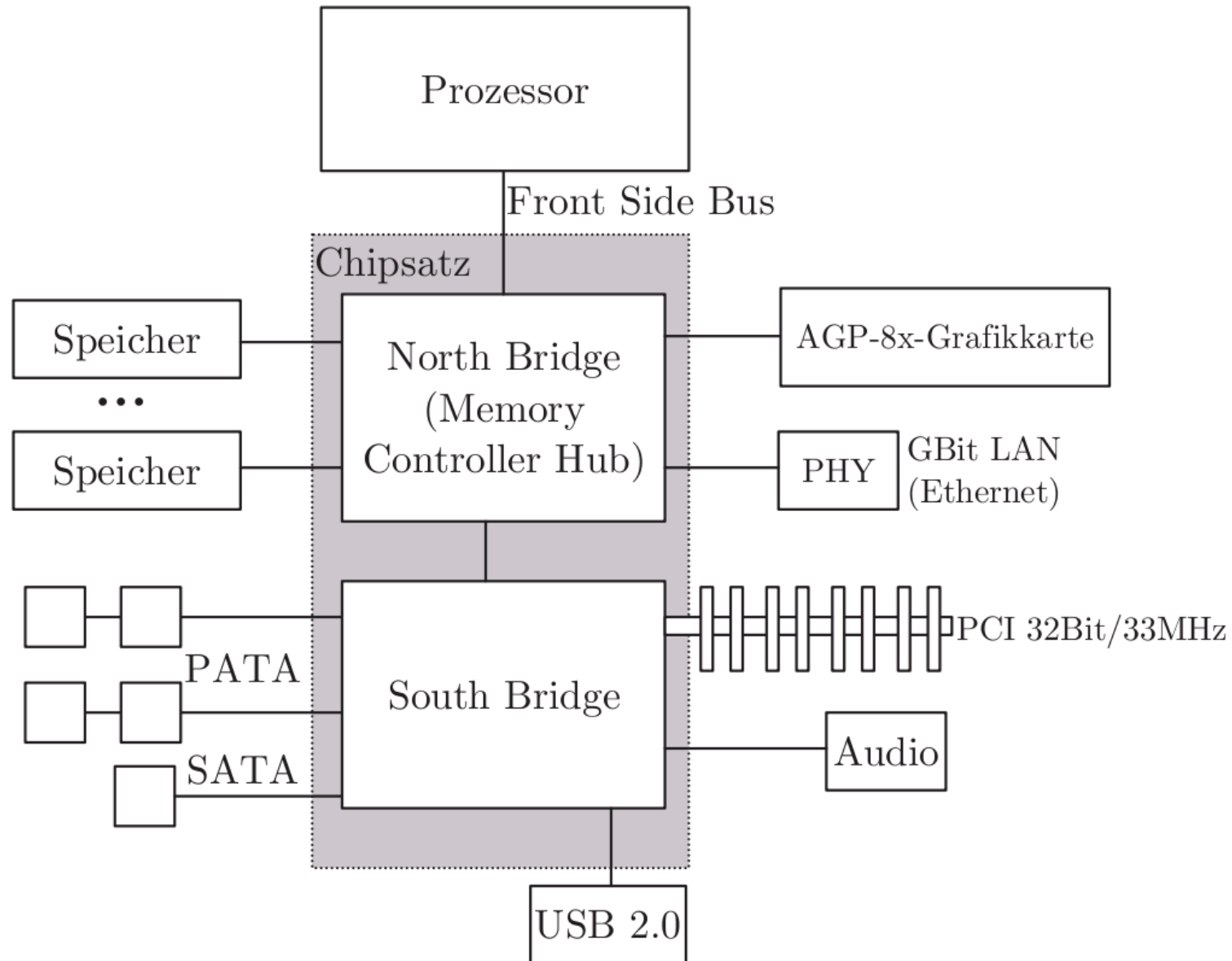
- 5 Funktionseinheiten: Steuerwerk, Rechenwerk, Speicherwerk, Eingabewerk und Ausgabewerk
- Aufbau unabhängig vom zu bearbeitenden Problem
- Programme und Daten, Zwischenergebnisse und Endergebnisse im gleichen Speicher
- Eingabewerk liest Daten und Programm-Code ein und legt sie im Speicherwerk ab.
- Speicher in gleich große Zellen aufgeteilt (fortlaufend nummeriert)
- Aufeinander folgende Befehle befinden sich in aufeinander folgenden Speicherzellen.
- Im Steuerwerk gibt es einen **Befehlszähler**, der immer auf den nächsten auszuführenden Befehl zeigt.

Befehlsausführung

- Der aktuelle Befehl wird aus dem Speicher ausgelesen und im **Befehlsregister** des Steuerwerks zwischengespeichert.
- Der Befehl wird dann dekodiert, und die Ausführung des Befehls durch Steuersignale veranlasst.

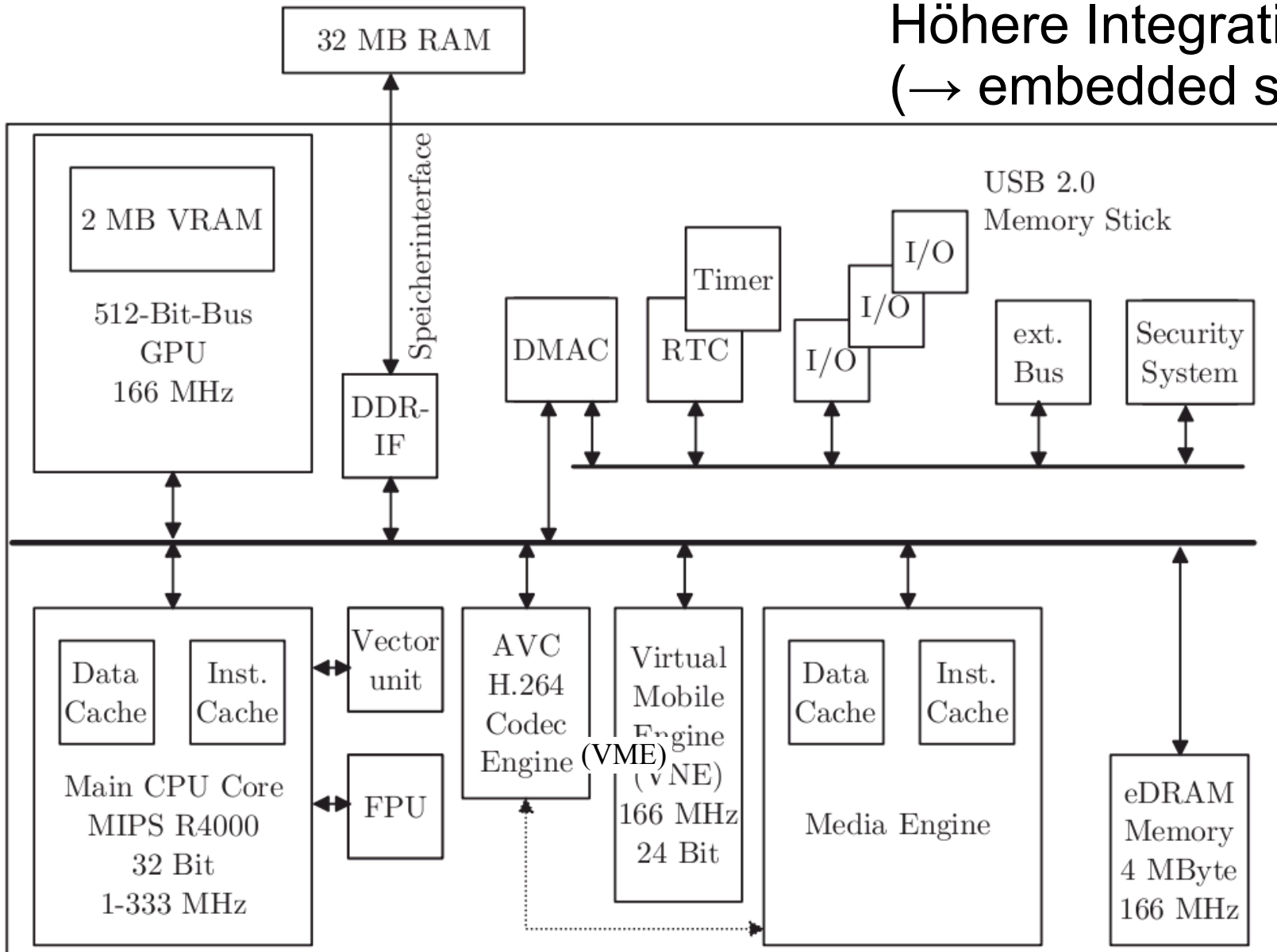
- Es gibt folgende Befehlsarten:
 - Arithmetische und logische Befehle zur Verknüpfung von Daten
 - Transportbefehle zum Verschieben von Daten zwischen diesen Komponenten
 - Bedingte und unbedingte Sprungbefehle, Unterprogrammaufrufe
 - Ein-/Ausgabebefehle zur Kommunikation mit der Peripherie
 - Sonstige Befehle wie Unterbrechen, Warten, Stop etc.

Beispiel: Aufbau eines PCs



Beispiel: Aufbau der PSP

Höhere Integrationsdichte
(→ embedded systems)



Chip		Anzahl Transistoren
Flipflop	Speichert ein Bit	6
Gatter (und/oder)	Verknüpft zwei binäre Werte	4
Addierer	Addiert 64-Bit-breite Worte	> 400
Datenpfad	Komplettes Rechenwerk mit Puffern	> 200.000
Pentium II	Ganzer Prozessor	4,5 Millionen
Pentium 4	Ganzer Prozessor	42 Millionen
Xeon-Dunnington	6-Kern-Prozessor	1,9 Milliarden

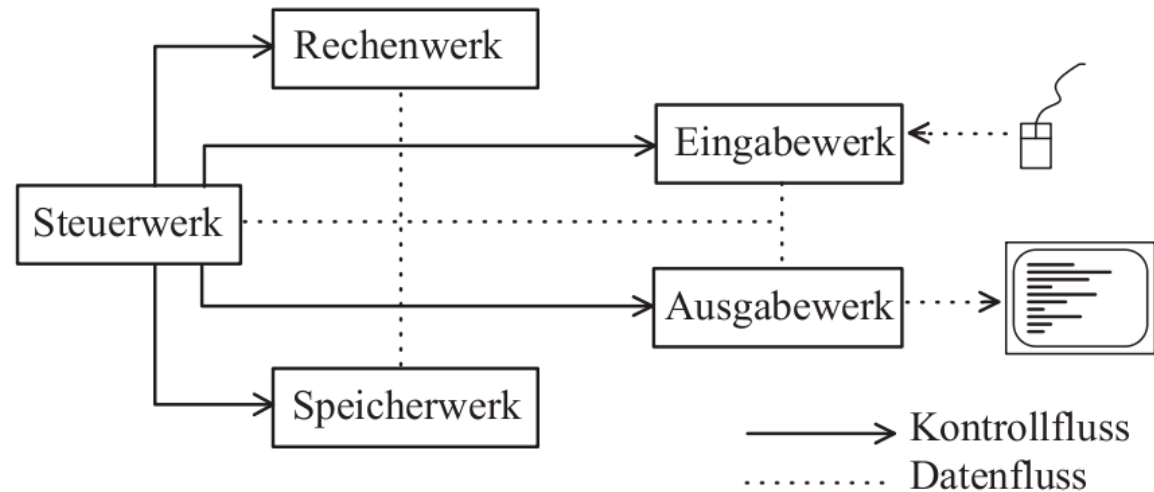
Landschaft der Prozessoren (Wichtige Begriffe)

- **Hardwarebeschreibungssprachen:**
Programmiersprachen zur Beschreibung von Hardware. Können verwendet werden, um Hardware zu generieren. Wichtigste Vertreter: VHDL (Very High Speed Integrated Circuit Hardware Description Language) und Verilog. (siehe HP-Forschungsprogramm PICO: „Program In – Chip Out“)
- **Synthetisierbarer Kern:**
Prozessorkern, der in einer Hardwarebeschreibungssprache vorliegt (**Softcore**) und von Lizenznehmern in eigene Entwürfe eingebunden werden kann; z. B.: MIPS32 24K, ARM1136J, Intel XScale 80200T.

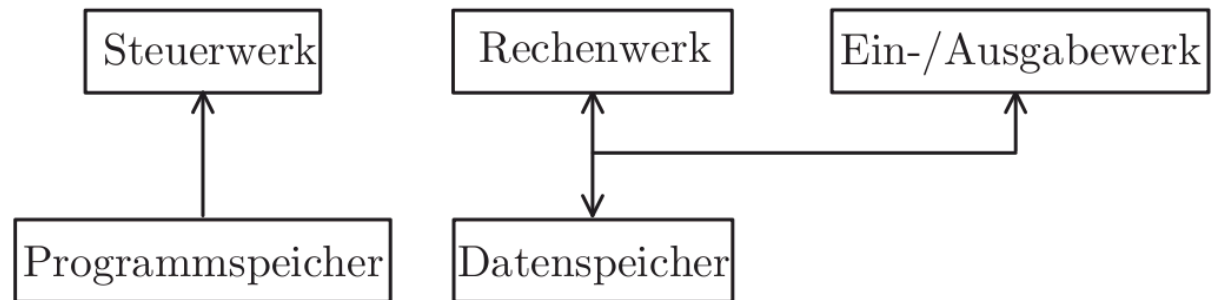
Landschaft der Prozessoren (Wichtige Begriffe)

- **System on a Chip (SoC):**
enthält auf einem einzigen Chip neben dem Prozessorkern auch Speicher, Schnittstellen, Timer und ggf. auch Grafik-, DMA- und Interrupt-Controller sowie PCMCIA, Touch-Panel-Interface etc. (z.B. PowerPC 405LP). Daher kein Chipsatz erforderlich
- **Embedded Systeme** (auch: Prozessrechnersysteme):
kleine Computer, die in bestimmten Produkten eingesetzt werden (also **eingebettet** sind). Sie übernehmen dort Steuerungs-, Kontroll- oder Bedienaufgaben.
Einsatzgebiete: Haushaltsgeräte, Unterhaltungselektronik, Fahrzeuge, ...

- Von-Neumann-Architektur



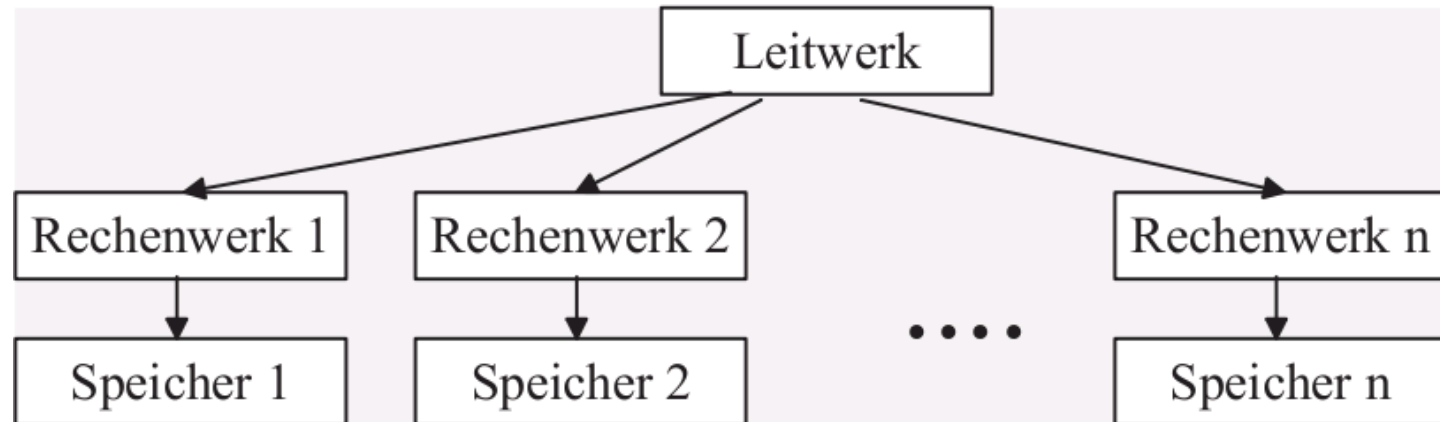
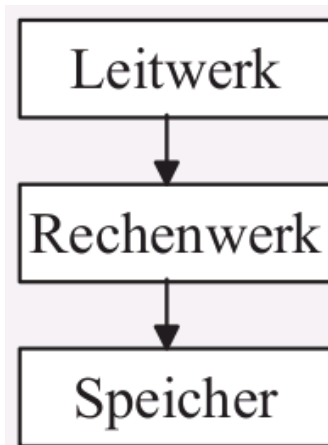
- Harvard-Architektur



Getrennte Speicher für Programm und Daten. Anzutreffen bei manchen Signalprozessoren. Als modifizierte Harvard-Architektur in praktisch allen modernen Prozessoren mit getrennten Level-1-Caches für Code und Daten verwendet

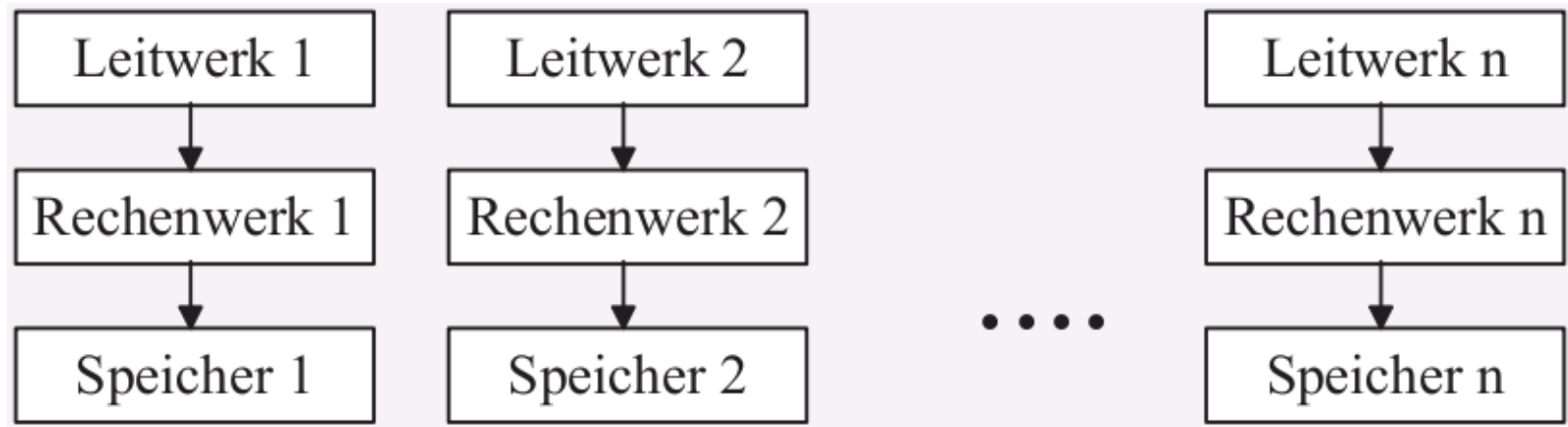
Klassifikation nach Flynn: SISD, SIMD

- Flynn (1966). Heute nicht mehr ganz tragfähig, aber die Begriffe werden noch verwendet.
- Single Instruction, Single Data (**SISD**):
- Single Instruction, Multiple Data (**SIMD**):

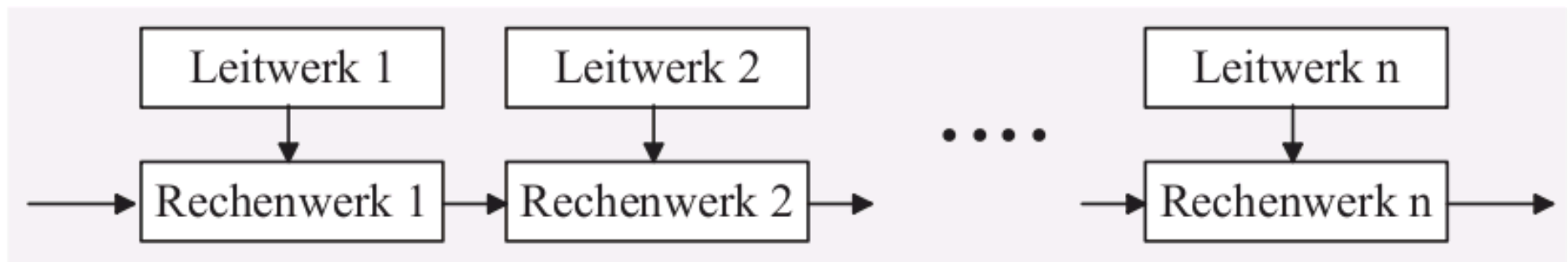


Klassifikation nach Flynn: MIMD, MISD

- Multiple Instruction, Multiple Data (**MIMD**):



- Multiple Instruction, Single Data (**MISD**):



2. ISA: Instruction Set Architecture (Befehlssatzarchitektur)

Befehlssatzarchitektur (Instruction Set Architecture, ISA)

Beschreibung umfasst:

- Maschinenbefehlssatz
- Registerstruktur
- Adressierungsarten
- Interruptbehandlung

Klassisch: Unterscheidung in
Ein-, Zwei- und Drei-Adress*maschinen*

Heute üblicher: unterscheiden nach
Ein-, Zwei- und Drei-Adress*befehlen*

- Register: die schnellsten speichernden Elemente eines Prozessors
- meist allgemein verwendbare Register (General Purpose Registers, GPR) und Spezialregister.
- Gesamtheit aus Befehlssatz und verfügbaren Registern heißt **Programmiermodell**

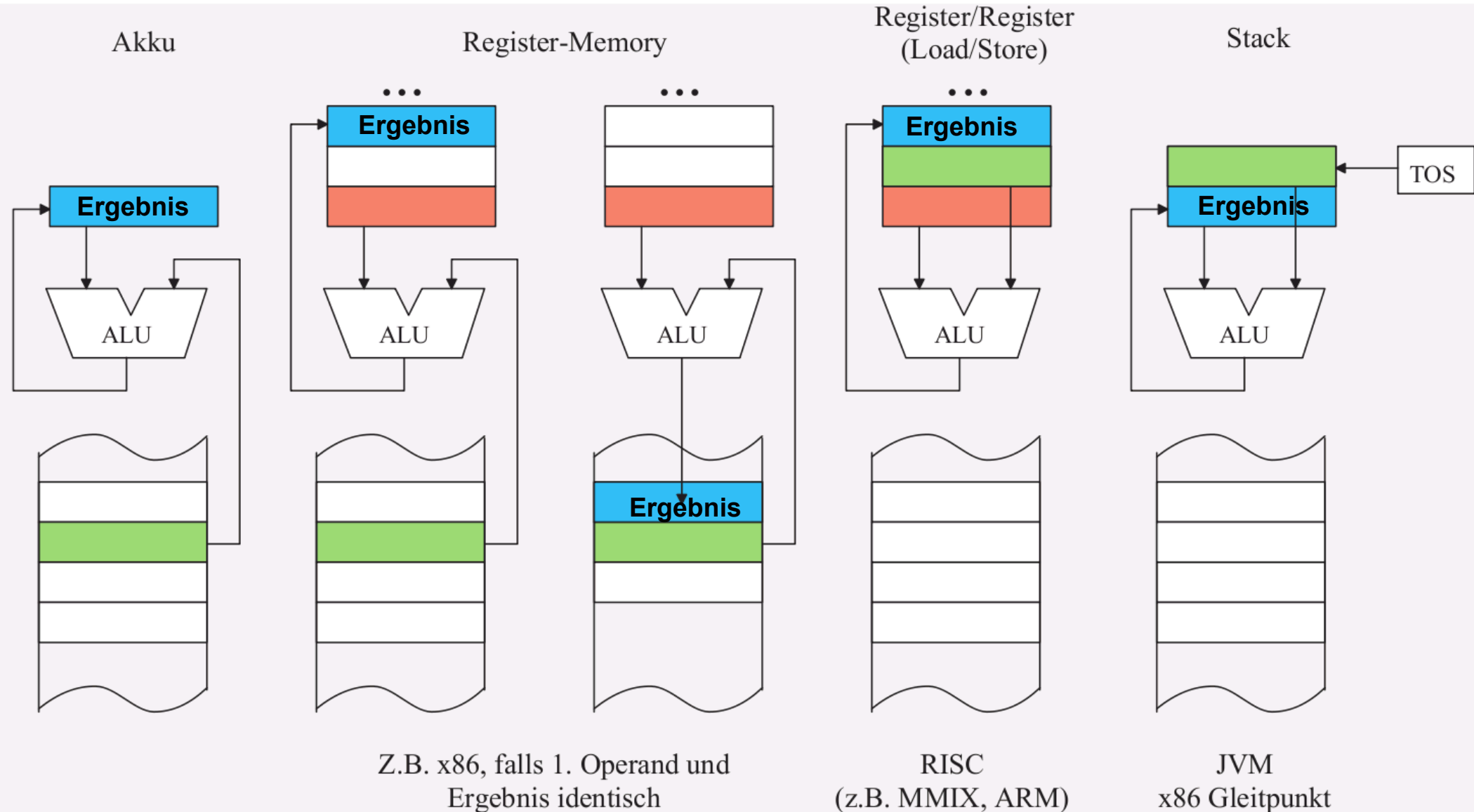
Typische Spezialregister

- Befehlszähler
- Stackpointer
- Statusregister (kann z. B. anzeigen, ob bei der letzten Operation ein Überlauf aufgetreten ist, oder ob das Ergebnis negativ war etc.)
- Indexregister (für Adressrechnungen)

- ISAs unterscheiden nach Zugriff auf Register und Speicherinhalte
- Aufgabe: Werte aus zwei Speicherzellen addieren und in dritter Zelle speichern

$C := A + B;$

Operanden und Ergebnis (2/6)



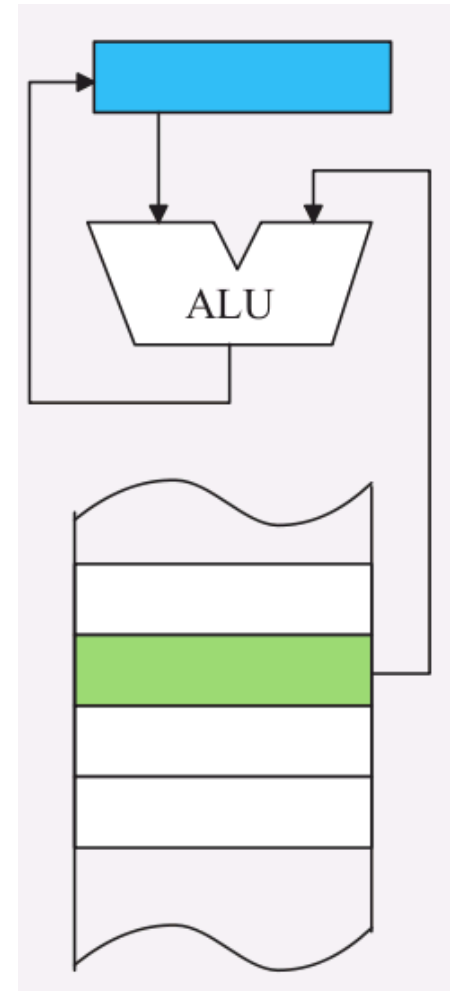
$C := A + B$

mit **Akku**:

```
LOAD  A
ADD   B
STORE C
```

nutzt implizit den Akku:

- LOAD A = lade A in Akku
- ADD B = addiere B zum Wert in Akku (Ergebnis im Akku)
- STORE C = schreibe Akku-Inhalt nach C

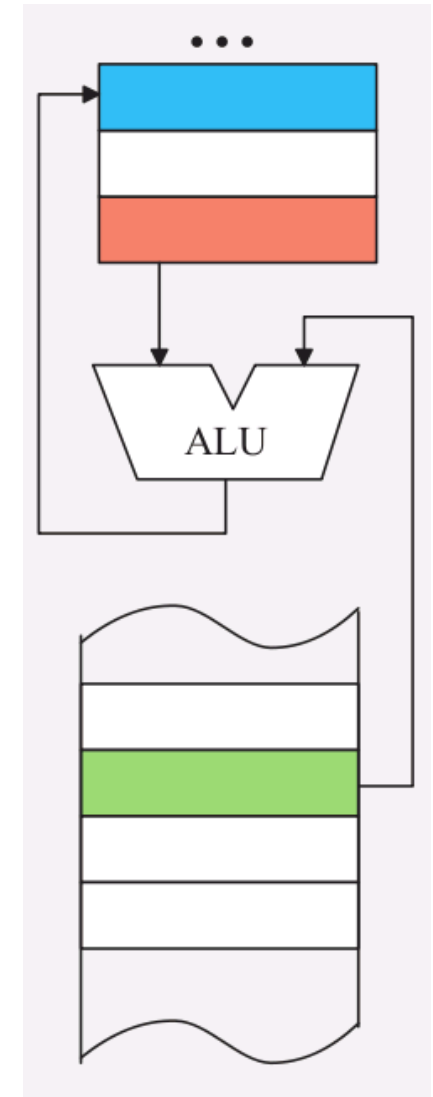


$C := A + B$

mit **Register-Memory**:

```
LOAD  R1, A
ADD   R3, R1, B
STORE R3, C
```

nennt alle Register explizit (R1, R3)
Argumente können Register oder
Speicherstellen sein (z. B. R1, B)

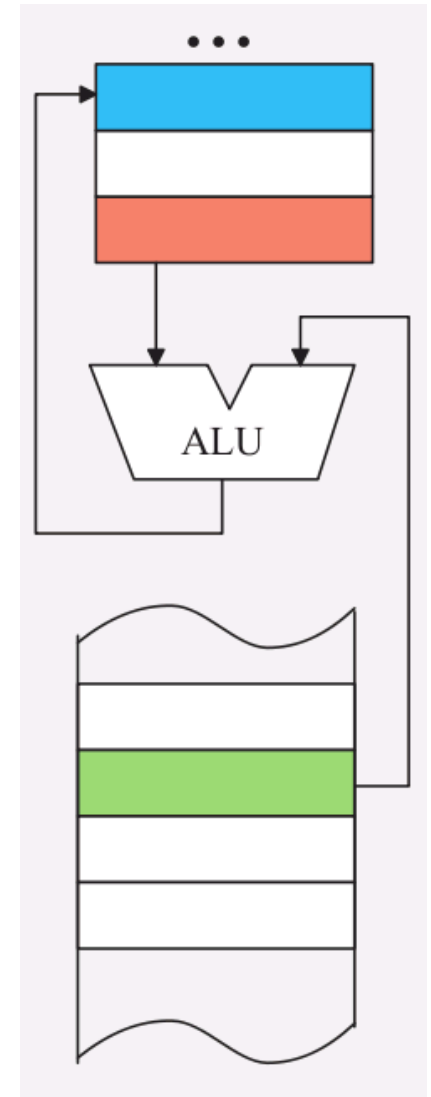


$C := A + B$

mit **Register-Register (Load/Store)**:

```
LOAD  R1, A
LOAD  R2, B
ADD   R3, R1, R2
STORE R3, C
```

- nennt alle Register explizit (R1, R3)
- Argumente für Operationen können nur Register sein (R1, R2)
- Zugriff auf Speicher nur durch LOAD, STORE



$C := A + B$

mit **Stack**:

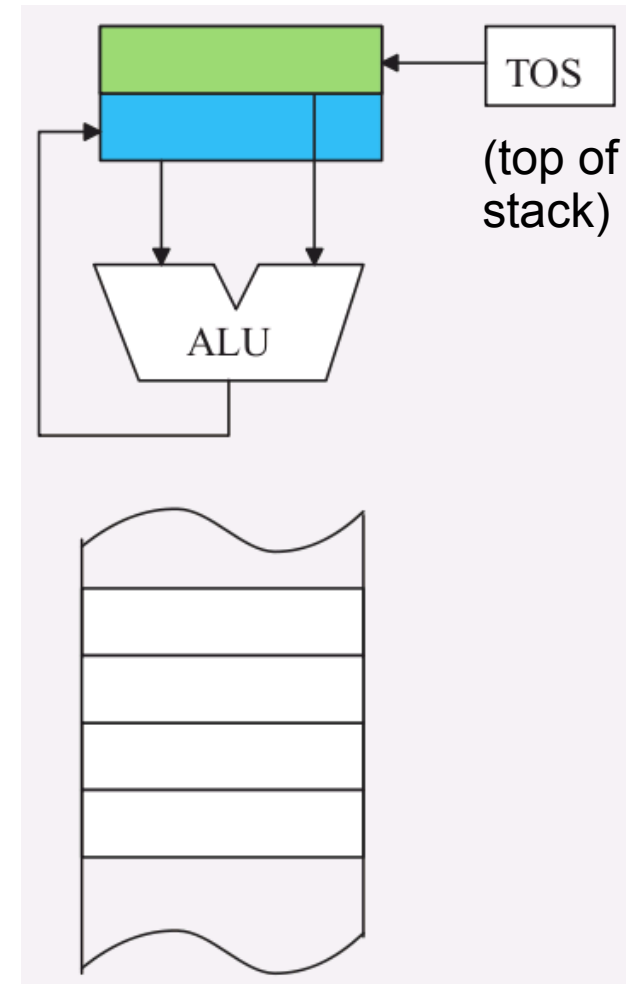
PUSH A

PUSH B

ADD

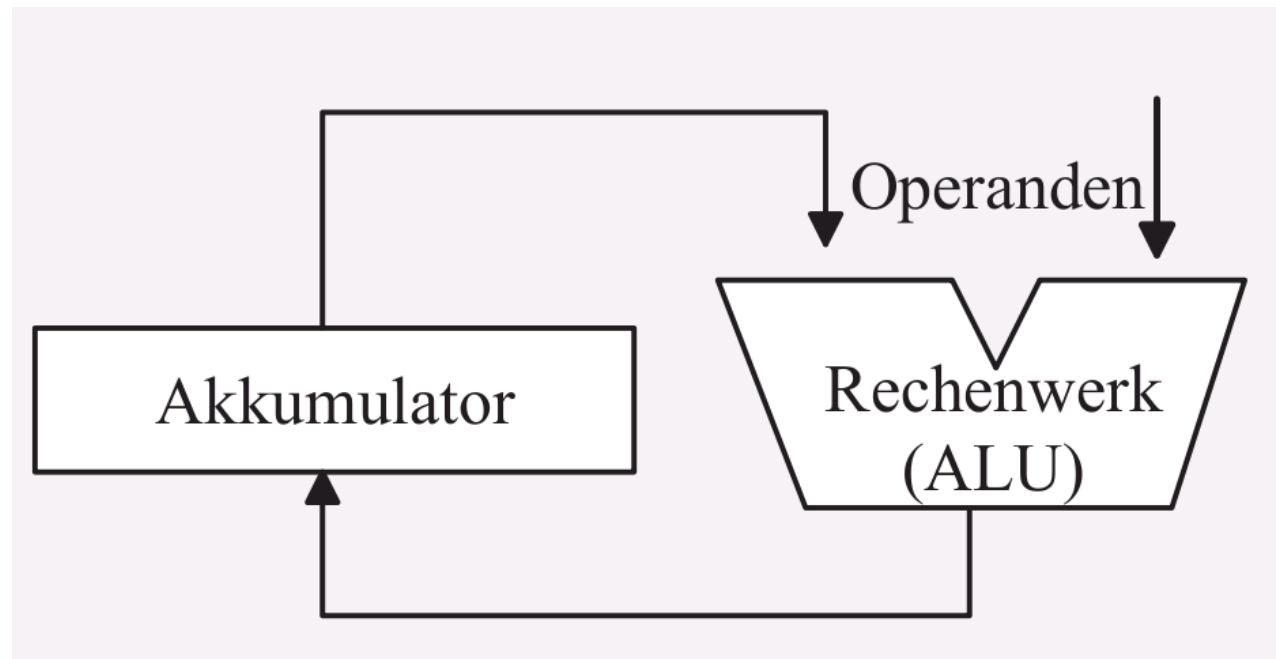
POP C

- keine Register
- Argumente auf Stack, dann Operation
- Operation implizit (oberste Werte auf Stack)
- Zugriff auf Speicher nur durch PUSH, POP



Ein-Adress-Maschinen (1/4)

- Maschinen, deren Befehle nur einen Operanden haben, heißen **Ein-Adress-Maschinen**.
- Akku: spezielles Register, impliziter linker Operand und Zielregister für das Ergebnis („Akkumulatormaschinen“)
- rechter Operand aus Speicher



- **Vorteile**

- Ausführung der einzelnen Befehle wegen der einfachen Hardware sehr schnell
- geringer Speicherbedarf für einen Befehl
- Fast jeder Befehl hat einen Operanden (außer z. B. NOP, INC, DEC) → einheitliche Befehlslänge, einfaches Aktualisieren des Program Counters

- **Nachteile**

- Programmierung in diesem Format erfordert Übung – vor allem das Auswerten von mathematischen Formeln
- Programme wegen des häufig erforderlichen Zwischenspeicherns von Hilfsgrößen etwas „länglich“

- Berechnung der Formel $y = \frac{x_1 + x_2 x_3}{x_1 - x_2}$

mit MMIX (als Ein-Adress-Maschine; \$1=Akku)

01	x1	OCTA	3
02	x2	OCTA	7
03	x3	OCTA	11
04	Accu	IS	\$1
05	Main	LD0	Accu, x2
06		LD0	\$2, x3
07		MUL	Accu, Accu, \$2
08		LD0	\$2, x1
09		ADD	Accu, Accu, \$2

10	ST0	Accu, x3
11	LD0	Accu, x1
12	LD0	\$2, x2
13	SUB	Accu, Accu, \$2
14	ST0	Accu, x1
15	LD0	Accu, x3
16	LD0	\$2, x1
17	DIV	Accu, Accu, \$2
18	TRAP	0, Halt, 0

- Gleiches Programm in „echter“ Akku-Syntax:

01	x1	OCTA	3	
02	x2	OCTA	7	
03	x3	OCTA	11	
04		LOAD	x2	Akku enthält x2
05		MUL	x3	Akku enthält $x2 * x3$
06		ADD	x1	Akku enthält $x2 * x3 + x1$
07		STOR	x3	$x3 \leftarrow x2 * x3 + x1$
08		LOAD	x1	Akku enthält x1
09		SUB	x2	Akku enthält $x1 - x2$
10		STOR	x1	$x1 \leftarrow x1 - x2$
11		LOAD	x3	Akku enthält orig. $x2 * x3 + x1$
12		DIV	x1	Akku enthält Ergebnis
13		TRAP	Halt...	

Zwei-Adress-Befehle (1/3)

- Befehle mit zwei Operanden (Register oder Speicheradressen)

- linker Operand ist implizit Ziel:

CMD x1 , x2 bedeutet $x1 \leftarrow x1 \otimes x2$

- z. B. Addition

ADD \$1 , \$2 bedeutet $\$1 \leftarrow \$1 + \$2$

- in MMIX-Syntax:

ADD \$1 , \$1 , \$2

- Wieder mit MMIX-Befehlen

01	x1	OCTA	3	
02	x2	OCTA	7	
03	x3	OCTA	11	
04	Main	LDO	\$1, x1	Startwerte in \$1,
05		LDO	\$2, x2	\$2,
06		LDO	\$3, x3	\$3.
07		MUL	\$3, \$3, \$2	Produkt $x2 * x3$ in \$3
08		ADD	\$3, \$3, \$1	$x1 + x2 * x3$ in \$3
09		SUB	\$1, \$1, \$2	$x1 - x2$ in \$1
10		DIV	\$3, \$3, \$1	Bruch in \$3, fertig

- und mit „echter“ Zwei-Adress-Syntax

01	x1	OCTA	3
02	x2	OCTA	7
03	x3	OCTA	11
04	Main	LDO	\$1, x1
05		LDO	\$2, x2
06		LDO	\$3, x3
07		MUL	\$3, \$2
08		ADD	\$3, \$1
09		SUB	\$1, \$2
10		DIV	\$3, \$1

01	x1	OCTA	3
02	x2	OCTA	7
03	x3	OCTA	11
04	Main	LDO	\$1, x1
05		LDO	\$2, x2
06		LDO	\$3, x3
07		MUL	\$3, \$3, \$2
08		ADD	\$3, \$3, \$1
09		SUB	\$1, \$1, \$2
10		DIV	\$3, \$3, \$1

- Format von MMIX-Befehlen
- ein Ziel (erster Operand)
- zwei Quellen (2. und 3. Operand)

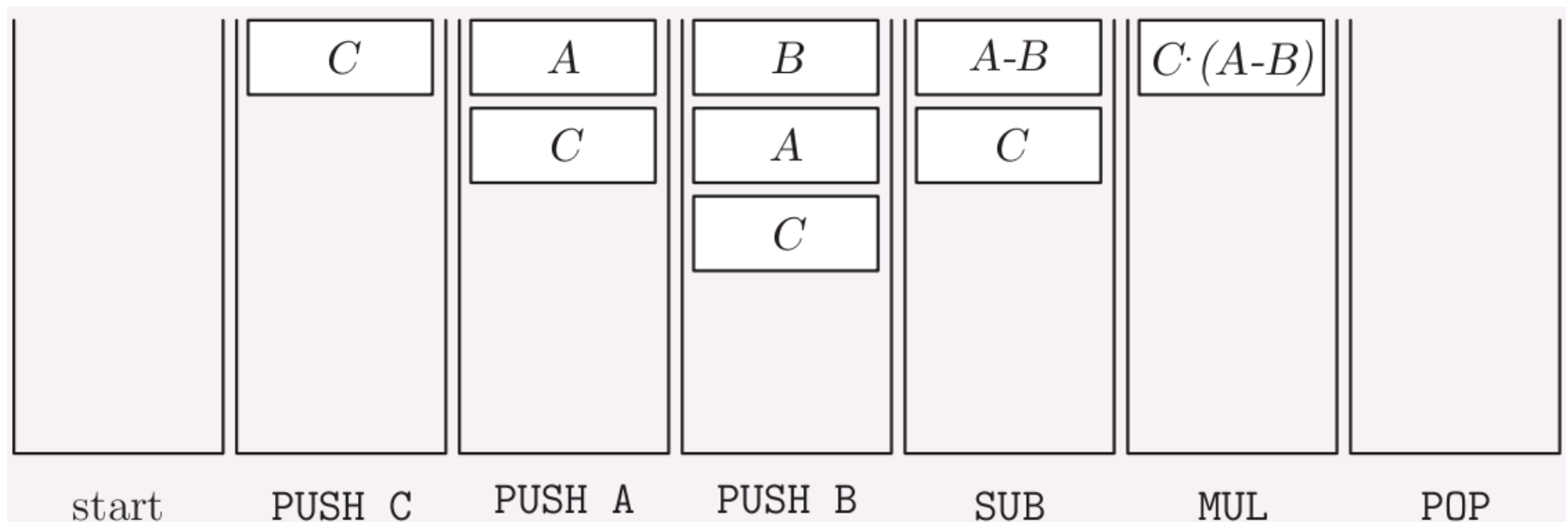
Beispiel

01	x1	OCTA	3
02	x2	OCTA	7
03	x3	OCTA	11
04	Main	LDO	\$1, x1
05		LDO	\$2, x2
06		LDO	\$3, x3
07		SUB	\$6, \$1, \$2
08		MUL	\$7, \$2, \$3
09		ADD	\$7, \$7, \$1
10		DIV	\$7, \$7, \$6

- Vorteile
 - bequeme Programmierung
 - kurze Programme (Anzahl der Befehle)
- Nachteil
 - Eine Speicheradresse ist 64 Bit lang
 - drei Operanden (zunächst Register oder Speicheradresse): enorm große Befehlsbreite
→ darum keine Speicheradressen als Operanden

Null-Adress- (Stack-) Maschine

- Operanden der ALU immer oben auf Stack
- keine normalen Register
- Anwendung: Java Byte Code
- Beispiel: Berechne $(a-b)*c$



- flexibel
 - Befehlslänge hängt vom Befehl ab
 - Auslesen komplizierter (erstes Byte entscheidet über Länge), keine Ausrichtung an Wortgrenzen
 - CISC
- fest
 - Alle Befehle gleich lang
 - Leichtes Auslesen
 - Einschränkung: Operanden nur in Registern (nicht genug Platz für Adressangaben)
 - RISC

$c := a+b; d := a-b; e := c*d$ – gesucht: nur $e = (a+b)(a-b)$

Zwei Varianten

- Register-Memory

ADD c , a , b	SUB d , a , b	MUL e , c , d
1 4 4 4	1 4 4 4	1 4 4 4

S=39

- Register-Register (Load/Store)

LD R1 , a	LD R2 , b	ADD R3 , R1 , R2	SUB R4 , R1 , R2
1 4	1 4	2	2
MUL R5 , R3 , R4	STO R5 , e		
2	1 4		

S=21

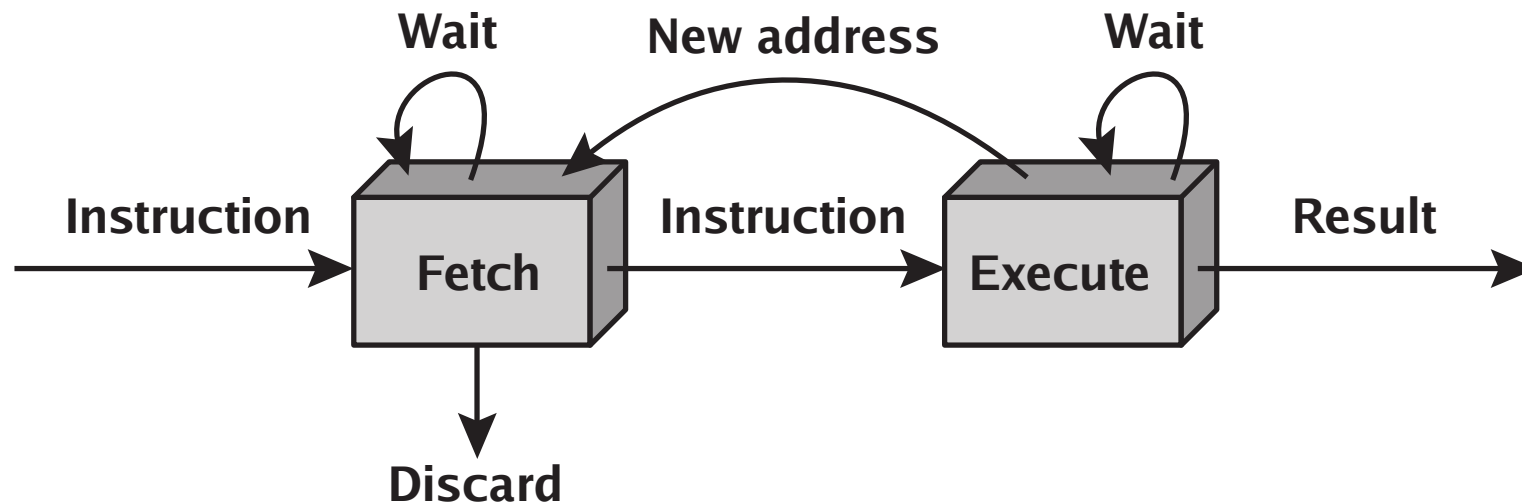
Pipelining

- Wesentliche Begriffe und Fakten über Pipelining kennen
- Vorteile des Pipelining kennen
- Bearbeitung von Sprungbefehlen – Arbeitsschritte für alle Befehle
 - Pipelining Hemmnisse verstehen: Sprünge, Speicherzugriff, Langläufer, Datenabhängigkeiten
- Zusammenhang Arbeitsschritte – Taktrate:
 - Implementierung analysieren können
 - MMIX-Programmstück analysieren und die Ausführung auf einer 5-stufigen Pipeline nachvollziehen können

Einleitung – einfache Pipeline



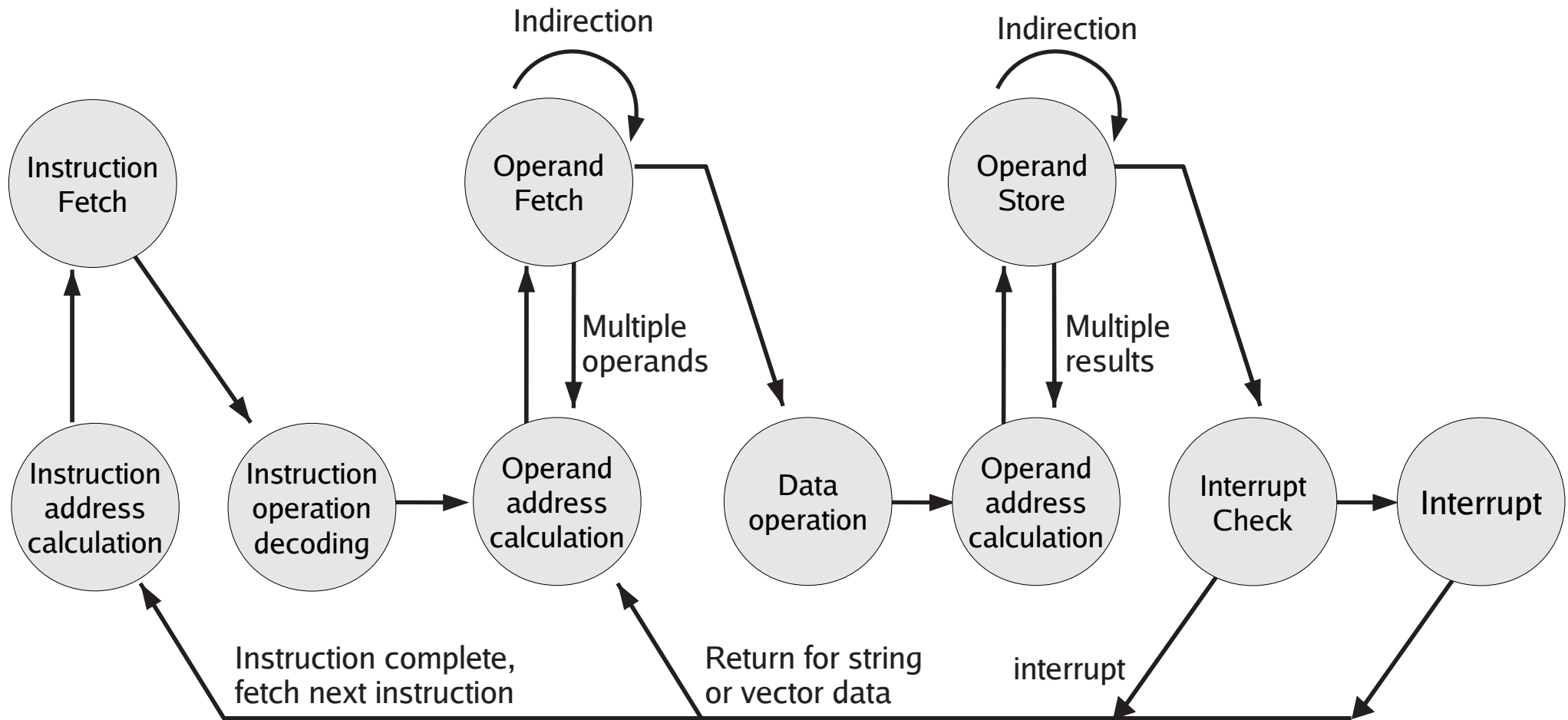
(a) Einfache Ansicht



(b) Erweiterte Ansicht

Quelle: Stallings, Computer Organization and Architecture, 8th ed., 2010, S. 463

Instruction Cycle State Diagram



Quelle: Stallings, Computer Organization and Architecture, 8th ed., 2010, S. 460

- 5-stufige RISC-Pipeline
 - Fetch – Decode – Execute – Memory Access – Write Back
 - siehe auch
http://en.wikipedia.org/wiki/Classic_RISC_pipeline
- 6-stufige CISC-Pipeline
 - Fetch Instruction – Decode – Calculate Operands – Fetch Operands – Execute – Write Back
 - siehe Stallings, S. 464

Klassische 5-Stufen-RISC-Pipeline

(1/3)

- **Holen des Befehls – Fetch (F)**
Befehlszähler liefert Adresse für Speicherzugriff und wird (um vier) erhöht.
- **Dekodieren des Befehls – Decode (D)**
Bei RISC üblicherweise „fixed field decoding“
 - 2 Register-Operanden (OP \$X,\$Y,\$Z):
Bereitstellen der beiden Operanden aus Registern
 - 1 Register-Operand, 1 Direktoperand (OP \$X,\$Y,Z):
Register \$Y auslesen; Z aus Befehlswort
 - Speicherzugriff: Adresse aus Register- und/oder Direktoperand auslesen.
Bei schreibendem Zugriff: Wert aus \$X lesen
 - Sprung: Für Bedingung \$X auslesen; Direktoperand aus Befehlswort (für Sprungzielberechnung)

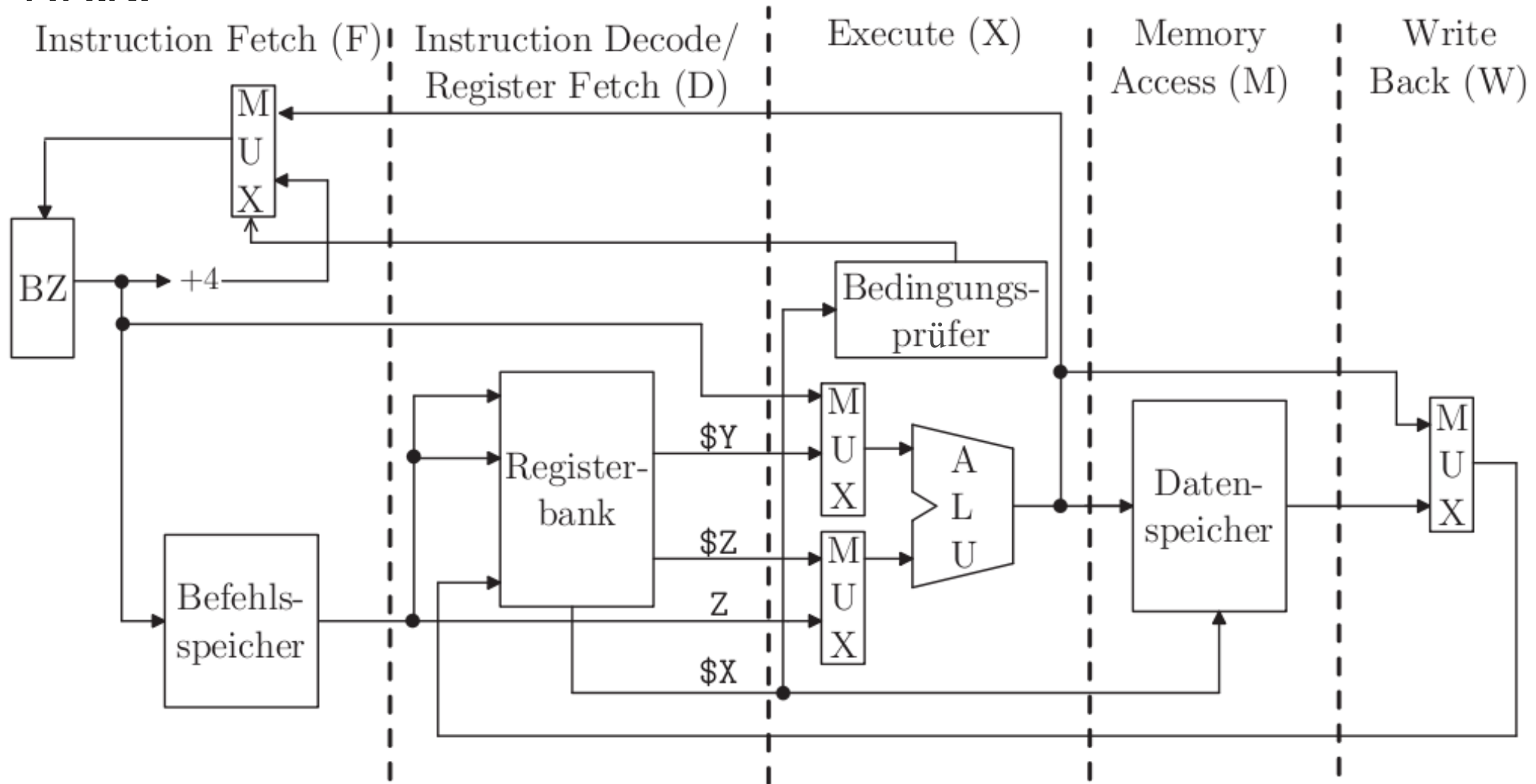
Klassische 5-Stufen-RISC-Pipeline

(2/3)

- Ausführen des Befehls – Execute (**X**)
 - Im Befehl spezifizierte ALU-Operation ausführen
 - Bei Speicherzugriff: (ggf.) Adresse berechnen
 - Bei Verzweigungen: Bedingungen prüfen, Adresse des Folgebefehls festlegen
- Speicherzugriff – Memory Access (**M**)
Lesender oder schreibender Speicherzugriff, falls erforderlich
- Ergebnis zurückschreiben – Result Write-back (**W**)
Ergebnis einer Operation in ein Register schreiben, falls erforderlich (ALU-Output oder Speicherwort)

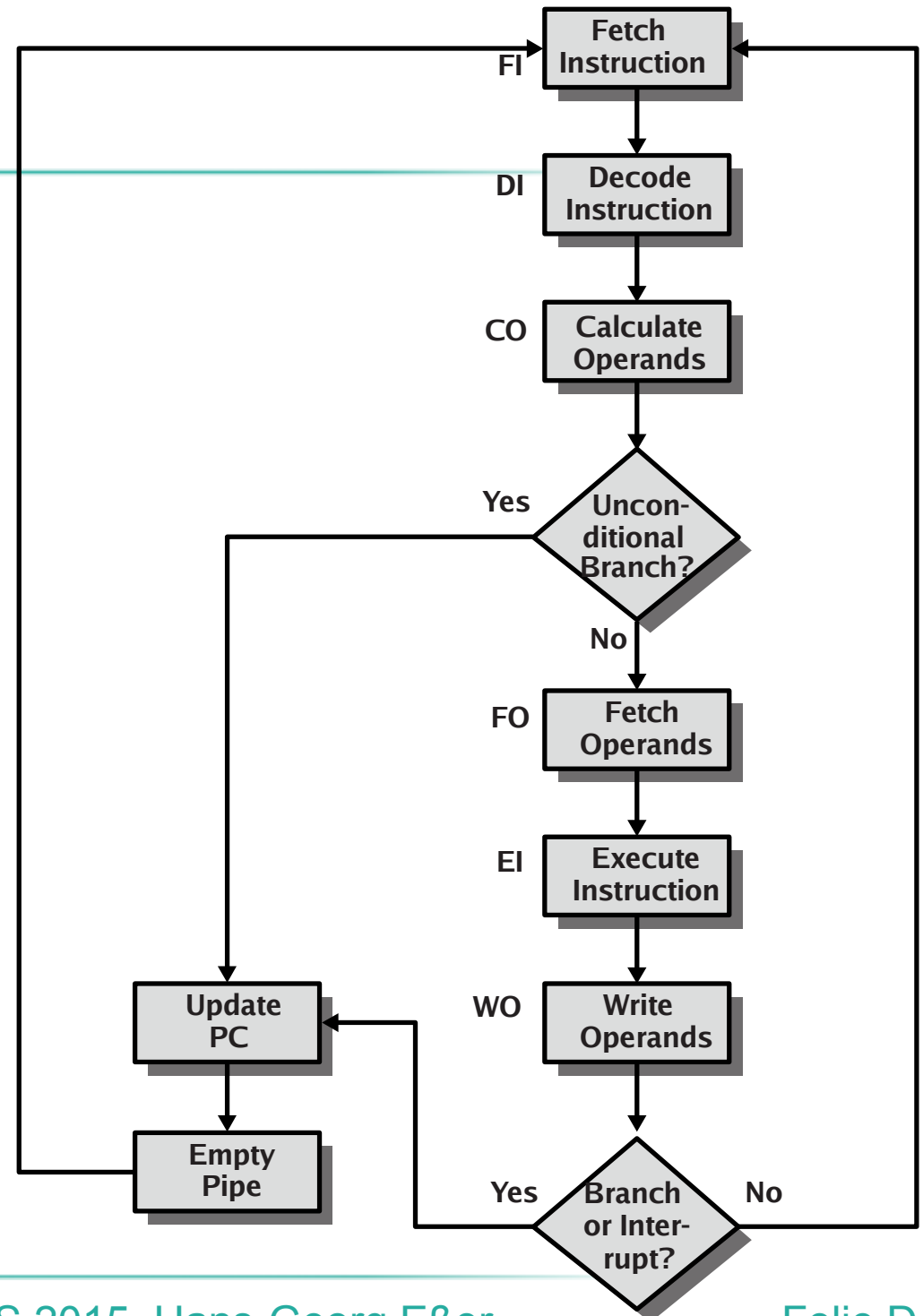
Klassische 5-Stufen-RISC-Pipeline (3/3)

MMIX:



MUX: Multiplexer

6-Stufen- CISC-Pipeline

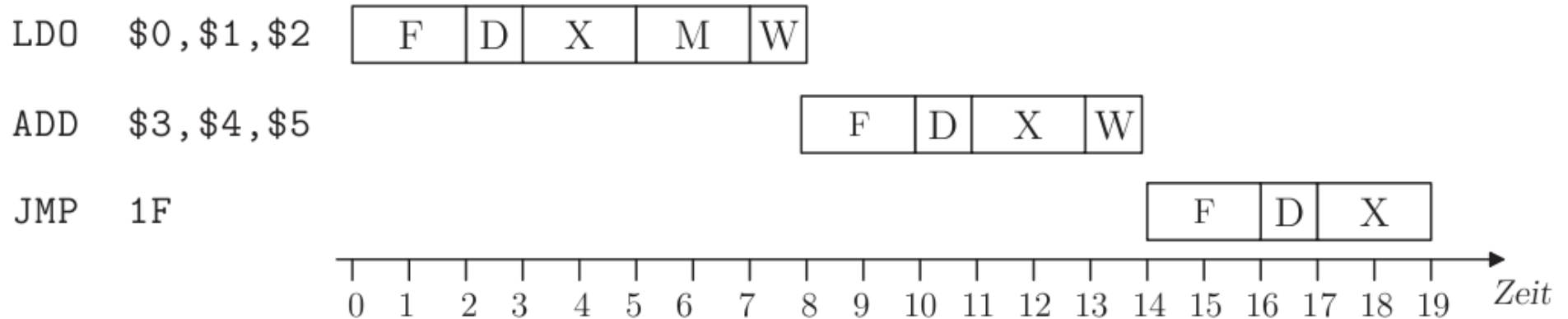


Quelle: Stallings, Computer Organization and Architecture

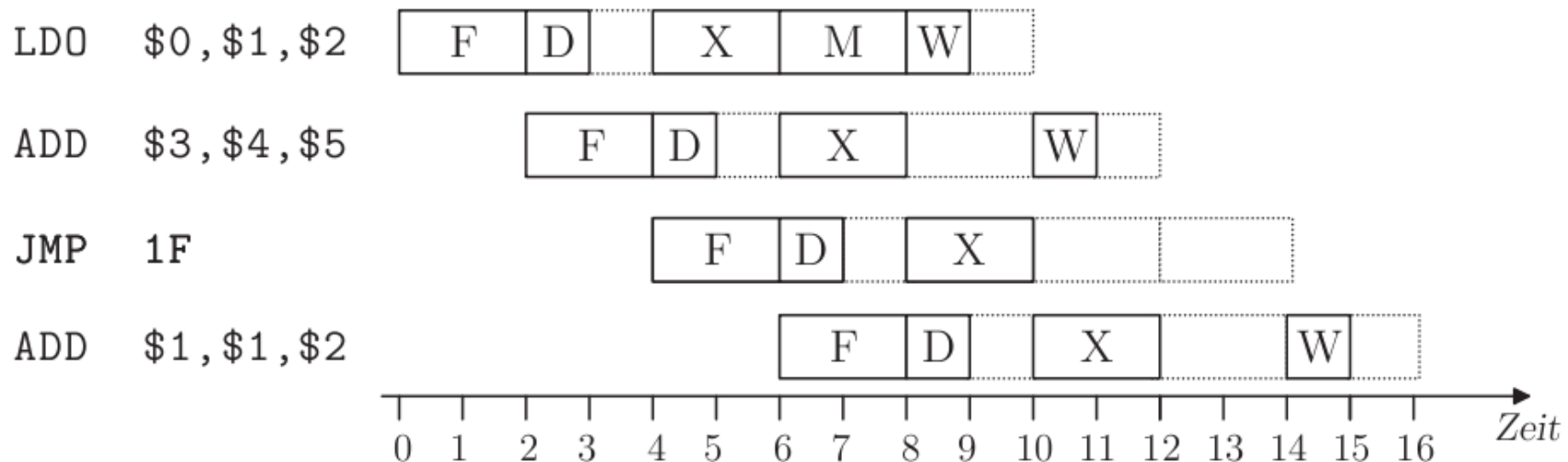
- Übergang zwischen Pipeline-Stufen muss getaktet sein
- Bei Übergang Werte-Weitergabe (CPU-interne Pipeline-Register)
- Annahmen:
 - Fetch, Execute, Memory Access: je $2 t$
 - Decode, Write Back: je t
 - Dauer einer Takteinheit hängt von längster Stufe ab, hier also $2 t$

Ausführung mit und ohne Pipelining

a) Ohne Pipelining



b) Mit Pipelining



Pipeline-Diagramme („Reservation Tables“)

a) aus Sicht der Stufen

	Takt 1	Takt 2	Takt 3	Takt 4	Takt 5	Takt 6	Takt 7
Fetch	Befehl 1	Befehl 2	Befehl 3	Befehl 4	Befehl 5	Befehl 6	Befehl 7
Decode		Befehl 1	Befehl 2	Befehl 3	Befehl 4	Befehl 5	Befehl 6
Execute			Befehl 1	Befehl 2	Befehl 3	Befehl 4	Befehl 5
Memory				Befehl 1	Befehl 2	Befehl 3	Befehl 4
Write Back					Befehl 1	Befehl 2	Befehl 3

b) aus Sicht der Befehle

	Takt 1	Takt 2	Takt 3	Takt 4	Takt 5	Takt 6	Takt 7
Befehl 1	F	D	X	M	W		
Befehl 2		F	D	X	M	W	
Befehl 3			F	D	X	M	W
Befehl 4				F	D	X	M
Befehl 5					F	D	X

- Theoretisch: In jedem Takt ein Befehl fertig
- Praktisch: „Hemmnisse“ („Hazards“) verhindern das:
 - Strukturelle Hemmnisse
 - Datenabhängigkeiten (data hazard)
 - Ablaufbedingte Hemmnisse

Pipeline-Hemmnisse: strukturell (1)

- Nur ein Speicherzugriff pro Takt möglich

	Takt 1	Takt 2	Takt 3	Takt 4	Takt 5	Takt 6	Takt 7
Befehl 1	F	D	X	M	W		
Befehl 2		F	D	X	M	W	
Befehl 3			F	D	X	M	W
Befehl 4				---	---	---	F
Befehl 5					---	---	---

- komplexe Befehle (z. B. Gleitkomma)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
SETH \$3,#4000	F	D	X	M	W									
FMUL \$4,\$4,\$3		F	D	X	X	X	X	M	W					
FADD \$4,\$4,\$2			F	D	--	--	--	X	X	X	X	M	W	
SET \$3, \$10				F	--	--	--	D	--	--	--	X	M	W

Pipeline-Hemmnisse: strukturell (2)

- Beispiele für Länge der Execute-Phase komplexer Befehle (bei MMIX):
 - Integer-Multiplikation: 10 Takte
 - Integer-Division: 60 Takte
 - Gleitkommabefehle: 4 Takte (i.d.R.)
- Taktrate auf Dauer des längsten Befehls anheben?

Pipeline-Hemmnisse: strukturell (3)

- Lösung für Speicherzugriff: Prefetching
 - In jeder Fetch-Phase mehrere Befehle aus dem RAM lesen und puffern („Fetch Buffer“)
 - z. B. MMIX: Befehlslänge 4 Bytes, eine Leseoperation liefert aber 8 Bytes
 - Ungenutzte Memory-Phasen (Befehle ohne Speicherzugriff) für weitere Fetch-Operationen verwenden
- getrennte Prozessor-Caches für Befehle und Daten (→ Pseudo-Harvard-Architektur)

Pipeline-Hemmnisse: Datenabhängigkeiten (1)

- Data hazards: Befehl benötigt ein Ergebnis, das noch nicht fertig berechnet wurde
- Beispiel: Tausch von zwei Werten (aus Quicksort)

	1	2	3	4	5	6	7	
XOR l,l,r	F	D	X	M	W			
XOR r,l,r		F	D	--	--	X	M	(l nicht bekannt)
XOR l,l,r			F	--	--	D	--	(r nicht bekannt)
CMP tmp,l,pivot				--	--	F	--	

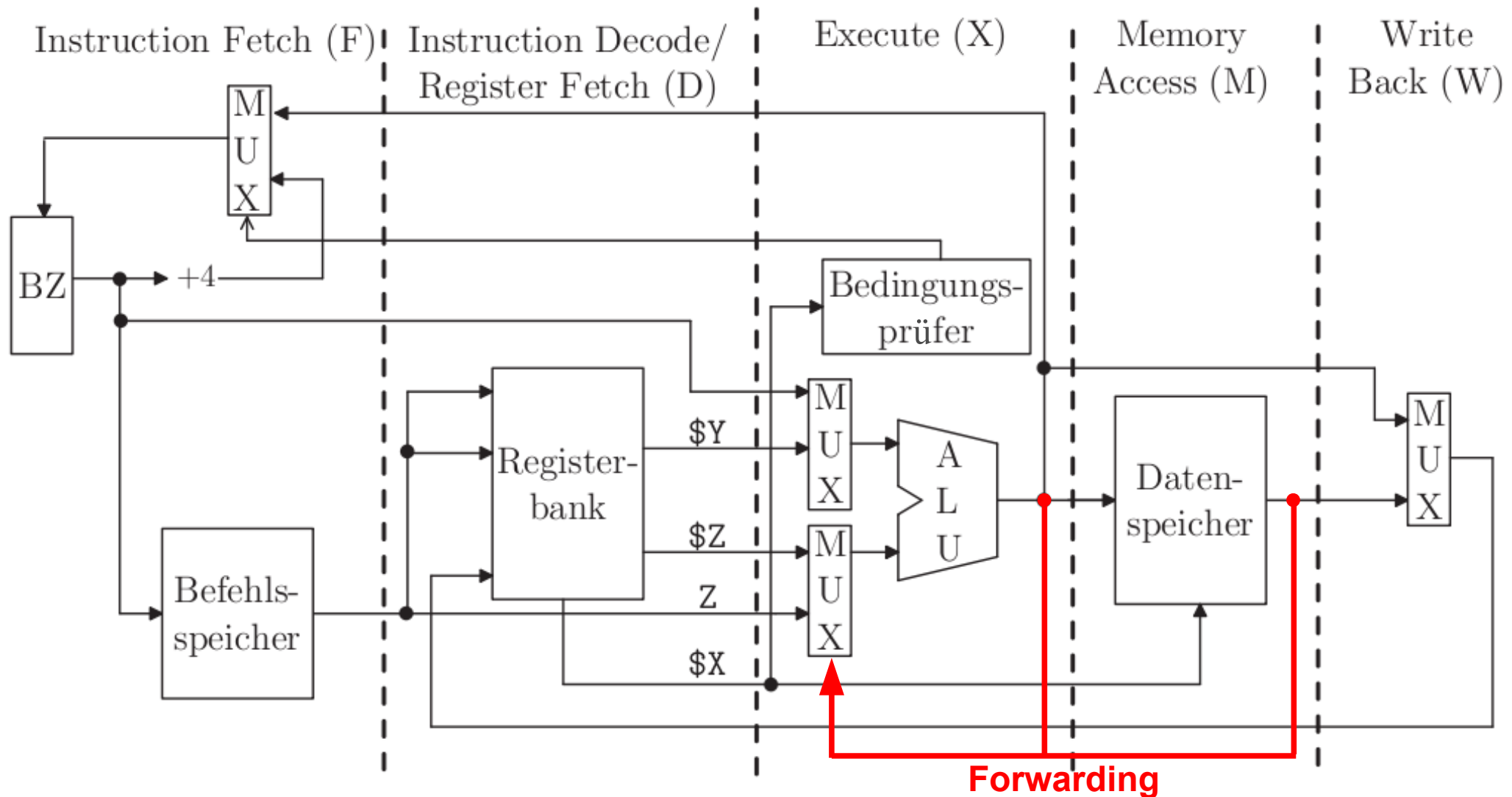
Read-after-Write- (RAW-) Konflikt (zweiter Befehl braucht das im ersten berechnete l)

- Lösung: Result Forwarding / Bypassing

Result Forwarding / Bypassing

- Ergebnisse von X- und M-Phasen immer als ALU-Input zurückreichen („feed back“)
- Spezielle Steuerlogik: Falls erforderlich, die zurückgereichten Ergebnisse (anstelle der Registerinhalte) verwenden

Pipeline-Hemmnisse: Datenabhängigkeiten (3)



Pipeline-Hemmnisse: Datenabhängigkeiten (4)

- ohne Forwarding

	1	2	3	4	5	6	7
XOR l,l,r	F	D	X	M	W		
XOR r,l,r		F	D	--	--	X	M
XOR l,l,r			F	--	--	D	--
CMP tmp,l,pivot				--	--	F	--

(l nicht bekannt)
(r nicht bekannt)

- mit Forwarding

	1	2	3	4	5	6	7
XOR l,l,r	F	D	X	M	W		
XOR r,l,r		F	D	X	M	W	
XOR l,l,r			F	D	X	M	W
CMP tmp,l,pivot				F	D	X	M

Pipeline-Hemmnisse: Datenabhängigkeiten (5)

- Result Forwarding hilft nicht immer
 - Lade-Befehle: Erst nach M-Phase steht Ergebnis zur Verfügung

	1	2	3	4	5	6	7
LD0 \$1,base,off	F	D	X	M	W		
ADD \$1,\$1,\$2		F	D	--	X	M	W
SUB \$3,\$4,\$5			F	--	D	X	M

- bei bedingtem Sprungbefehl:
unklar, an welcher Stelle es weiter geht
- Sprungziel steht erst nach Execution-Phase
des Sprungbefehls fest
- Frage: Was in die Pipeline schreiben?
 - einfach: Immer davon ausgehen, dass *nicht*
gesprungen wird
 - MMIX: „probable branch“ vs. „branch“
 - komplizierter: Sprungvorhersage (eigenes Thema)

Pipeline-Hemmnisse: ablaufbedingt (2)

Beispiel: Berechne $\sum_{i=1}^9 i$ und $2 \sum_{i=1}^9 i$

i	IS \$1												
sum	IS \$2	1	2	3	4	5	6	7	8	9	10	11	12
	SET i,10	F	D	X	M	W							
	SET sum,0		F	D	X	M	W						
loop	SUB i,i,1			F	D	X	M	W					
	ADD sum,sum,i				F	D	X	M	W				
	BNZ i,loop					F	D	X	M	W			
cont	STB sum,erg						F	D					
	ADD \$3,sum,sum						F						
	STB \$3,erg2												

Erst nach Schritt 7 ist klar, dass gesprungen wird

Zwei Stufen der Pipeline löschen

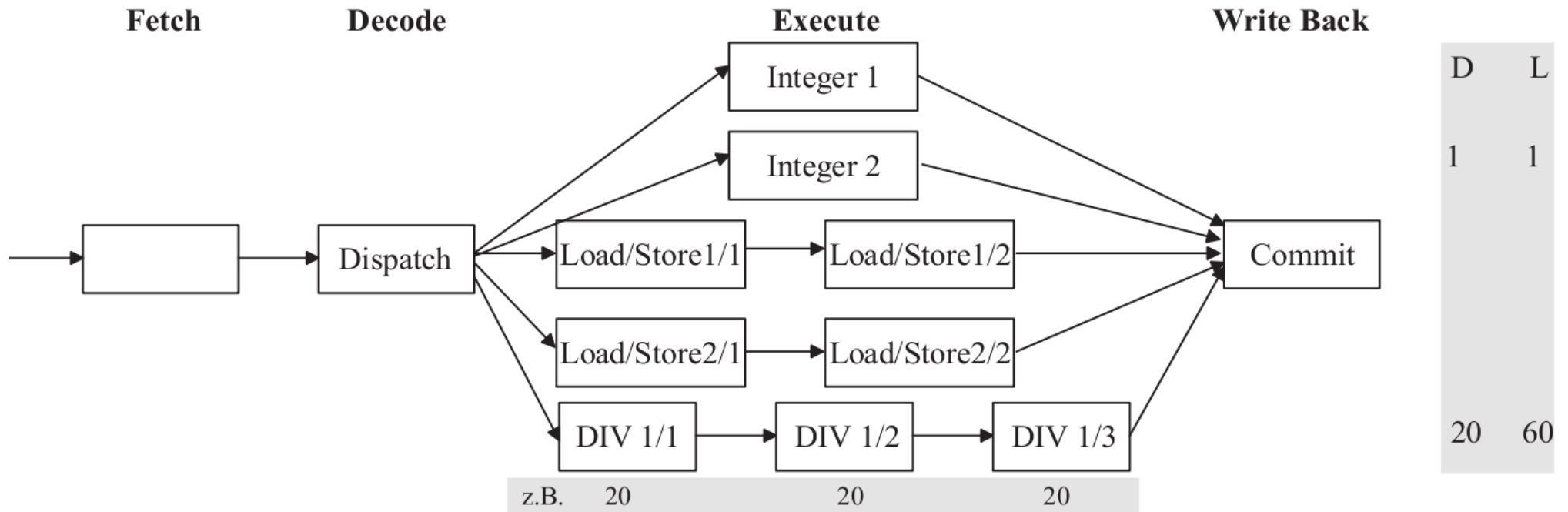
- Auf externe (asynchrone) Interrupts muss die CPU schnell reagieren und in den Interrupt-Handler verzweigen
- Es dürfen keine Befehle unterbrochen werden, die bereits einen Teil der Zustandsänderungen bewirkt haben
- Beispiel: STB \$1, label
 - nicht zwischen M- und W-Phase unterbrechen!
 - M-Phase: Wert aus \$1 in Speicher schreiben
 - W-Phase: Falls Überlauf, Register rA (*arithmetic status register*) anpassen

- Darum Befehle in „später“ Pipeline-Phase (X oder M), noch – vor Interrupt-Behandlung – fertig bearbeiten
- Befehle, die gerade erst dekodiert werden, können aber wieder verworfen werden

Superskalare Architekturen

- Idee: Parallelisierung durch mehrere (potenziell) parallel arbeitende Ausführungseinheiten (Funktionseinheiten)
- Die Funktionseinheiten können für sich wiederum mehrere Pipelinestufen enthalten
- Der Prozessor „entdeckt“ Möglichkeiten zur Parallelverarbeitung (*impliziter* Parallelismus)
(*expliziter* Parallelismus: vom Programmierer festgelegt;
Intel EPIC: **E**xplicit **P**arallel **I**nstruction **C**omputing;
VLIW-Befehle – **V**ery **L**arge **I**nstruction **W**ord)

Superskalare Architekturen (2)



- **Durchsatzzeit (D):** Minimaler Abstand zwischen zwei Befehlen, die eine Ausführungseinheit verlassen
- **Latenzzeit (L):** Minimale Laufzeit eines Befehls durch eine Ausführungseinheit

Einführendes Beispiel: Parallele Pipelines (1/3)

- abhängige Pipelines
- unabhängige Pipelines

Beispiel: Rechnung aus Mandelbrot-Programm

```

23          FADD    p,p,plow
24          FADD    q,q,qlow
25          SET     xk,0
26          SET     yk,0
27          SET     k,0
28      * Nächste Iteration:  $x_{k+1} = x_k^2 - y_k^2 + p$ 
29      1H    INCL    k,1
  
```


Einführendes Beispiel: Parallele Pipelines (2/3)

- abhängige Pipelines (z. B. Pentium)
 - Instruktionen müssen synchron durch die Pipelines laufen: gleichzeitiger Wechsel in nächste Stufe
 - Befehle müssen also ggf. „aufeinander warten“
 - Pipelines heißen **U** und **V** (wie bei Pentium)

	1	2	3	4	5	6	7	8	9	Pipeline
FADD q, q, qlow	F	D	X	X	X	X	M	W		U
SET xk, 0	F	D	X	**	**	**	M	W		V
SET yk, 0		F	D	**	**	**	X	M	W	U
SET k, 0		F	D	**	**	**	X	M	W	V
INCL k, 1			F	**	**	**	D	X	M	U

Einführendes Beispiel: Parallele Pipelines (3/3)

- unabhängige Pipelines
 - nicht synchron durch die Stufen; kein Warten
 - mehr „Logistik“ für Steuerung der Stufen nötig

	1	2	3	4	5	6	7	8	Pipeline
FADD q,q,q_low	F	D	X	X	X	X	M	W	U
SET xk,0	F	D	X	M	W				V
SET yk,0		F	D	**	**	**	X	M	U
SET k,0		F	D	X	M	W			V
INCL k,1			F	**	**	**	D	X	U

- ○: „**out-of-order completion**“ (Fertigstellung in falscher Reihenfolge)

- Problem: Ergebnis eines späten Befehls vor Fertigstellung eines vorangehenden Befehls verfügbar
- Befehle sind aber voneinander abhängig
- Pipeline muss Befehl evtl. anhalten, bis Ergebnisse aus logisch vorangehenden Befehlen vorliegen
 - kann wieder Result Forwarding verwenden

- Zwei Pipelines (U, V); abhängig
- Zwei Pipelines; unabhängig
⇒ out of order completion
- Superskalare Pipelines allgemein: unabhängig und evtl. mehr als zwei
⇒ Es kann (wie bei zwei unabhäng. Pipelines) verschiedene Datenabhängigkeiten geben
- Ziel ist immer: Ergebnis muss identisch mit dem bei sequentieller Ausführung sein

- Keine Abhängigkeit:

```

30      FMUL  temp1, xk, xk       $x_k^2$ 
31      FMUL  temp2, yk, yk       $y_k^2$ 

```

(kann man einfach vertauschen)

- **Read-after-Write-Abhängigkeit (RAW):**

```

50      ADD   temp2, temp2, bidx
51      STBU  k,    temp1, temp2

```

tauschen verboten (sonst wird in #51 der Wert an der falschen Adresse temp1+temp2 gespeichert)

- **Write-after-Read-Abhängigkeit (WAR):**

```

35      FMUL  yk,    xk, yk
36      SETH  xk,    #4000      2, 0 (Gleitkommawert!)
37      FMUL  yk,    yk, xk      2 * yk

```

- #36 muss zwischen #35 und #37 bleiben
- #35/#36 tauschbar, falls anderes Register verwendet wird
- auch: **Antidependence**, „falsche“ Abhängigkeit
- außerdem: **Name Dependence**

- **Write-after-Write-Abhängigkeit (WAW):**

```

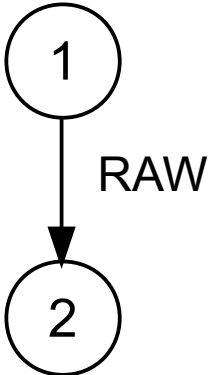
37      FMUL  yk,    yk,xk    2 * yk
38      FADD  yk,    yk,q

```

- Beide Befehle schreiben yk. Würde #38 vor #37 fertiggestellt (Write-Back-Phase), wäre das Ergebnis falsch
- auch: **Output Dependence**
- außerdem: **Name Dependence**

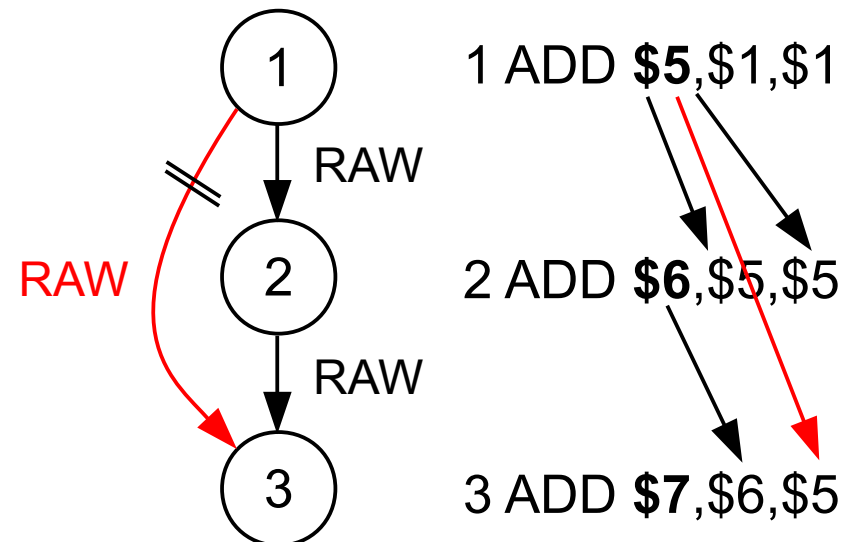
RAW	Read-After-Write true dependence	Befehl <i>B</i> verwendet das Ergebnis von Befehl <i>A</i> (Vorgänger). Konflikt, falls Ergebnis noch nicht da
WAR	Write-After-Read antidependence (name dependence)	<i>B</i> überschreibt Wert, den der Vorgänger <i>A</i> benötigt. Konflikt, falls <i>B</i> schreibt, bevor <i>A</i> liest
WAW	Write-After-Write output dependence (name dependence)	<i>A</i> und <i>B</i> schreiben in dasselbe Register (oder dieselbe Speicherzelle) Konflikt, wenn <i>B</i> vor <i>A</i> schreibt
RAR	Read-After-Read	<i>A</i> und <i>B</i> lesen denselben Wert; <i>kein Konflikt</i>

- Abhängigkeiten grafisch darstellen

- 

bedeutet eine RAW-Abhängigkeit zwischen Instruktionen 1 und 2, also: (2) liest Wert, den (1) schreibt

- mehrere Abhängigkeiten gleichzeitig: alle angeben (z. B. „RAW, WAW“)
- keine **transitiven** Abhängigkeiten einzeichnen



Abhängigkeitsgraph (2)

Auszug aus einem Beispiel-Programm

```

30      FMUL    temp1,xk,xk
31      FMUL    temp2,yk,yk
32      FSUB    temp1,temp1,temp2
33      FADD    temp1,temp1,p
34 *  $y_{k+1} = 2x_k y_k + q$ 
35      FMUL    yk, xk,yk
36      SETH    xk, #4000

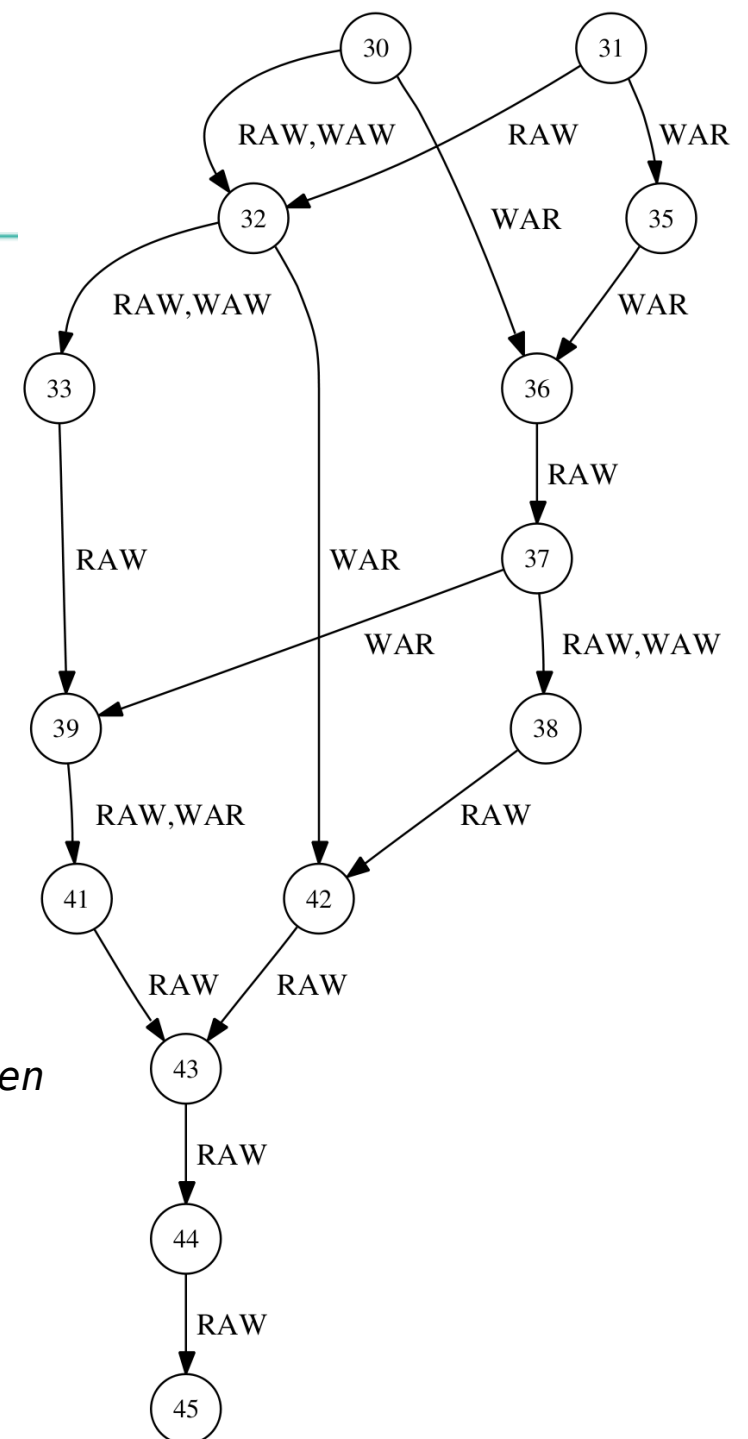
37      FMUL    yk, yk,xk
38      FADD    yk, yk,q
39      SET     xk, temp1

40 *  $r = x_{k+1}^2 + y_{k+1}^2$ 
41      FMUL    temp1,xk,xk
42      FMUL    temp2,yk,yk
43      FADD    r, temp1,temp2
44      FCMP    test, r,:M
45      BNP     test, 2F
    
```

x_k^2
 y_k^2
 x_{k+1}

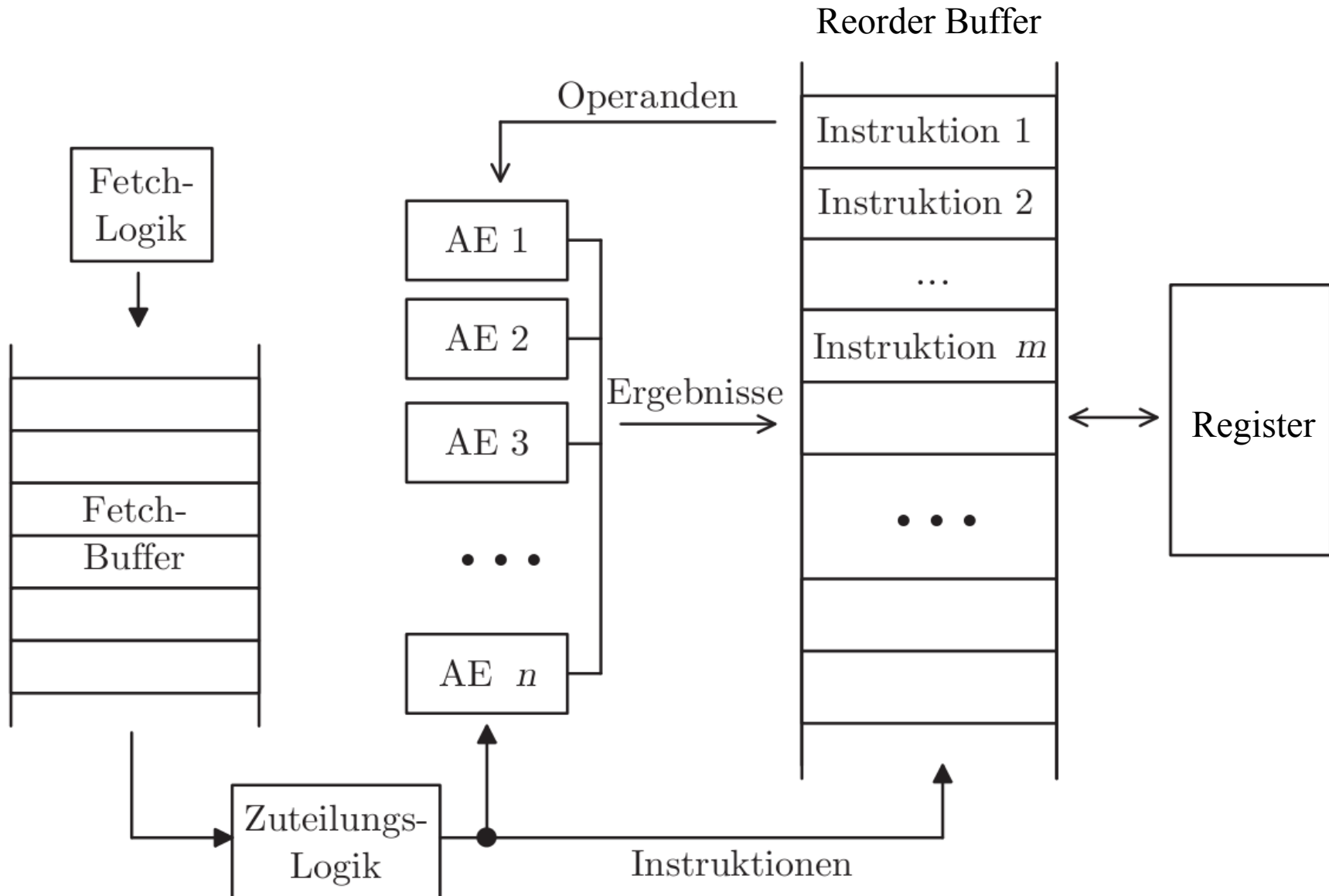
$2, 0$ (Gleit-
kommawert!)
 $2 * y_k$

x_k kann über-
schrieben werden



- **Instruction Fetch:** Es steht ein Puffer zur Verfügung (Fetch buffer), in dem mehrere Instruktionen bereit stehen können.
- **Instruction Issue:** Instruktionen werden der Reihe nach (in order) auf freie Ausführungseinheiten (functional units, FUs) verteilt („issued“).
Dieser Prozess stoppt, falls keine freien FUs vorhanden sind.
- **Read Operands:** Instruktionen warten ggf. auf ihre Operanden (RAW); ab hier ist Überholen möglich.
- **Execute:** out of order (OOO). Instruktionen warten, bis alle WAR-Vorgänger fertig sind.
- **Write Back:** Endgültiges Zurückschreiben in order: Write-after-Write (WAW) beachten. Ergebnisse stehen allerdings schon vorher zur Verfügung (Forwarding).

Reorder Buffer



1. Freie Ausführungseinheit suchen.
Falls verfügbar, durch den Befehl belegen und einen Eintrag im **Reorder Buffer (ROB)** anlegen
2. Operanden bereitstellen. Für jeden Operanden den ROB von unten nach oben durchsuchen:
 - Wert aus Register (kein Ergebnis eines Bef. im ROB) *oder*
 - Ergebnis eines vorangehenden (fertigen) Befehls *oder*
 - vorangehenden Befehl beobachten, falls er noch nicht fertig ausgeführt ist (Pause!)
3. Befehl ausführen. Ergebnis im Eintrag des Befehls im ROB vermerken
4. Befehl bestätigen (commit): Ergebnisse in die Register übernehmen

t_0			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL y_k, y_k, x_k	MUL	►	\$4	\$3	—	—	\$4 (ungültig)
FADD y_k, y_k, q	FPU	II	—	\$2	FMUL	—	\$4 (ungültig)

$q = \$2$
 $x_k = \$3$
 $y_k = \$4$

- Befehl
- Unit (welche Ausführungseinheit?)
- Status: ► = läuft, II = Pause, ✓ = fertig
- Operanden: Q_Y , Q_Z : Werte der zwei Operanden, evtl. noch nicht verfügbar
- V_Y , V_Z : ggf. Zeiger auf ROB-Eintrag $\Rightarrow$ dann Q_Y bzw. Q_Z bereits (im ROB) verfügbar
- Ziel

Beispiel: Reorder Buffer (1)

- wieder unser Beispiel-Programm; diesmal Zeilen 34-41

34 $* y_{k+1} = 2 x_k y_k + q$

35 FMUL yk,xy,yk

36 SETH xk,#4000

37 **FMUL yk,yk,xk**

38 **FADD yk,yk,q**

39 **SET xk,temp1**

40 $* r = x_{k+1}^2 + y_{k+1}^2$

41 **FMUL temp1,xk,xk**

das schauen
wir uns an ...

Legende

q = \$2

xk = \$3

yk = \$4

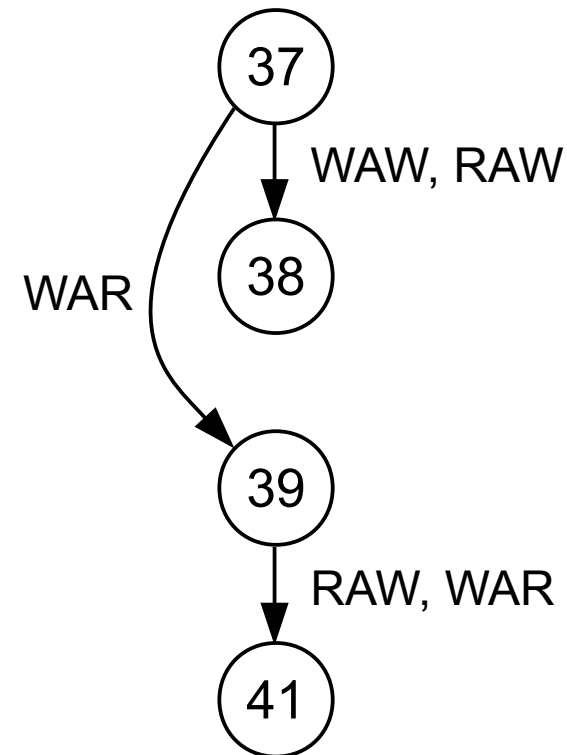
temp1 = \$10

Beispiel: Reorder Buffer (2)

- Abhängigkeitsgraph:

```

37  FMUL  yk,    yk,xk
38  FADD  yk,    yk,q
39  SET   xk,    temp1
40
41  FMUL  temp1, xk,xk
    
```



Annahmen:

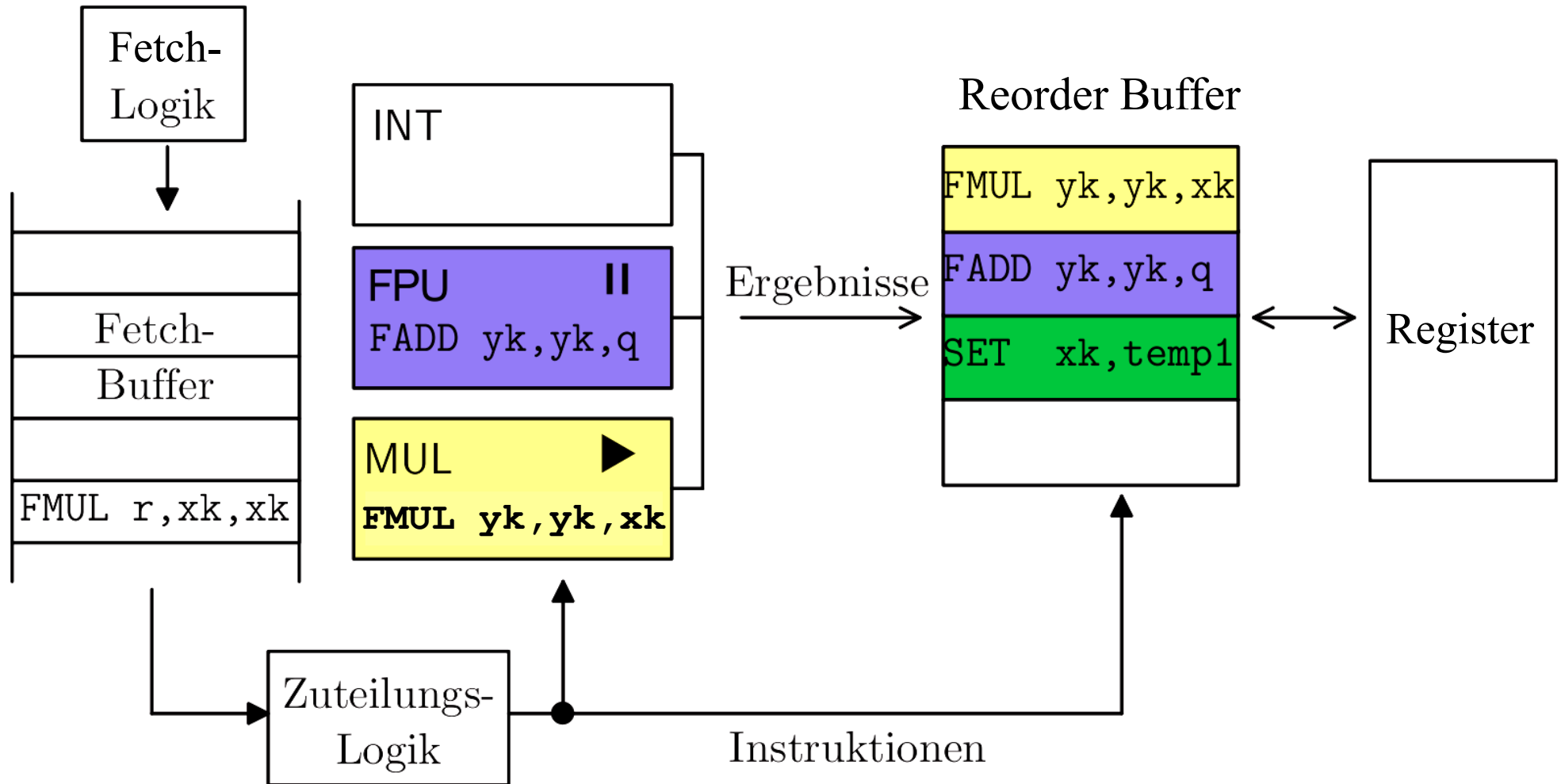
- 1 FPU (alle Floating-Point-Ops außer FMUL)
- 2 Integer-Einheiten (INT1, INT2), alles außer Multiplikation
- 1 MUL (FP- und Integer-Multiplikation)
- FPU und MUL: je 4 Takte, INT: je 1 Takt
- Reorder Buffer hat Platz für 4 Einträge
- Je Takt max. 2 Befehle zuteilen (issue) und max. 2 Befehle bestätigen (commit)

q = \$2, xk = \$3, yk = \$4, temp1 = \$10

t_0			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL yk,yk,xk	MUL	▶	\$4	\$3	—	—	\$4 (ungültig)
FADD yk,yk,q	FPU		—	\$2	FMUL	—	\$4 (ungültig)
—							
—							

t_1			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL yk,yk,xk	MUL	▶	\$4	\$3	—	—	\$4 (ungültig)
FADD yk,yk,q	FPU		—	\$2	FMUL	—	\$4 (ungültig)
SET xk,temp1	INT1	▶	\$10	×	—	—	\$3 (ungültig)
—							

t_2 und t_3			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL yk,yk,xk	MUL	▶	\$4	\$3	—	—	\$4 (ungültig)
FADD yk,yk,q	FPU		—	\$2	FMUL	—	\$4 (ungültig)
SET xk,temp1	—	✓	\$10	×	—	—	\$3
—							



$q = \$2, \quad xk = \$3, \quad yk = \$4, \quad temp1 = \10

$q = \$2$, $xk = \$3$, $yk = \$4$, $temp1 = \$10$

t_4			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL yk, yk, xk	—	✓	\$4	\$3	—	—	\$4
FADD yk, yk, q	FPII	▶	\$4*	\$2	—	—	\$4 (ung.)
SET $xk, temp1$	—	✓	\$10	×	—	—	\$3
FMUL $temp1, xk, xk$	MUL	▶	\$3*	\$3*	—	—	—

t_5			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FADD yk, yk, q	FPU	▶	\$4*	\$2	—	—	\$4 (ung.)
SET $xk, temp1$	—	✓	\$10	×	—	—	\$3
FMUL $temp1, xk, xk$	MUL	▶	\$3*	\$3*	—	—	\$10 (ung.)
—							

*) Wert aus ROB übernommen (Forwarding)

Beispiel: Reorder Buffer (7)

Klassisch:

			0	1	2	3	4	5	6	7	8		
FMUL yk, yk,xk	F	D	Xm	Xm	Xm	Xm	W						(Xm: 0-3)
FADD yk, yk,q	F	D	-	-	-	-	Xf	Xf	Xf	Xf	W		(Xf: 4-7)
SET xk, temp1		F	D	Xi	W								(Xi: 1)
FMUL temp1, xk,xk		F	D	-	-	-	Xm	Xm	Xm	Xm	W		(Xm: 4-7)

Leicht geändertes Programm:

			0	1	2	3	4	5	6	7	8		
FMUL yk, yk,xk	F	D	Xm	Xm	Xm	Xm	W						(Xm: 0-3)
FADD yk, yk,q	F	D	-	-	-	-	Xf	Xf	Xf	Xf	W		(Xf: 4-7)
SET yk, temp1		F	D	-	-	-	-	-	-	-	Xi	W	(Xi: 8)
FMUL temp1, yk,yk		F	D	-	-	-	-	-	-	-	-	Xm	(Xm: 9-)

Register Renaming (1)

- Jeder Eintrag im Reorder Buffer benötigt Register zum Speichern von Operanden und spekulativen (= noch nicht bestätigten) Ergebnissen.
- Diese heißen **Schattenregister** oder **Rename Registers** (im Unterschied zu den **Befehlssatz-Registern**; **Architectural Registers**)
- *Was wird denn hier gespeichert?*

Register Renaming (2)

- Was im Reorder Buffer steht...

Befehlssatz-
Register 0..n

t_0			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL y_k, y_k, x_k	MUL	▶	\$4	\$3	—	—	\$4 (ungültig)
FADD y_k, y_k, q	FPU	■	—	\$2	FMUL	—	\$4 (ungültig)
—							
—							

Register-Nummer (0..n)

dito

dito (nur Hinweis, wo Ergebnis später
hin gehört; sowie Status)

Verweis auf Ergebnis im ROB

Reg.-Nr. (0..n) für Ziel, Status **und(!)** Ergebniswert

t_4			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL y_k, y_k, x_k	—	✓	\$4	\$3	—	—	\$4
FADD y_k, y_k, q	FPU	▶	\$4*	\$2	—	—	\$4 (ung.)
SET $x_k, temp1$	INT1	✓	\$10	×	—	—	\$3
FMUL $temp1, x_k, x_k$	MUL	▶	\$3*	\$3*	—	—	—

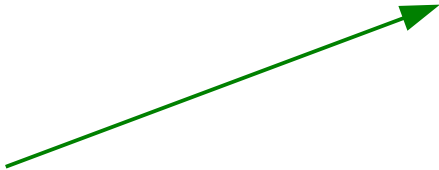
- Pro „Zeile“ im ROB muss die Ziel-Spalte also Platz bieten für
 - eine Register-Nummer (Zielregister)
 - Status
 - einen berechneten Wert (in „Registergröße“, diese Teile des ROB heißen „Rename-Register“)
- Es ist ökonomisch sinnvoll, lediglich Pointer auf Rename Register in einem Register-Pool (für Zwischenergebnisse) zu speichern.

Register Renaming (4)

Register-Pool:

Wert	frei	gültig
	+	-
	-	+
	-	+
	-	-
	+	-
	-	+
	-	-
	-	+
	+	-
	-	+

Pointer aus ROB-
Ziel-Spalte



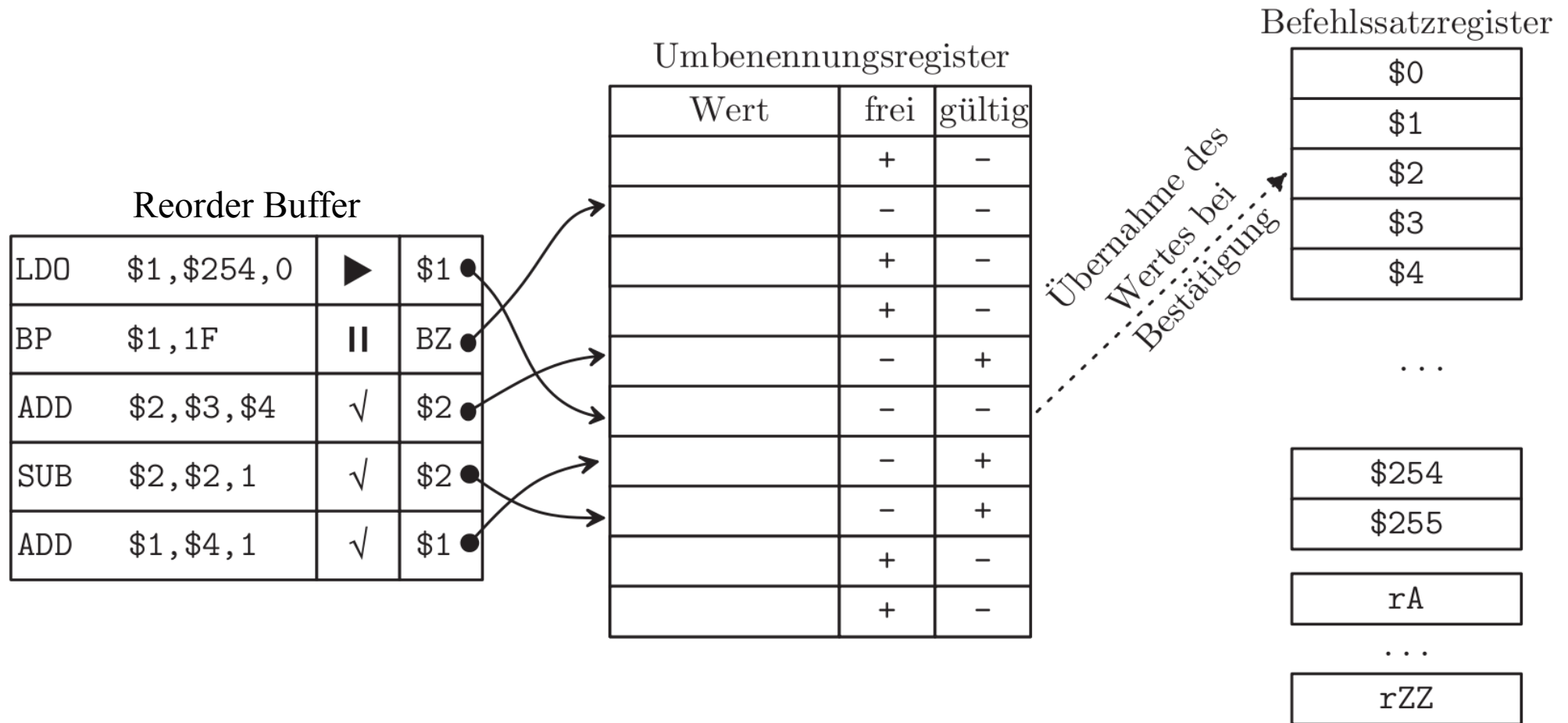
frei: Eintrag
nicht benutzt

gültig: enthält
fertig berech-
neten Wert,
kann für wei-
tere Berech-
nungen ge-
nutzt werden

Implizite Umbenennung

- Zwei getrennte Sätze physischer Register für Befehlssatz-Register und Rename Register
- Es ist nirgends explizit vermerkt, wo sich der jüngste spekulative Wert eines Registers befindet. Finden des Wertes: ROB durchsuchen
 - Neuer Befehl `ADD $1, $3, $3` kommt in den ROB.
„Welches“ \$3 nehmen? Reg.-Inhalt? ROB-Inhalt?
Warten auf Berechnung von \$3?
- Das Verwerfen spekulativer Werte (bei Interrupts oder falsch vorhergesagten Sprüngen) ist einfach.

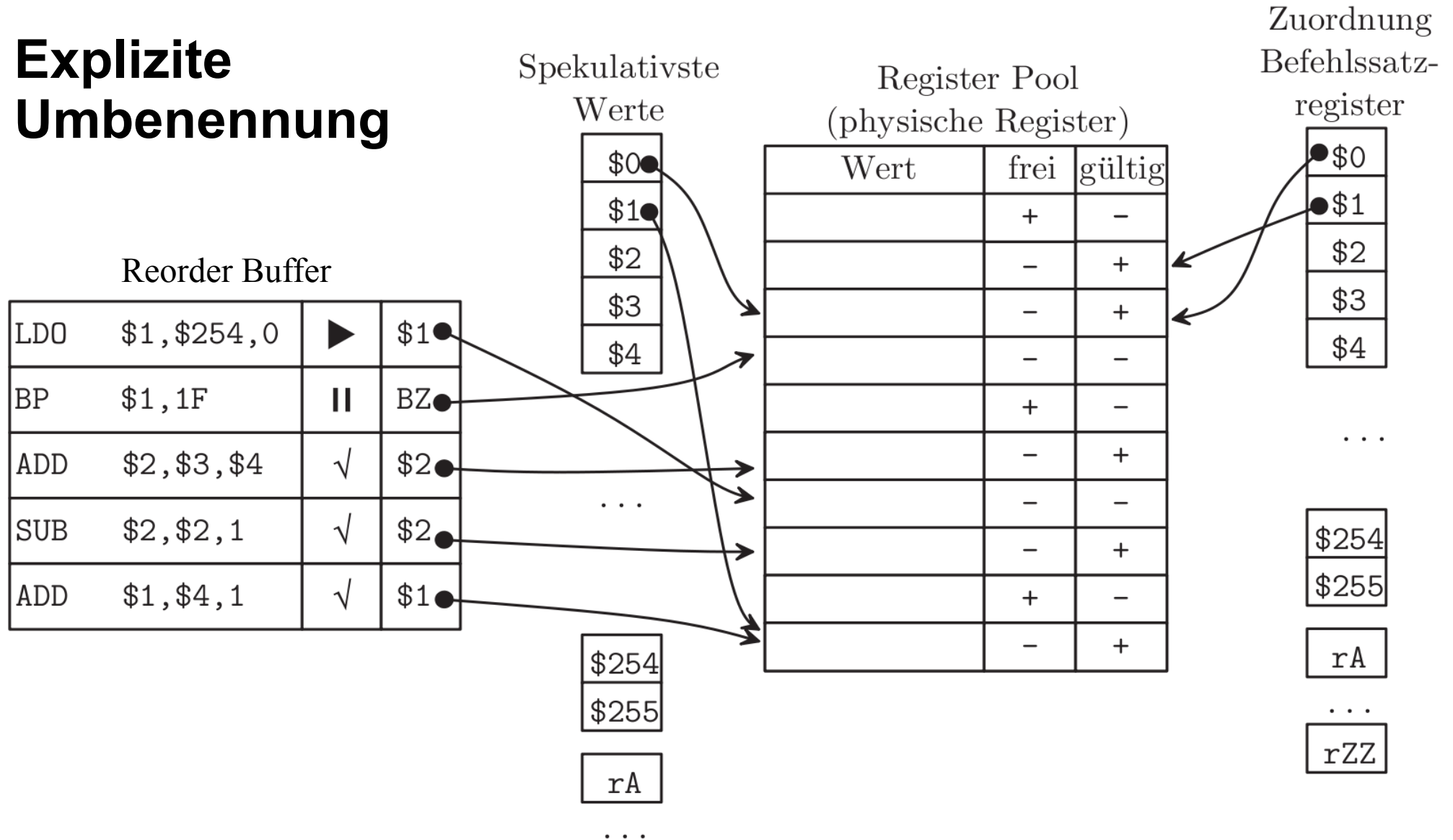
Implizite Umbenennung



Explizite Umbenennung

- Es gibt nur einen Register-Pool (gemeinsam für Befehlssatz- und Rename-Register)
- Eine Tabelle gibt an, wo die jüngsten (spekulativsten) Werte der Register stehen
- Kein Umkopieren der Register nach Bestätigen eines Befehls;
Pointer auf die physischen Register beschreiben Registerzustand

Explizite Umbenennung



Datenabhängigkeiten

- **Read-after-Write (RAW):**
führt stets zum Stillstand, bis das Ergebnis zur Verfügung steht.
- **Write-after-Read (WAR) und Write-after-Write (WAW):**
Schattenregister können Stillstand verhindern (wenn es genügend freie gibt)

- „Register Renaming“ auch direkt aus dem Wort verständlich:

1. $R1 = M[1024]$		Nach Umbenennen von R1 in R2:	
2. $R1 = R1 + 2$			
3. $M[1032] = R1$		1. $R1 = M[1024]$	4. $R2 = M[2048]$
4. $R1 = M[2048]$		2. $R1 = R1 + 2$	5. $R2 = R2 + 4$
5. $R1 = R1 + 4$		3. $M[1032] = R1$	6. $M[2056] = R2$
6. $M[2056] = R1$			

WAR-Konflikt zwischen (3) und (4); Befehl (4) darf erst schreiben, wenn (3) fertig ist

- echte Beispiel-Architekturen:
 - Pentium Pro: bis zu 16 Stufen
 - PowerPC 970: 16–24 Stufen

Anhang: Mandelbrot-Programm (1/2)

```

01      PREFIX :Mandel:
02      LOC    #100
03      plow   GREG    #c002000000000000    -0,025
04      pdelta GREG    #3f83333333333333    0,009375=(0,75- (-2,25))/320
05      qlow   GREG    #bff8000000000000    -1,5
06      qdelta GREG    #3f8999999999999a    0,0125=(1,5 - (-1,5))/240
07      p      IS      $1
08      q      IS      $2
09      xk      IS      $3          x_k
10      yk      IS      $4          y_k
11      k      IS      $5          Iteration k
12      r      IS      $6
13      bildx   IS      $7          Gerätekoordinaten (integer)
14      bildy   IS      $8
15      test    IS      $9
16      temp1   IS      $10         Zwischenergebnisse
17      temp2   IS      $11
18
19      :Mandel  FLOT    p,bildx
20              FLOT    q,bildy
21              FMUL    p,p,pdelta
22              FMUL    q,q,qdelta
23              FADD    p,p,plow
24              FADD    q,q,qlow
25              SET     xk,0
26              SET     yk,0
27              SET     k,0
28      * Nächste Iteration:  $x_{k+1}=x_k^2-y_k^2+p$ 
29      1H      INCL    k,1
30              FMUL    temp1,xk,xk     $x_k^2$ 
31              FMUL    temp2,yk,yk     $y_k^2$ 
32              FSUB    temp1,temp1,temp2
33              FADD    temp1,temp1,p     $x_{k+1}$ 

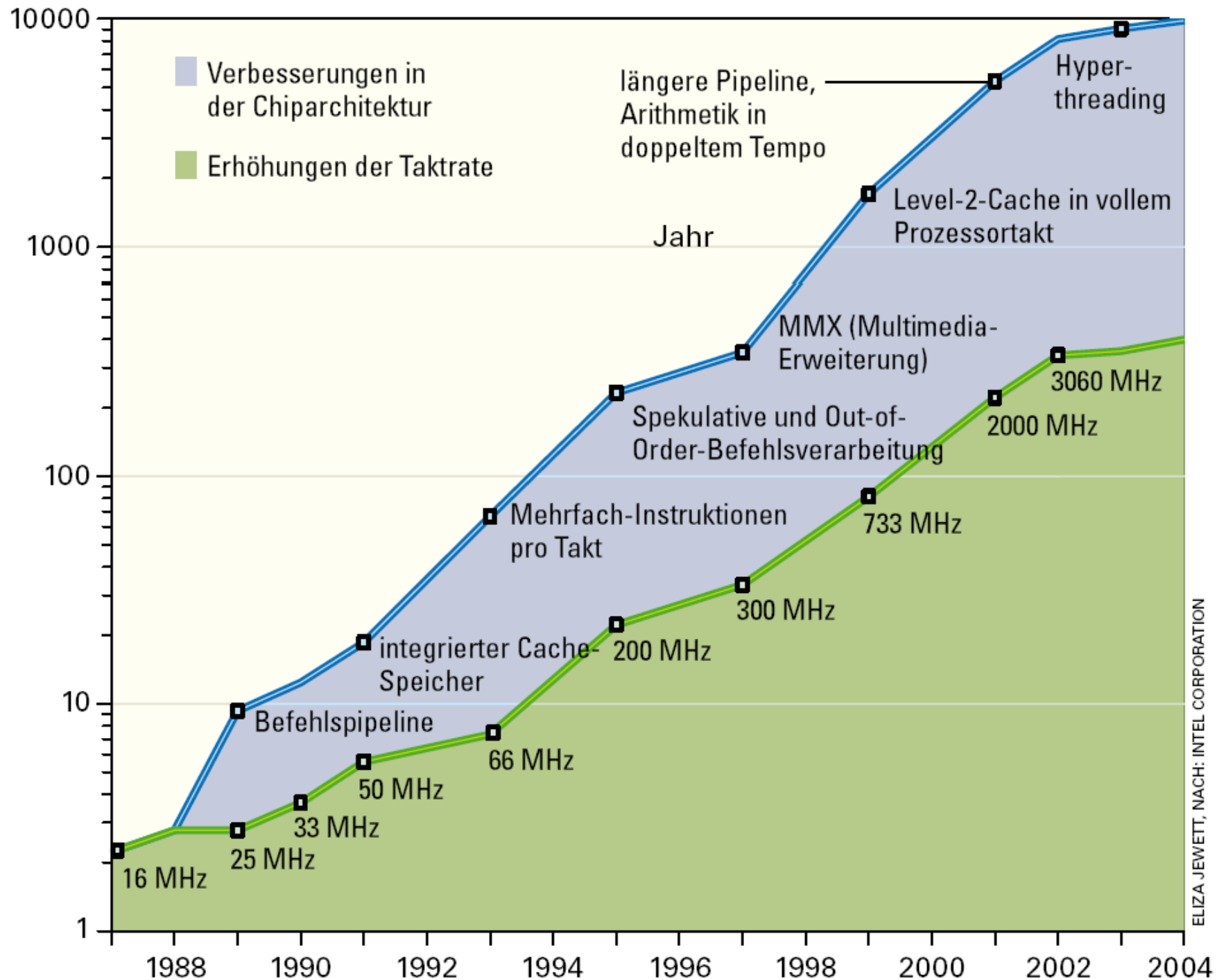
```

Quelle: Böttcher, „Rechneraufbau und Rechnerarchitektur“

Mehrprozessor- und Multi-Core-Systeme

- Uni-Prozessoren, ohne Pipeline
- Beschleunigen:
 - Prozessortakt (hat Grenzen)
 - Pipelining, skalar und superskalar (hat auch Grenzen)
 - mehr Leistung nur noch durch echte Parallelität erreichbar, also:
mehr als eine CPU
→ **Multiprozessor- und Multi-Core-Systeme**

theoretische maximale Rechenleistung
in Millionen Operationen pro Sekunde



- Einfaches Problem: zehn unabhängige Aufgaben parallel bearbeiten
 - z. B.: zehn separate Rechner einsetzen, perfekt parallelisierbar
- Schwierigeres Problem: eine komplexe Aufgabe parallel bearbeiten
 - wie aufteilen? Automatismus?

- Cluster: mehrere unabhängige Rechner, nur durch Netzwerk verbunden
- Multi-Prozessor: mehrere CPUs auf Hauptplatine
- Multi-Core: mehrere vollwertige CPUs-Kerne in einem CPU-Chip
- Hyper-Threading: mehrere logische CPUs in einem CPU-Chip (auch kombinierbar mit Multi-Core)

- hardwareseitiges Multithreading (Intel)
- mehrere vollständige Registersätze und ein komplexes Steuerwerk
- parallel arbeitende Pipeline-Stufen (aber genauso viele Ausführungseinheiten wie in „normaler“ CPU)
- aus BS-Sicht: mehrere (virtuelle) CPUs
- mehrere parallele Befehls- und Datenströme (Threads) werden auf diese parallelen Stufen verteilt
- (erhöht die Anzahl *unabhängiger* Instruktionen in der Pipeline)

- mehrere CPUs auf einem Chip
- alles mehrfach vorhanden (außer L2 Cache und höher sowie Bus)
- aus BS-Sicht: mehrere (echte) CPUs
- aktuell üblich: 2-/4-/6-/8-/12-/16-Core
- Beispiele:
 - AMD Opteron 16-Core
 - Intel Tera-Scale, Teraflops Research Chip (Polaris, 80 Cores)

- mehrere CPUs auf einem Mainboard
- weniger effiziente (ältere) Variante von Multi-Core-Systemen
- auch hier aus BS-Sicht: mehrere echte CPUs

Bild: Wikipedia, <http://de.wikipedia.org/wiki/Mehrprozessorsystem>

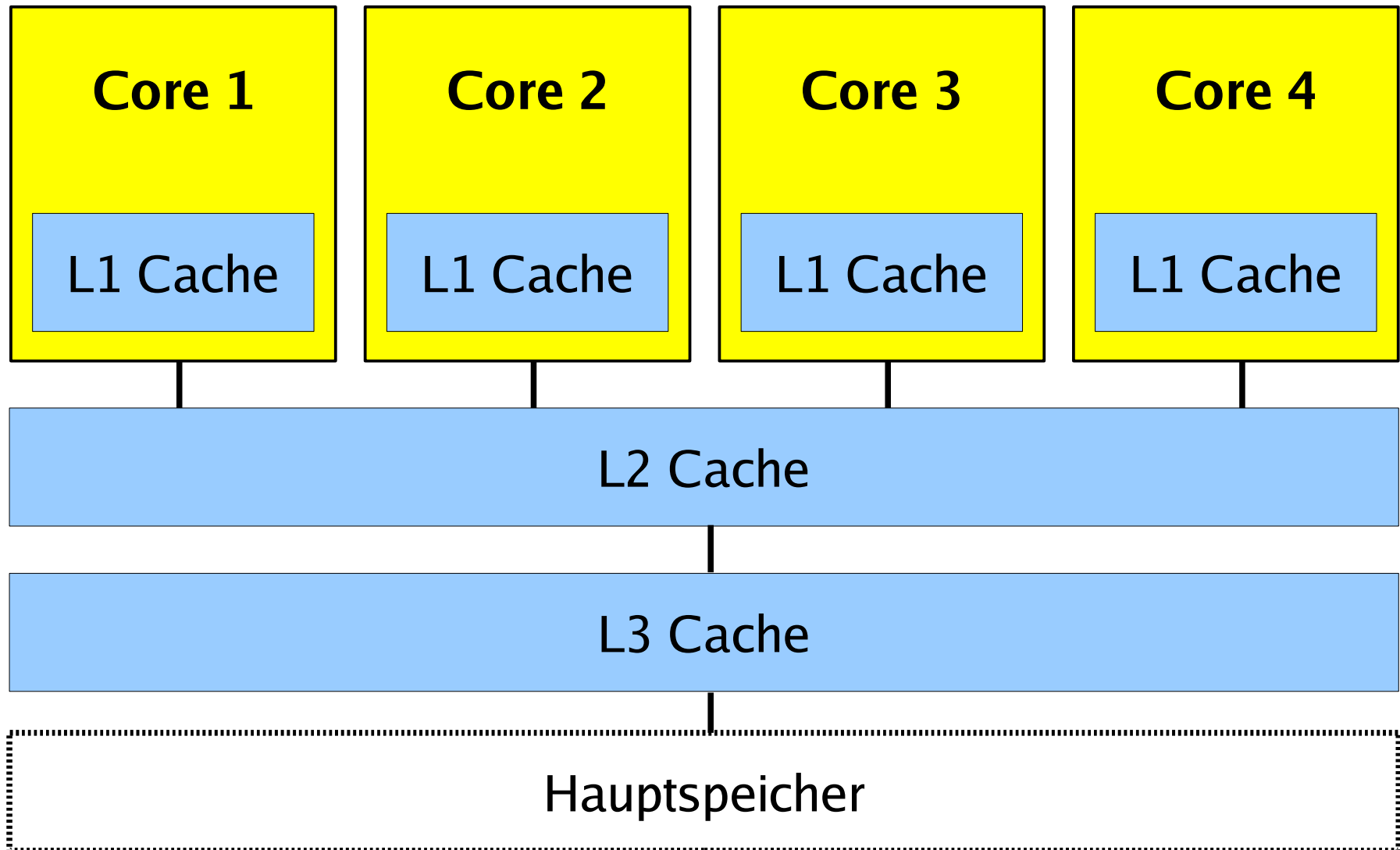


- mehrere Rechner mit je einer oder mehreren CPUs
- lokaler Speicher in jedem Rechner
- „lose gekoppeltes System“
- verteilte Anwendungen, die gleichzeitig auf mehreren Rechnern arbeiten
- Austausch zwischen Rechnern über Netzwerk

- MOSIX2: <http://www.mosix.org/>
- Linux-basierter Cluster mit
 - automatischem Load-Balancing
 - Prozess-Migration
 - migrierbare Sockets
 - technisch: Virtualisierungsschicht
- am besten für Anwendungen mit wenig I/O geeignet

- Zugriff auf (gemeinsamen!) Hauptspeicher
 - Anbindung an RAM über gemeinsam genutzten Bus
 - Was tun bei parallelen Zugriffen auf den Hauptspeicher?
 - Parallel Write: Wer setzt sich durch?
- Cache
 - interner Cache: was tun, wenn mehrere (echte oder virtuelle) Kerne die gleiche Speicheradresse cachieren?
 - Stichworte: Cache-Kohärenz, Cache-Konsistenz

Cache-Hierarchie bei Multi-Core



- konsistente Daten in allen Caches
- Beispiel:
 - CPU 1 liest Mem[x] und speichert Cache-Line im lokalen CPU-Cache
 - CPU 2 liest auch Mem[x] und speichert Cache-Line im lokalen CPU-Cache
 - CPU 1 schreibt Mem[x] und aktualisiert dabei auch den lokalen Cache
 - CPU 2 liest Mem[x] – was steht im lokalen Cache?

- Idee: Zu jedem Zeitpunkt ist ein bestimmter Wert Z für eine Speicherzelle $\text{Mem}[x]$ gültig (der zuletzt geschriebene)
- Jeder Prozessor, der $\text{Mem}[x]$ liest, sollte Z erhalten
- Unmittelbar nach Schreiben von $\text{Mem}[x]$ muss man eine kurze Verzögerung akzeptieren, in der es „unterschiedliche Meinungen“ über $\text{Mem}[x]$ gibt

- Cache-Kohärenz-Protokolle garantieren Kohärenz
- zwei Ansätze
 - Verzeichnis: zentrale Liste mit dem Status aller Cache-Lines (in allen Caches)
 - Liste der CPUs mit Read-only-Kopie (Status *Shared*)
 - CPU mit exklusivem Schreibzugriff (Status *Exclusive*)
 - **Snooping:** Alle Cache Controller lauschen auf Speicherbus und erkennen Schreib- und Lesezugriffe auf Cache-Line, die sie auch speichern

MESI Cache Coherence Protocol (1)

- Ziel: Verwalten, wo der aktuell(st)e Inhalt Mem[x] einer Speicherzelle x zu finden ist
- Für jede Cache-Line vier mögliche Zustände **M**, **E**, **S**, **I**:
 - **Modified**: Cache-Line nur im lokalen Cache, „*dirty*“: wurde verändert → Cache muss Daten ins RAM zurück schreiben, bevor weitere Lesezugriffe auf diese Adresse im RAM erlaubt sind. Nach dem Zurückschreiben Zustandsänderung in **Exclusive**.
 - **Exclusive**: Cache-Line nur im lokalen Cache, „*clean*“: identisch mit RAM. Kann jederzeit in Status **Shared** wechseln, wenn andere CPU den Wert lesen will. Auch Wechsel zu **Modified** möglich, wenn Wert überschrieben wird.

MESI Cache Coherence Protocol (2)

- (vier Zustände...)
- **Shared:** Diese Cache-Line wird evtl. auch in anderen Caches vorrätig gehalten, „*clean*“: identisch mit RAM. Bei Schreibzugriff müssen alle Kopien (in anderen Caches) auf **Invalid** gesetzt werden.
- **Invalid:** Diese Cache-Line ist veraltet (der Wert im Hauptspeicher hat sich geändert); nicht benutzen.

Zustandskombinationen (zwei Caches):

(Quelle: http://en.wikipedia.org/wiki/MESI_protocol)

	M	E	S	I
M	✗	✗	✗	✓
E	✗	✗	✗	✓
S	✗	✗	✓	✓
I	✓	✓	✓	✓

MESI Cache Coherence Protocol (3)

- **Lesezugriff:** In allen Zuständen außer **invalid** erlaubt
- **Schreibzugriff:** nur im (lokalen!) Zustand **modified** oder **exclusive** erlaubt –
Im Zustand **shared** müssen zuerst alle Kopien der Cache-Line auf **invalid** gesetzt werden.

	M	E	S	I
M	✗	✗	✗	✓
E	✗	✗	✗	✓
S	✗	✗	✓	✓
I	✓	✓	✓	✓

- „Wer“ profitiert von mehreren Kernen/CPU's?
 - Takterhöhung beschleunigt jede Anwendung
 - Pipelining beschleunigt (automatisch) die meisten Anwendungen
 - Einsatz mehrerer Kerne / CPU's / HT:
 - zunächst gar keine Beschleunigung einer einzelnen Applikation
 - schlimmstes Szenario: Ein Kern durch Anwendung belegt, restliche Kerne untätig
 - Betriebssystem und Anwendungen müssen mehrere Kerne unterstützen

- BS-Support: Scheduler muss das Verteilen von mehreren Prozessen/Threads auf mehrere Kerne unterstützen
- Anwendungs-Support: Anwendung muss „parallelisiert“ sein, also aus mehreren (relativ) unabhängigen Anwendungssträngen (Threads) bestehen

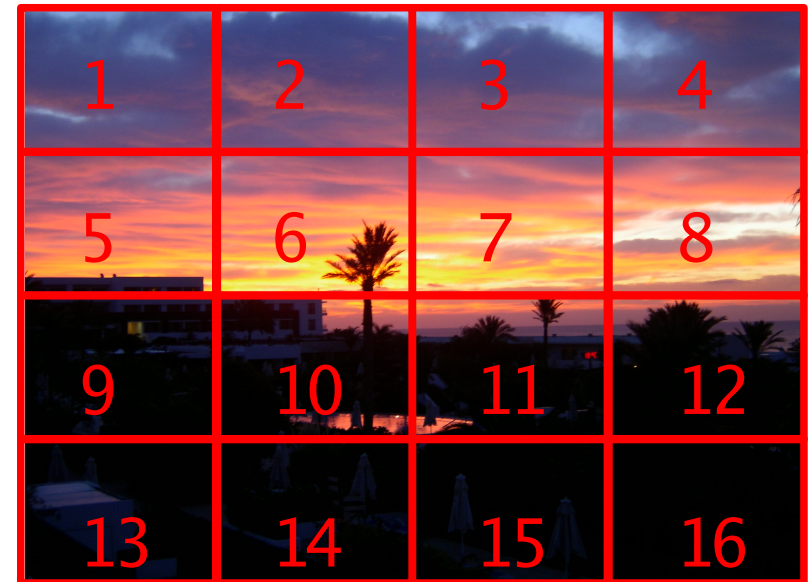
- Master/Worker
 - Ein Master-Thread erhält oder erstellt Aufträge und verteilt diese an ...
 - ... mehrere Worker-Threads: Die rechnen / erledigen je einen konkreten Auftrag, geben das Ergebnis an den Master zurück und beenden sich
 - Beispiel: Apache Webserver



```
[esser@vserver:~]$ pstree -p|grep apache
init(1)-+-apache2(19459)-+-apache2(1798)
                        | -apache2(3400)
                        | -apache2(3976)
                        | -apache2(3977)
                        | -apache2(3978)
                        | -apache2(23603)
                        | -apache2(26067)
                        | -apache2(26238)
                        | -apache2(5995)
                        \ -apache2(16194)
```

Typische Anwendungs-Designs (2)

- Örtliche Parallelisierung
- z. B. Bildverarbeitung:



- einzelne Teilbereiche separat bearbeiten
- regelmäßig „Randinformationen“ austauschen

- Threads in Standardsprache (C, C++, ...) von Hand erstellen
- Standardsprache mit Bibliothek um spezielle Parallelisierungsfunktionen erweitern (z. B. OpenMP, siehe <https://computing.llnl.gov/tutorials/openMP/>)
- spezielle parallele Programmiersprache nutzen
 - Occam, Erlang, Scala, Clojure, Fortress, ...
 - kurze Beschreibungen: z. B. unter <http://pvs.uni-muenster.de/pvs/lehre/SS10/seminar/>

- Beispiel in der Sprache Fortress:

```
for k<-1:5 do
  print k " "
  print k " "
end
```

erzeugt z. B. 4 1 4 1 5 2 5 2 3 3
und nicht 1 1 2 2 3 3 4 4 5 5

- For-Schleife **implizit parallel**
- Während Laufzeit des Programms werden neue Threads erzeugt, die Teile der Schleife berechnen
- alternativ: neue Threads von Hand starten (für klassisches Modell, manuelle Parallelisierung)

- Klassisch / manuell

```
while (true) {  
    req = read_request();           // Warten auf Arbeit  
    T = new WorkerThread(req);      // neuen Thread ...  
    T.start()                       // ... starten  
}
```

```
Class WorkerThread extends Thread {  
    ...  
}
```

- Alternative „gute“ Nutzung eines Mehrkernsystems: Multi-User-Betrieb
- viele Anwender starten eigene Prozesse
- Nichts zu tun: Szenario ist schon parallelisiert

- Parallelprogrammierung, in verschiedenen Hardware-Modellen:
https://computing.llnl.gov/tutorials/parallel_comp/
- Kapitel 5 (Mehrprozessorsysteme) der Vorlesung Rechnerarchitektur, Univ. Dortmund, SS 2009,
<http://ls12-www.cs.tu-dortmund.de/de/patrec/teaching/SS09/rechnerarchitektur/>

IT-Infrastruktur

SS 2015

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz E:

v1.1, 2015/03/05

- Zentrale und verteilte IT-Infrastrukturen
- Zusammenfassung der Vorlesung

Vorlesungsübersicht

Seminar

Wiss. Arbeiten

Datenformate und Wandlung

PC als Arbeitsplatz

Ergonomie und Arbeitsschutz

Rechnerstrukturen

Zentrale / verteilte IT-Infrastrukturen

Folien E

- Parallelität
- Verteilte Systeme
- Client / Server
- Grid
- Cloud
- Virtualisierung

- Cloud-Computing basiert auf Grundlagen in den Bereichen
 - parallele und verteilte Systeme
 - Client/Server, Thin Clients
 - Kommunikationsprotokolle

- Parallele Systeme
 - Probleme lösen, für welche die Ressourcen einer einzelnen Maschine nicht ausreichen
 - Benötigte Zeit für die Problemlösung reduzieren („Speed-up“)
 - Erfolg abhängig vom Anteil α des nicht-parallelisierbaren Codes
 - Amdahls Gesetz (1967):
Speed-up $S(n) \rightarrow 1 / \alpha$ (für $n \rightarrow \infty$)
z. B. $\alpha = 5 \% = 0,05$: $1/0,05 = 20$ ist max. Speed-up

- Parallele Systeme

- Beispiel zu Amdahl:

$\alpha = 5 \%$; Code braucht insgesamt 1000 Zeiteinheiten (ZE), 950 parallelisierbar, 50 sequentiell

n	Par.	Seq.	Summe	Speed-up
1	950	50	1000	1
2	475	50	525	1,90
10	95	50	145	6,90
50	19	50	69	14,49
950	1	50	51	19,6
∞	0	50	50	20,0

- Was verhindert Parallelisierbarkeit?
 - Zwei Ereignisse sind nebenläufig, wenn keines die Ursache des anderen ist.
 - Datenabhängigkeit: z. B. RAW-Konflikt (vgl. Pipelines)
- Was verlangsamt parallele Threads?
 - Synchronisation (Mutex, Semaphore, Barrieren, Inter-Thread-Kommunikation)

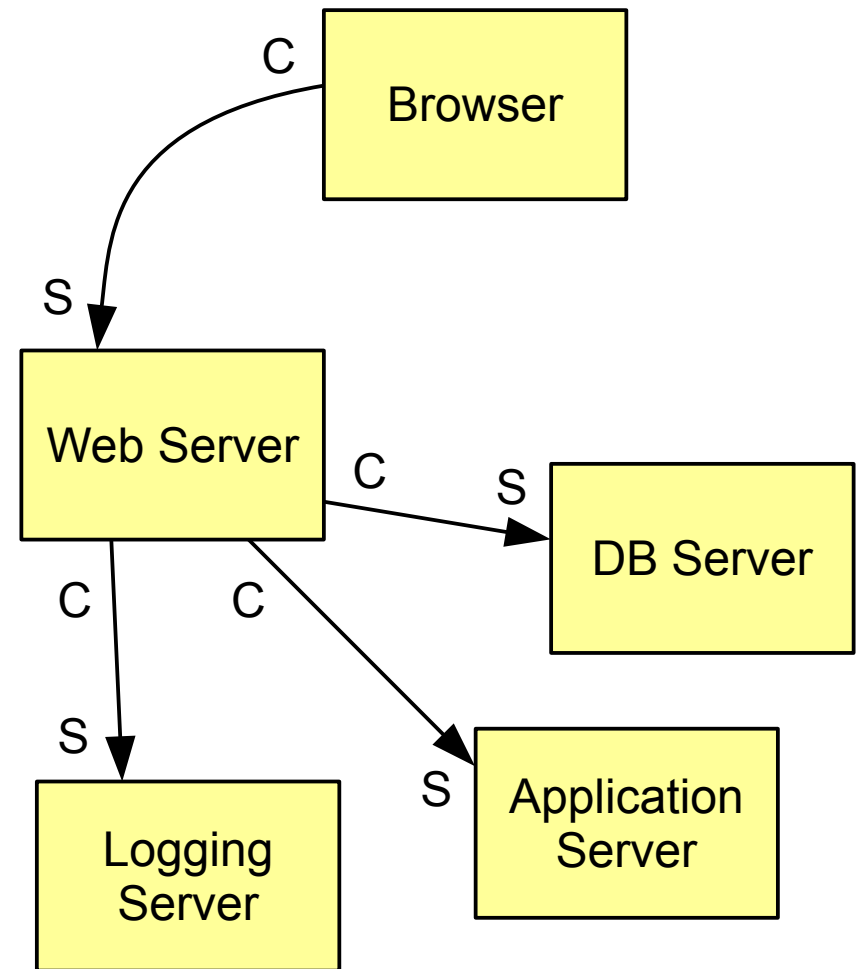
- Arten der Parallelität
 - **Bit-Level-Parallelität:** Zahl der Bits, die eine CPU-Instruktion bearbeitet (entspricht i.d.R. der Breite der Register)
 - **Instruction-Level-Parallelität:** Pipelining
 - **Daten-/Schleifen-Parallelität:** Schleifen können parallel bearbeitet werden
 - **Task-Parallelität:** Programm in unabhängige Threads aufteilen

- Verteiltes System:
 - Zusammenschluss mehrerer Computer, auch mit unterschiedlicher Architektur
 - präsentiert sich gegenüber Nutzer als ein System
 - Komponenten arbeiten autonom, auch bzgl. Scheduling und Ressourcen-Verwaltung
 - ist durch Hinzufügen weiterer Rechner skalierbar
 - kein gemeinsamer Speicher, darum Synchronisation nur durch Nachrichtenversand möglich

- Beispiele:
 - P2P-Netze (z. B. für Filesharing)
 - Verteiltes Dateisystem (z. B. IBM GPFS, Google Filesystem)
 - HPC, inkl. Cluster- und Grid-Computing, auch Seti@Home u. ä.

- Server und Client sind Prozesse, die auf derselben Maschine oder auf verschiedenen, übers Netz verbundenen Rechnern laufen
- Server bietet eine Dienstleistung
- Clients können diese nutzen, indem sie eine Anfrage an den Server schicken
- wahlweise verbindungs basiert (z. B. TCP) oder verbindungslos (z. B. UDP)
- Thin Client: nutzt überwiegend Dienste eines oder mehrerer Server

- Client / Server-Beispiele:
 - WWW (HTTP, HTTPS)
 - FTP
 - Datenbank
 - Authentifikation



- Remote Procedure Call (RPC)
 - Server erlaubt den Aufruf von Programm-Prozeduren (Funktionen) durch Clients
 - Aufruf (und ErgebnISRückgabe) durch Message Passing
 - Problem: Client und Server können verschiedene Architektur haben → andere Art der Datenablage (z. B. Little-Endian/Big-Endian, verschiedene Kodierungen für Strings)
 - **Marshalling / Serialisieren:** Packen der Prozedur-Argumente in ein universelles Format (Unmarshalling: auspacken)

- RPC-Beispiele
 - Sun RPC (heute: Open Network Computing RPC)
 - Dienste über einen Portmapper registrieren, der auf Port 111 Anfragen annimmt
 - unterstützt Authentifikation
 - serialisiert Daten mit XDR (External Data Representation) → nächste Folie
 - z. B. NFS (Network File System) nutzt Sun RPC
 - XML-RPC
 - Web Services Description Language (WSDL), SOAP (Nachfolger von XML-RPC)

```
const MAXUSERNAME = 32;    /* Länge username */
const MAXFILELEN = 65535; /* max. Dateigröße */
const MAXNAMELEN = 255;   /* Länge Dateiname */

/* Dateitypen: */
enum filekind {
    TEXT = 0,      /* ASCII-Text      */
    DATA = 1,     /* Binäre Daten   */
    EXEC = 2       /* ausführbare Datei */
};

/* Datei-Informationen: */
union filetype switch (filekind kind) {
case TEXT:
    void;
case DATA:
    /* Ersteller */
    string creator<MAXNAMELEN>;
case EXEC:
    /* Interpreter */
    string interpreter<MAXNAMELEN>;
};

/* komplette Datei: */
struct file {
    string filename<MAXNAMELEN>; /* Name      */
    filetype type;                /* Infos    */
    string owner<MAXUSERNAME>;   /* Besitzer  */
    opaque data<MAXFILELEN>;     /* Inhalt    */
};
```

Beispiel:

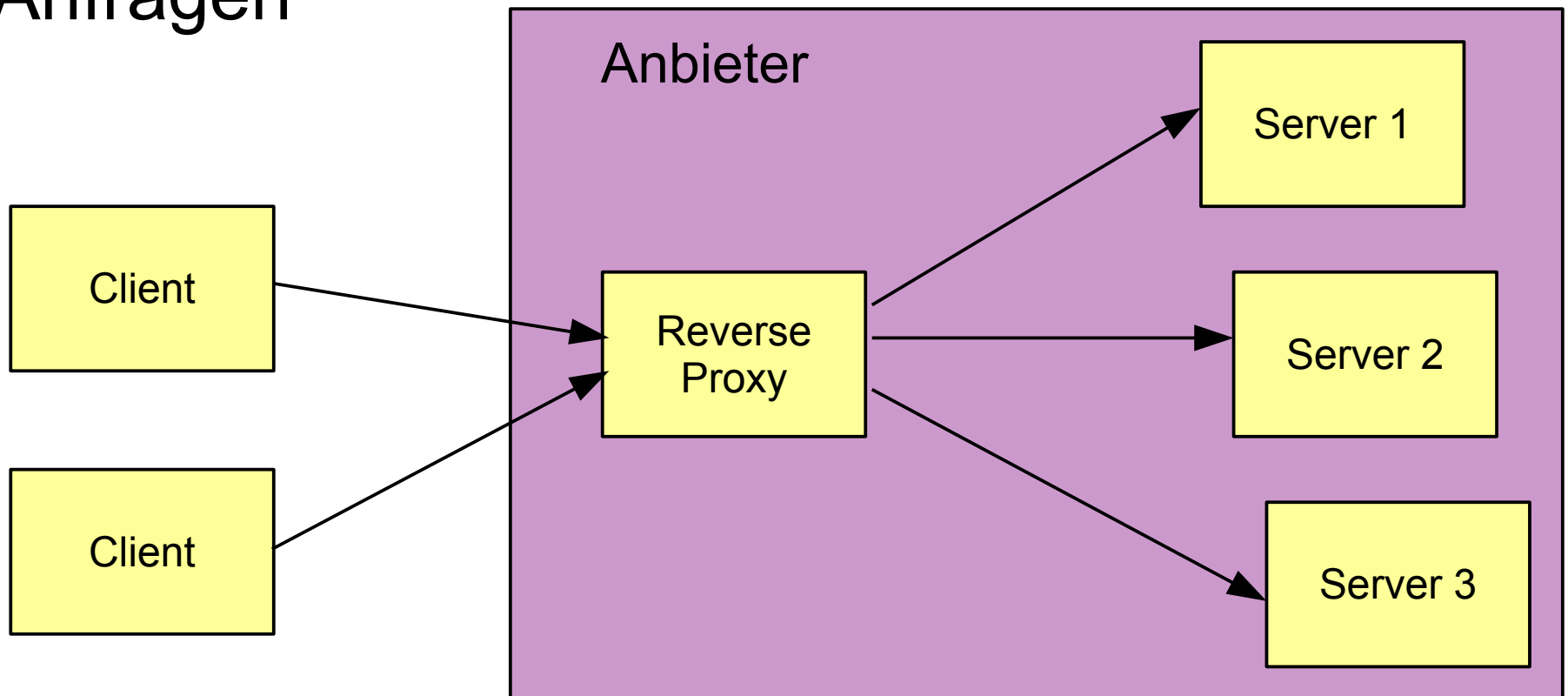
Versand einer Lisp-Datei „sillyprog“ mit Inhalt „(quit)“

Pos.	Hex	ASCII	Kommentare
0	00 00 00 09		Länge Dateiname = 9
4	73 69 6c 6c	sill	Dateiname
8	79 70 72 6f	ypro	...
12	67 00 00 00	g...	... (und 3 Füller-Bytes)
16	00 00 00 02		Dateityp 2 (EXEC)
20	00 00 00 04		Interpreter-Länge 4
24	6c 69 73 70	lisp	Interpreter-Name
28	00 00 00 04		Besitzer-Länge 4
32	6a 6f 68 6e	john	Besitzer
36	00 00 00 06		Dateigröße = 6
40	28 71 75 69	(qui	Dateiinhalt
44	74 29 00 00	t)...	... (und 2 Füller-Bytes)

Quelle: <http://tools.ietf.org/html/rfc1014>

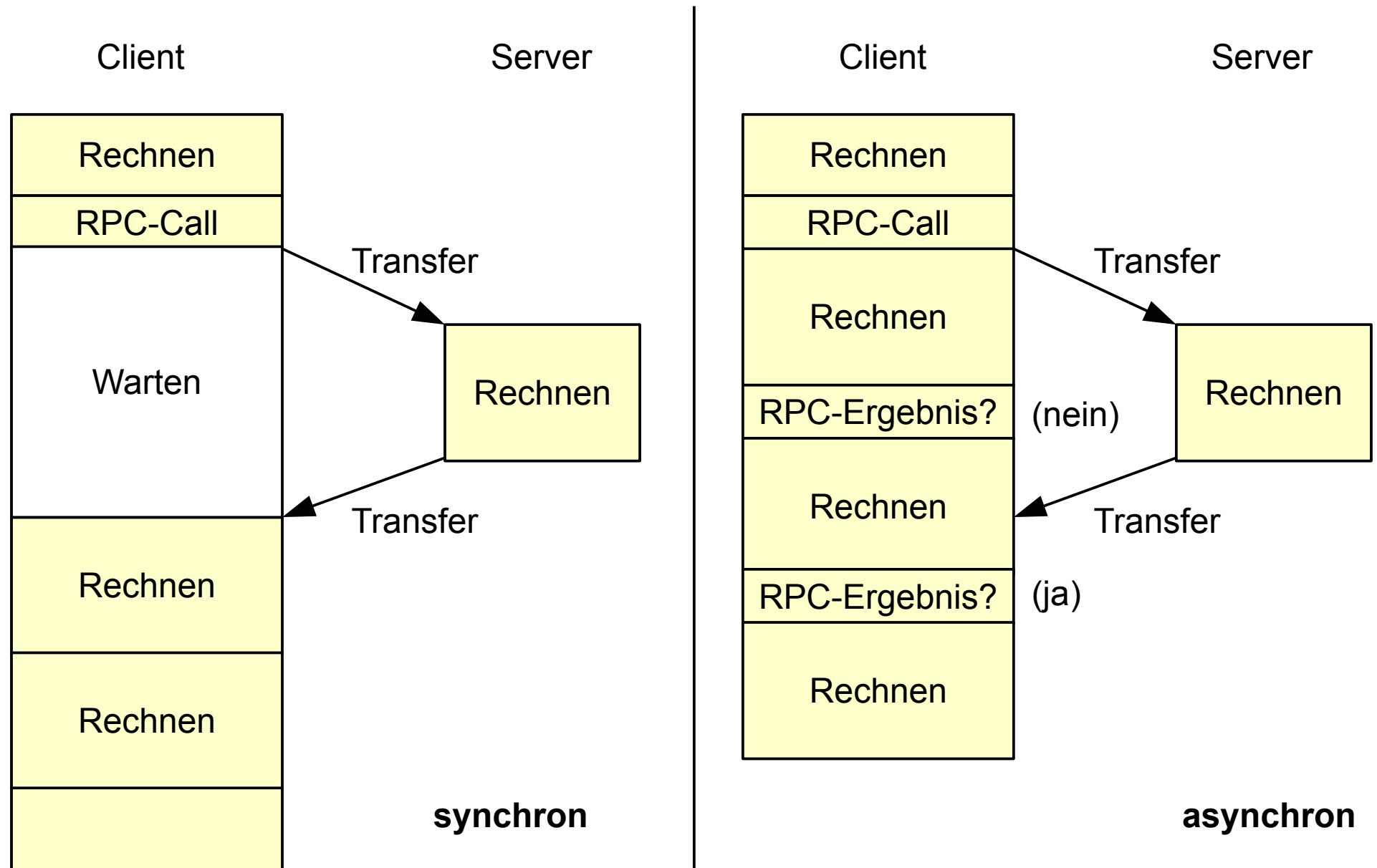
- klare Trennung der Aufgaben
- Kommunikation über definiertes Protokoll
- leichter Austausch des Servers (oder des Clients)
- wenn komplexe (aufwendige) Aufgaben von Servern erledigt werden, kann Client klein sein (→ Thin Client)
- Server kann auch ein **Reverse Proxy** sein und Anfragen an einen Server weiterleiten

- Lastverteilung
- Puffern / Cachen der Antworten auf wiederholte Anfragen



- Aufruf von RPC-Prozeduren kann wie Aufruf lokaler Funktionen arbeiten
- normales Verhalten: blockierend (wie Funktionsaufruf)
- Alternative: Asynchronous RPC
 - Zugriff auf langsame / überlastete Server
→ nicht warten, bis Ergebnis da ist
 - Transfer großer Datenmengen
 - Rückgabewert muss dann (später) separat abgefragt werden → Callback

Asynchronous RPC



- Super-Computer
 - zahlreiche (tausende) CPUs
 - gemeinsamer Speicher oder separater Speicher je CPU (→ Message Passing)
 - Plattform für High Performance Computing (HPC)
- Cluster
 - zahlreiche separate Rechner, typischerweise mit identischer Hardware
 - verbunden durch spezielles schnelles Netzwerk
 - Kooperation via Message Passing

- Message Passing mit MPI
 - MPI-Standard (Message Passing Interface)
 - wird in parallelen Programmen zur Aufteilung der Arbeit verwendet
 - typische Operationen:
 - **Send** (an einzelnen) und **Recv** (von einzelnen)
 - **Broadcast** (Nachricht an alle)
 - **Gather** (Daten von allen Prozessen erhalten; entspricht n mal Recv) und **Reduce** (z. B. Summe der Werte von allen n Prozessen erhalten)
 - **Barrier** (Schranke, die alle Prozesse erreichen müssen)

```
int main(int argc, char *argv[]) {
    char idstr[32]; char buff[BUFSIZE];
    int numprocs; int myid; int i;
    MPI_Status stat;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs); // wie viele?
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);    // wer bin ich?
    if(myid == 0) { // Master (Rank 0)
        printf("%d: We have %d processors\n", myid, numprocs);
        for (i=1; i<numprocs; i++) {
            sprintf(buff, "Hello %d! ", i);
            MPI_Send(buff, BUFSIZE, MPI_CHAR, i, TAG, MPI_COMM_WORLD);
        }
        for (i=1; i<numprocs; i++) {
            MPI_Recv(buff, BUFSIZE, MPI_CHAR, i, TAG, MPI_COMM_WORLD, &stat);
            printf("%d: %s\n", myid, buff);
        }
    } else { // Slave (Rank != 0)
        // Nachricht von Prozess 0 empfangen
        MPI_Recv(buff, BUFSIZE, MPI_CHAR, 0, TAG, MPI_COMM_WORLD, &stat);
        sprintf(idstr, "Processor %d ", myid);
        strncat(buff, idstr, BUFSIZE-1);
        strncat(buff, "active", BUFSIZE-1);
        // Nachricht an Prozess 0 senden
        MPI_Send(buff, BUFSIZE, MPI_CHAR, 0, TAG, MPI_COMM_WORLD);
    }
    MPI_Finalize(); return 0;
}
```

```
$ gcc -lmpi mpitest.c
$ mpirun -n 8 ./a.out
0: We have 8 processors
0: Hello 1! Processor 1 active
0: Hello 2! Processor 2 active
0: Hello 3! Processor 3 active
0: Hello 4! Processor 4 active
0: Hello 5! Processor 5 active
0: Hello 6! Processor 6 active
0: Hello 7! Processor 7 active
```

Quelle: http://en.wikipedia.org/wiki/Message_Passing_Interface#Example_program, leicht verändert

- Idee auch beim Grid: Lösung einer Aufgabe auf mehrere Maschinen verteilen
- Funktion: ähnlich wie Cluster, aber
 - zum Grid gehörende Rechner sind räumlich weit entfernt (keine schnelle Verbindung)
 - Rechner haben verschiedene Architektur und Hardware-Ausstattung
 - schlecht für Aufgaben, die viel Kommunikation der beteiligten Prozesse benötigen
 - schlecht, wenn parallele Anwendung voraussetzt, dass alle Nodes in etwa gleich schnell arbeiten
 - Normalfall: völlig unabhängige Prozesse

- „Shared Computing“: Idle-Zeiten (ungenutzte Ressourcen) nutzen
 - SETI@Home
 - kryptographische Verfahren brechen (→ z. B. RSA Secret-Key Challenge)
 - BOINC (Berkeley Open Infrastructure for Network Computing), Projekte: siehe <http://boincstats.com/en/stats/projectStatsInfo>
 - Auswertung riesiger Datenmengen, die nicht realistisch von Einzelpersonen oder Unternehmen verarbeitbar sind

- Beteiligung an Shared-Computing-Projekten
 - Account anlegen
 - Client installieren und laufen lassen
 - bei Interesse Statistik über eigenen Beitrag abrufen
- Grid Computing (und auch Cluster / Super-computer) eher im wissenschaftlichen Umfeld relevant
- Kommerzielle Bedeutung von parallelen Systemen erst durch Cloud Computing

- erste Umsetzung verteilter Systeme mit starker kommerzieller Bedeutung
- Anbieter u. a. Amazon, Google, Oracle, SAP, Microsoft, Dropbox ... (Liste der Top-100-Cloud-Provider: <http://talkincloud.com/tc100>)
- drei sehr unterschiedliche Angebotsvarianten:
 - Software as a Service (SaaS)
 - Platform as a Service (PaaS)
 - Infrastructure as a Service (IaaS)

- Clouds bieten *skalierbare* und *elastische* Services
 - *skalierbar*: tatsächlich genutzte Ressourcen vom Anwender konfigurierbar
 - *elastisch*: Überwachung der Nutzung; je nach Bedarf wird Rechenleistung, Plattenplatz, Netzwerk-Bandbreite automatisch angepasst
- Messbarer Ressourcenverbrauch → individuelle Gebühren entsprechen der Nutzung
- Cloud-Provider übernehmen Verwaltung und Sicherheit
- zentralisierte Cloud-Rechnerfarmen

- Cloud-Anbieter setzen **Virtualisierung** ein und profitieren dadurch von der Lastverteilung (nicht alle Instanzen arbeiten ständig mit Höchstlast)
- geringere Kosten für RZ-Betrieb können an Cloud-Anwender weitergegeben werden
- Daten werden in der Nähe der Rechner vorgehalten, die sie verwenden

- Cloud-Arten
 - **Public Cloud:** Infrastruktur gehört einem Cloud-Anbieter, Anwender greifen über das Internet darauf zu. Keine Einschränkung der Benutzergruppe
 - **Private Cloud:** Infrastruktur wird von/für ein einzelnes Unternehmen betrieben
 - **Community Cloud:** ähnlich Public Cloud, aber mit eingeschränkter Benutzergruppe, z. B. nur Krankenhäuser, öffentliche Verwaltungen etc.
 - **Hybrid Cloud:** Kooperation mehrerer Clouds (public, private, community) mit Schnittstellen

- Vorteile beim Cloud-Einsatz (für Anwender):
 - keine Anfangsinvestition für eigenes RZ
 - „pay as you go“
 - Elastizität: Leistungen können mit wachsender Benutzergruppe dynamisch mitwachsen; in Zeiten der Nicht-Nutzung keine Kosten
 - Virtualisierung lässt Benutzer (bei IaaS) mit gewohnten Umgebungen arbeiten

- Clouds erfolgreicher als Grid, Cluster etc.:
 - Fokus auf Enterprise Computing, nicht wissenschaftliches Rechnen (HPC)
 - Homogene Hardware (anders als beim Grid)
 - Standard-Hardware (anders als beim Supercomputer)
 - Alle Hardware unter einheitlicher Kontrolle des Cloud-Anbieters → Sicherheit, Fehlertoleranz, Quality of Service: leichter umsetzbar

- Software as a Service
 - Service Provider stellt Anwendungen zur Verfügung
 - Benutzer verwaltet *nicht* die zugrundeliegende Cloud-Infrastruktur oder Anwendungseinstellungen
 - Services:
 - Enterprise services: workflow management, groupware and collaborative, supply chain, communications, digital signature, customer relationship management (CRM), desktop software, financial management, geo-spatial, search.
 - Web 2.0 applications: metadata management, social networking, blogs, wiki services, portal services.

- Software as a Service
 - ungeeignet, wenn Daten nicht extern gehostet werden dürfen
 - Beispiele: GMail, Google-Suchmaschine

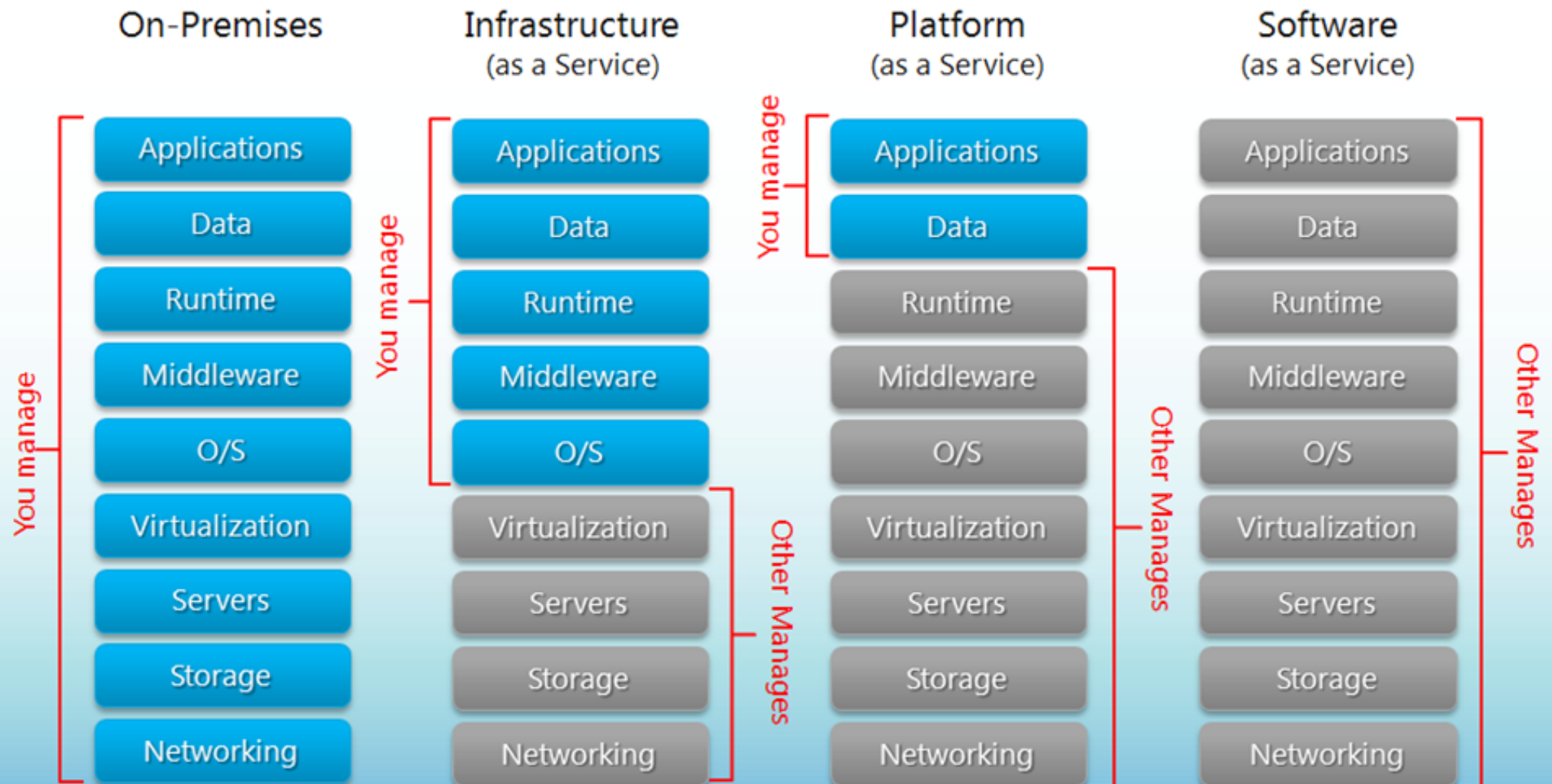
- Platform as a Service
 - Service Provider stellt Plattform zur Verfügung, auf der vom Anwender entwickelte Anwendungen laufen
 - Programmiersprachen und Tools vom Provider vorgegeben
 - Benutzer verwaltet *nicht* die zugrundeliegende Cloud-Infrastruktur (wie etwa Betriebssystem, Netzwerk-Konfiguration etc.)

- Platform as a Service
 - ungeeignet, wenn Optimierung der „Hardware“ und Software nötig ist oder spezielle (proprietäre) Programmiersprachen verwendet werden
 - Beispiele: Microsoft's Windows Azure (PaaS und IaaS), Google App Engine

- Infrastructure as a Service
 - Benutzer können beliebige Software ausführen, darunter auch (weitgehend) beliebige Betriebssysteme
 - Benutzer kontrolliert *nicht* die Cloud-Infrastruktur, *aber* die in seinen VMs laufenden Betriebssysteme, Netzwerk- und Firewall-Einstellungen, Software und deren Konfiguration

Separation of Responsibilities

Quelle: <http://blogs.technet.com/b/kevinremde/archive/2011/04/03/saas-paas-and-iaas-oh-my-quot-cloudy-april-quot-part-3.aspx>



- Gründe für den Einsatz von Public Clouds:

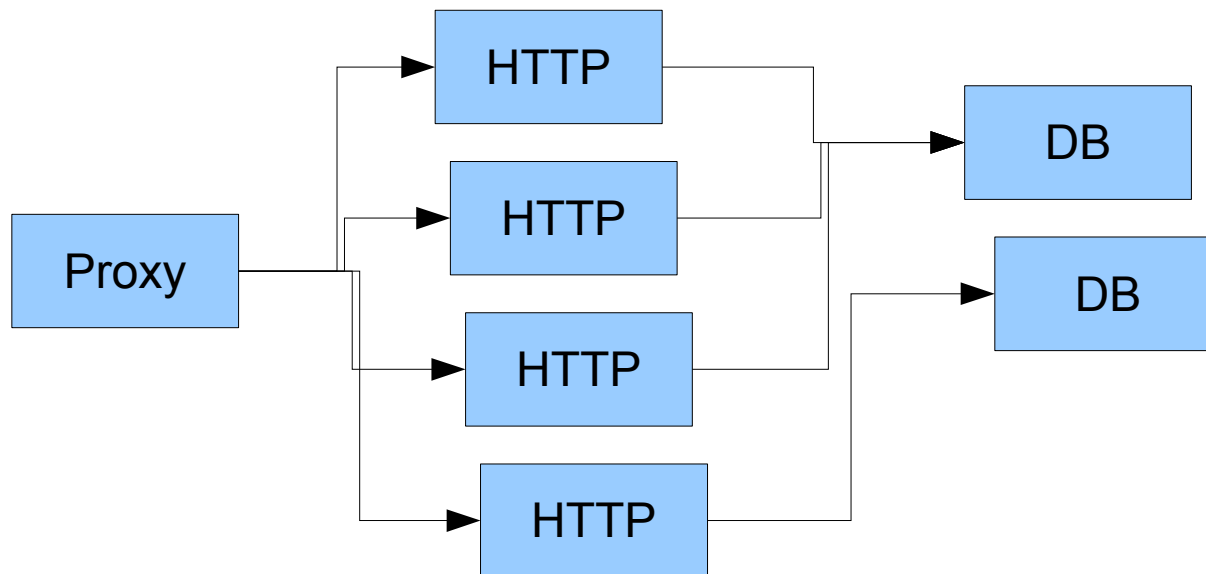
Reason	Percentage who agree
Improved system reliability and availability	50%
Pay only for what you use	50%
Hardware savings	47%
Software license saving	46%
Lower labor costs	44%
Lower maintenance costs	42%
Reduced IT support needs	40%
Ability to take advantage of the latest functionality	40%
Less pressure on internal resources	39%
Solve problems related to updating/upgrading	39%
Rapid deployment	39%
Ability to scale up resources to meet the needs	39%
Ability to focus on core competencies	38%
Take advantage of the improved economics of scale	37%
Reduced infrastructure management needs	37%
Lower energy costs	29%
Reduced space requirements	26%
Create new revenue streams	23%

- Nächster Termin:
 - technische Grundlagen der Clouds
 - ausgewählte Beispiele
 - Sicherheit in der Cloud
 - Entwickeln von Cloud-Anwendungen (nur ein grober Überblick)

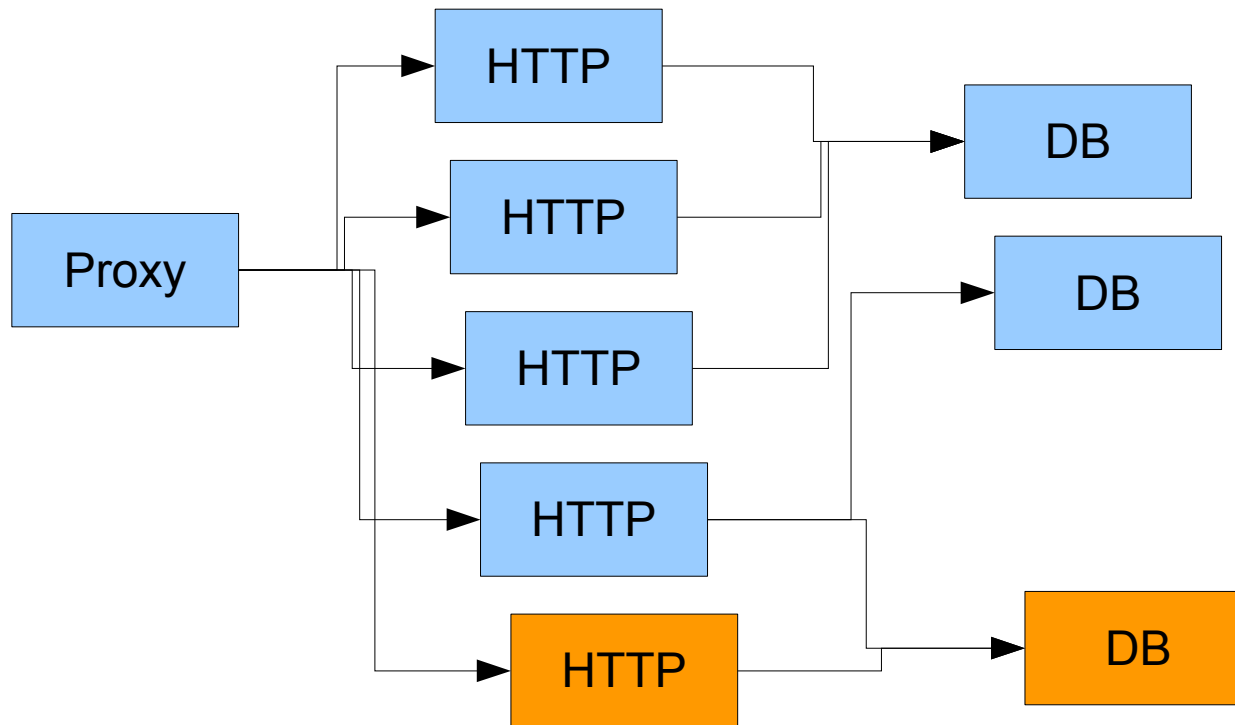
- Verteilte Konfiguration mit ZooKeeper
- Software-Entwicklung mit MapReduce
- Verteilte Dateisysteme

22.01.2015: ab hier neue Folien

- Verteiltes Repository von Konfigurationsdaten
- Beispiel: elastische HTTP/DB-Umgebung



- Elastisch: neue VM mit HTTP/DB starten



- Proxy/HTTPs müssen neue Server kennen

- ZooKeeper verwaltet eine Art Verzeichnisbaum

```

/
/config
/config/http-servers
/config/http-servers/http1
/config/http-servers/http2
/config/http-servers/http3
/config/http-servers/http4
/config/db-servers
/config/db-servers/db1
/config/db-servers/db2

```

} speichern u.a.
– IP-Adresse
– Versionsinformation

} dito

- neue Server tragen sich hier ein
- Proxy und HTTPs fragen Informationen ab

- Load-Verwaltung bemerkt wachsende Last
 - neue VM mit HTTP und DB starten
 - VM im ZooKeeper registrieren
 - Wegen *WATCH* werden Proxy und HTTPs benachrichtigt



```
/
/config
/config/http-servers
/config/http-servers/http1
/config/http-servers/http2
/config/http-servers/http3
/config/http-servers/http4
/config/http-servers/http5
/config/db-servers
/config/db-servers/db1
/config/db-servers/db2
/config/db-servers/db3
```


- Zwei DB-Server wollen gleichzeitig einen Datensatz ändern → Locking!
 - ZooKeeper bietet systemübergreifende Locks
 - Nur eine der Maschinen kann das Lock erhalten, die übrigen müssen warten
 - Bei Freigabe des Locks werden die wartenden Maschinen benachrichtigt (WATCH)

```
/
/locks
/locks/db
/locks/db/table_customers
```

- *Persistent vs. Ephemeral* Nodes
 - *persistent*: Node bleibt permanent erhalten
 - *ephemeral*: Node verschwindet, wenn sich die erzeugende Maschine abmeldet (oder ausfällt)

Basics

- Listen
 - `[]` ist leere Liste
 - `a :: list` – `a` wird vorne an Liste `list` angehängt (Ergebnis ist neue Liste)
 - `::` heißt **Prepend**-Operator (vgl. Append)

- Map-Operation

- $\text{map}(f, []) = []$
- $\text{map}(f, a :: \text{list}) = f(a) :: \text{map}(f, \text{list})$
- z. B.: $f = (x \rightarrow 3 * x)$, $\text{map}(f, [1, 2, 3]) = [3, 6, 9]$

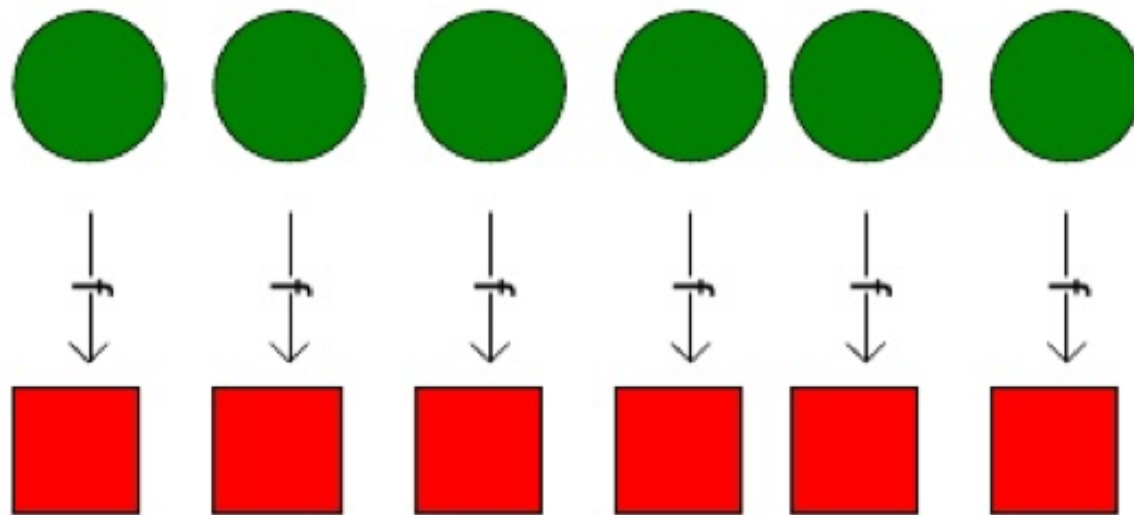


Bild: <http://de.slideshare.net/tugrulh/presentations>
(Google-Präsentation)

- Fold-Operation
 - wendet eine Funktion $f(x, y)$ mit zwei Argumenten der Reihe nach auf alle Elemente der Liste an

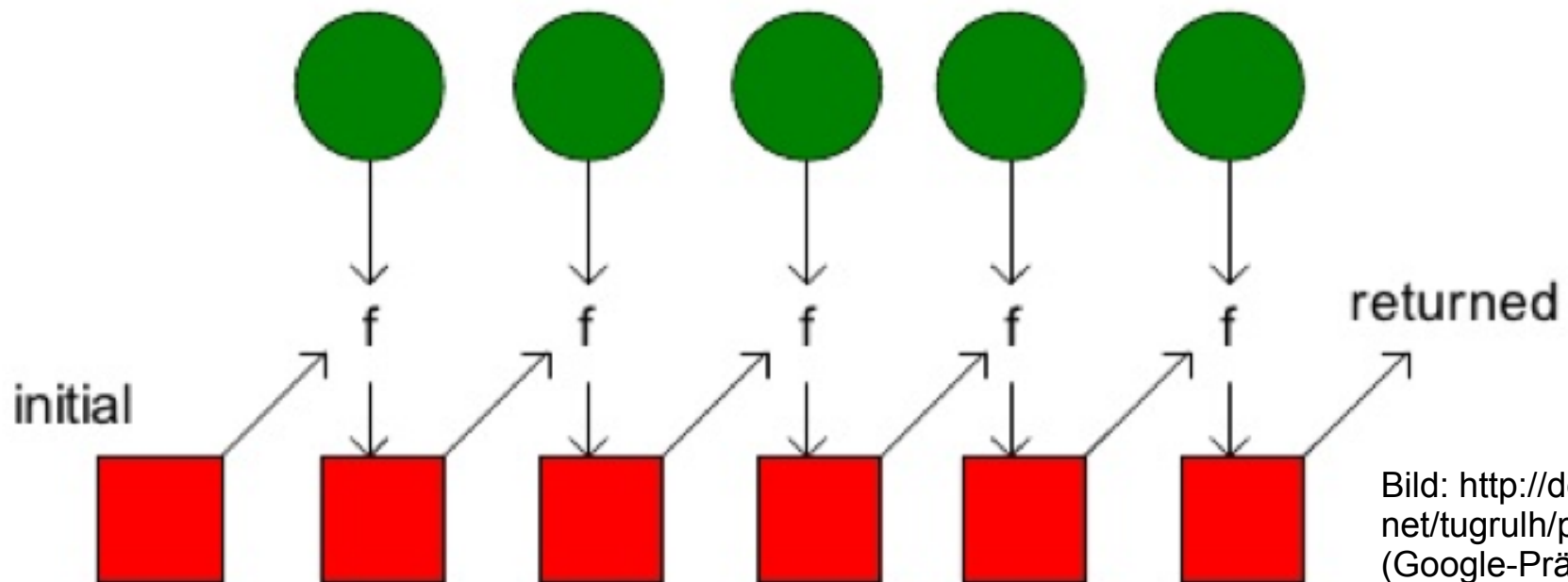


Bild: <http://de.slideshare.net/tugrulh/presentations>
(Google-Präsentation)

- Fold-Operation (Forts.)
 - braucht Startwert
 - $\text{foldl}(\text{add}, 0, [1, 2, 3]) = 3 + (2 + (1 + 0)) = 6$
(Summe aller Listenelemente)
 - $\text{foldl}(\text{mult}, 1, [3, 4, 5]) = 5 * (4 * (3 * 1)) = 60$ (Produkt aller Listenelemente)
 - `foldr`: arbeitet sich von rechts nach links durch

- Implementierung von `foldl` und `foldr`
 - `foldl(f, a, []) = a`
 - `foldl(f, a, x::rest)`
`= foldl(f, f(x,a), rest)`
 - `foldr(f, a, []) = a`
 - `foldr(f, a, x::rest)`
`= f(x, foldr(f, a, rest))`
 - `a` heißt „Akkumulator“

- Dabei muss `foldl` nicht zwingend ein einzelnes Element als Ergebnis erzeugen
→ kann auch Liste erzeugen
- Verwende „Prepend“-Operator `::` als Funktion

- Beispiel: Reverse-Funktion für Listen
 - $f(x,a) = x :: a$ (a ist Liste)
 - `reverse (list) = foldl (f, [], list)`
 - `reverse ([1,2,3])`
= `foldl (f, [], [1,2,3])`
= `foldl (f, f(1, []), [2,3])`
= `foldl (f, [1], [2,3])`
= `foldl (f, f(2, [1]), [3])`
= `foldl (f, [2,1], [3])`
= `foldl (f, f(3, [2,1]), [])`
= `foldl (f, [3,2,1], [])` = `[3,2,1]`

- Beobachtung: `map` verarbeitet alle Listenelement unabhängig (während `foldl` und `foldr` sequentiell arbeiten)
- Darum lässt sich `map` gut parallelisieren.

- Verarbeitung großer Datenmengen, z. B. Terabytes
- parallelisieren: auf hunderte oder tausende CPUs verteilen
- MapReduce erlaubt
 - automatische Parallelisierung und Verteilung
 - Fehler-Toleranz
 - Statusanzeige, Monitoring

- übernimmt Konzepte der funktionalen Programmierung
 - zu verarbeitende Daten auf mehrere Maschinen aufteilen
 - jede Maschine führt auf den (ihr zugeteilten) Daten eine **Map**-Funktion aus
 - nach Abschluss aller Berechnungen werden die Ergebnisse gesammelt
 - Dann kombiniert eine **Reduce**-Funktion (wie etwa fold) alle Ergebnisse

- Beispiel: gemeinsame Freunde (Facebook o. ä.)
 - Ausgangssituation: Für jeden Teilnehmer X eine Liste seiner Freunde (key $\rightarrow$ value)

A $\rightarrow$ B C D	D $\rightarrow$ A B C E
B $\rightarrow$ A C D E	E $\rightarrow$ B C D
C $\rightarrow$ A B D E	

- Jede Zeile an eine Maschine schicken und dort eine **Map-Funktion** starten

A $\rightarrow$ B C D	$\longrightarrow$	(A B) $\rightarrow$ B C D
		(A C) $\rightarrow$ B C D
		(A D) $\rightarrow$ B C D

Quelle für Beispiel: <http://stevekrenzel.com/finding-friends-with-mapreduce>

MapReduce (4) – Beispiel

- Nach Abschluss der Berechnungen alle Ergebnisse zusammenfassen, nach *key* gruppiert

(A B) → (A C D E) (B C D)
 (A C) → (A B D E) (B C D)
 (A D) → (A B C E) (B C D)
 (B C) → (A B D E) (A C D E)
 (B D) → (A B C E) (A C D E)

(B E) → (A C D E) (B C D)
 (C D) → (A B C E) (A B D E)
 (C E) → (A B D E) (B C D)
 (D E) → (A B C E) (B C D)

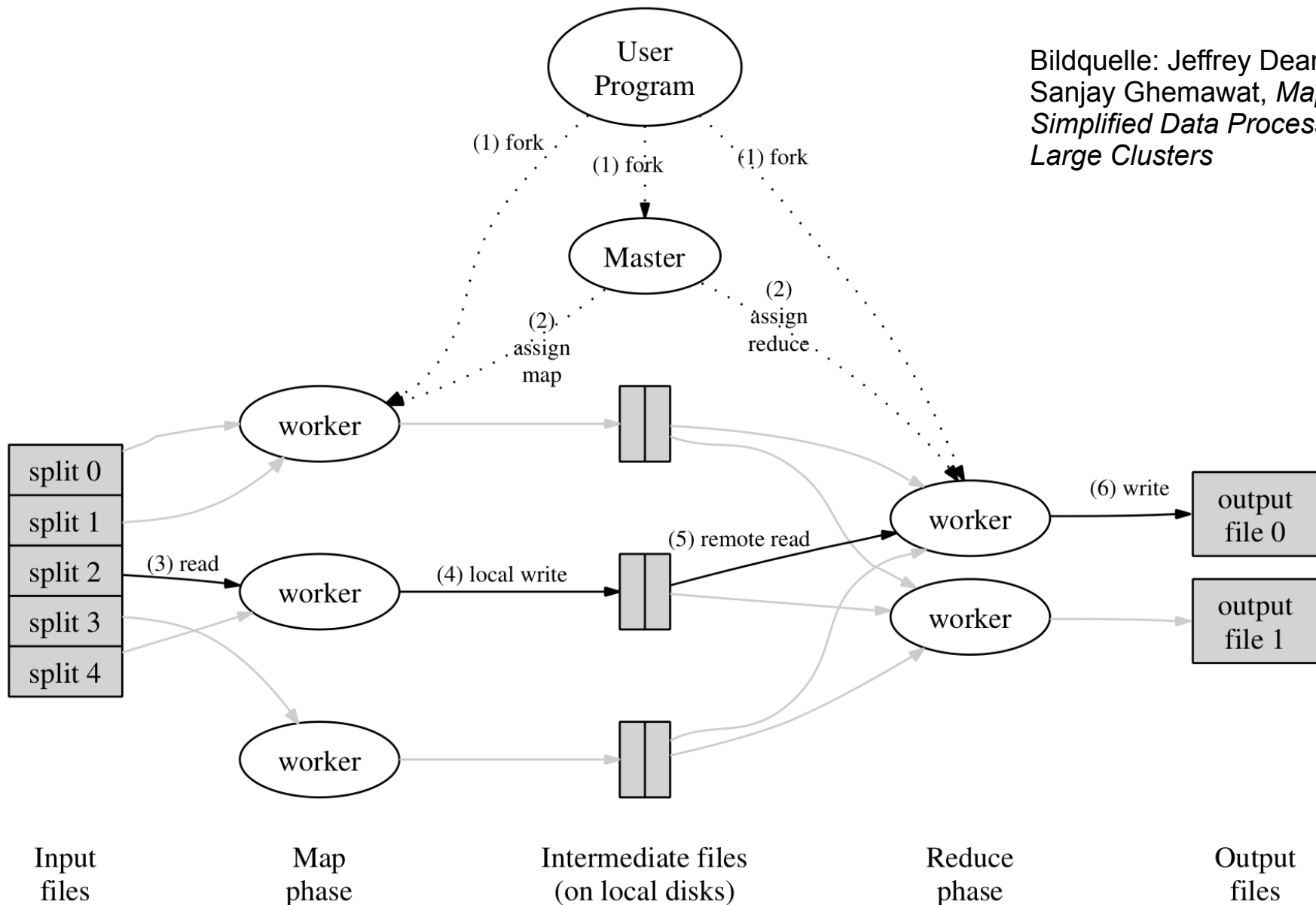
- Abschluss: für jeden Key die **Reduce**-Funktion aufrufen (hier: Schnittmenge berechnen)

(A B) → (C D)
 (A C) → (B D)
 (A D) → (B C)
 (B C) → (A D E)
 (B D) → (A C E)

(B E) → (C D)
 (C D) → (A B E)
 (C E) → (B D)
 (D E) → (B C)

MapReduce (5)

Bildquelle: Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*



- Gemeinsames (globales) Dateisystem für alle Mitglieder der Cloud
- Einfacher NFS-Server, dessen Shares an alle Clients exportiert werden?
 - Daten sind immer remote
 - langsame Übertragung aller Daten
 - NFS-Server ist ein Bottleneck, weil etliche Rechner in der Cloud gleichzeitig darauf zugreifen
 - skaliert nicht

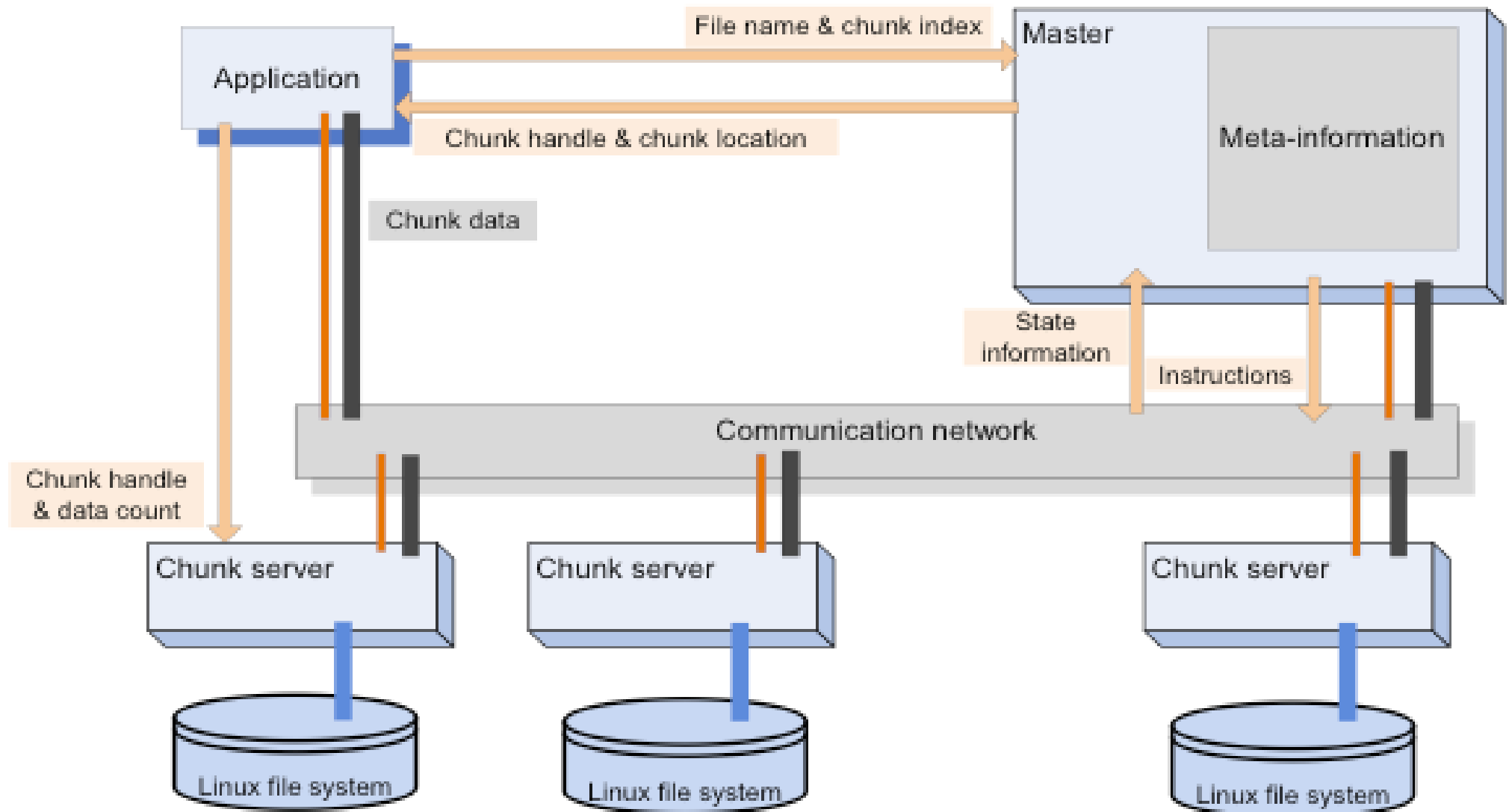
- Statt zentralem Fileserver: verteilten Server nutzen
- mehrere Server im Netz bilden gemeinsam einen Fileserver
 - Ausfallsicherheit durch Duplikation (Redundanz)
 - Datenlokalität: Daten dort speichern, wo sie benötigt werden

- Verteiltes Dateisystem
- ab 2000; zunächst für HPC-Cluster entworfen
- parallele Zugriffe mehrerer Rechner
- Locking schützt Schreibvorgänge
 - Locking verwendet Tokens mit Ablaufzeit
 - keine explizite Rückgabe des Locks nötig
 - reduzierter Traffic
- File Placement Optimizer (FPO)
 - Daten liegen auf Platten in der Nähe des Servers
 - für MapReduce-Berechnungen vorbereitet

- Ausgangssituation bei Google
 - einige Millionen Dateien mit jeweils > 100 MByte
 - Dateien vom Typ „write-once“, mit Append
 - Lese-Zugriff auf diese Dateien über Streaming
 - bei Zugriff: hoher Durchsatz wichtiger als niedrige Latenz

- Design-Entscheidungen für GFS
 - Dateien in 64-MByte-Chunks aufteilen
→ Performance-Optimierung für große Dateien
 - Atomare Append-Operation; mehrere Prozesse können an gleiche Datei Daten anhängen
 - Replikation: jeder Chunk auf ≥ 3 Chunk-Servern
 - ein Master speichert Metadaten, koordiniert Zugriffe
→ soll Konsistenz garantieren (aber: Bottleneck)

Google-Filesystem (3)

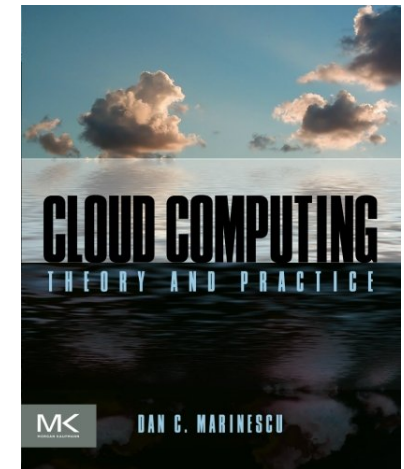


Bildquelle: Marinescu, *Cloud Computing, Theory and Practice*

- GFS Chunks
 - **Chunk**-Größe 64 MByte, aber zerlegt in **Blöcke** der Größe 64 KByte
 - Jeder 64-KByte-Block hat 32-Bit-Prüfsumme
- Locking erfolgt auch Chunk-Ebene

- Schreiben eines Chunks
 - Master wählt einen der Chunk-Server als **Primary** aus, der schreib-berechtigt ist
 - Client (der schreiben will) bittet Master um Schreibberechtigung
 - Master schickt Client Information über Primary und alle Secondary Chunk Servers
 - Client schickt Daten an alle (!) Chunk-Server
 - Alle Chunk-Server schicken ACK an Client
 - Client schickt Write-Request an Primary
 - Primary schickt Write-Request an alle Seconarys

- Dan C. Marinescu: *Cloud Computing – Theory and Practice*, Morgan Kaufmann 2013
- Jimmy Lin and Chris Dyer: Data Intensive Text Processing with MapReduce, 2010,
<http://lintool.github.io/MapReduceAlgorithms/MapReduce-book-final.pdf>



ZUSAMMENFASSUNG und letzte Fragen

- Information vs. Daten
- Bits, Bytes, dual/oktal/hex
- ASCII, Unicode (UTF-8, UTF-16)
- Digitalisierung:
 - rastern (diskretisieren, sampeln) und
 - quantisieren (Wertebereich einschränken)
- Kompression (verlustfrei/verlustbehaftet)
- Prüfsummen, Fehlerkorrektur
- Mark-up: HTML/CSS, LaTeX, Wiki, XML

- Mark-up: HTML/CSS, LaTeX, Wiki,
- XML, XSLT, DTD, XHTML, DocBook, Open Document Format (XML)
- Rastergrafiken (JPG vs. PNG), Vektorgrafiken
- PostScript, PDF
- MPEG, Frames
- Kommunikation
 - Client / Server; HTTP, FTP, SMTP

- Datenträgeraustauschformat (DTaus)
- Applikationsformate (CAD, GIS, DTP)
- Archive und Software-Pakete
 - zip, tar, gz, tar.gz (plattform-übergreifend?)
 - Linux: RPM, Debian;
 - Pakete mit Metadaten
 - Abhängigkeiten, Konflikte
 - Repositories: Abh. auflösen, Upgrade
 - Windows: MSI, OS X: DMG mit *.app-Ordnern

- Exkurs: LaTeX, BibTeX
- Statistik: Programmiersprache R
- Numerik: GNU Octave
- Computer-Algebra-Systeme, Wolfram Alpha
- GIS: Mark-up-Sprachen, Nielsen-Gebiete
- Versionsverwaltung, Mercurial (hg), Klonen, Merge-Operation

- „Gebrauchstauglichkeit“
- Nutzungskontext(e), Zielgruppe(n)
- effektiv, effizient, zufrieden stellend
- Kriterien
 - Aufgabenangemessenheit
 - Selbstbeschreibungsfähigkeit
 - Steuerbarkeit
 - Erwartungskonformität
 - Fehlertoleranz
 - Individualisierbarkeit
 - Lernförderlichkeit

- Beispiele:
 - Microsoft Ribbons
 - Seriennummern-Eingabe
 - Secure Shell (ssh, scp; Optionen -p, -P)
- Kriterien auch auf Webdesign anwendbar
- Arbeitsschutz

- Universalrechner, von-Neumann vs. Harvard
- ISA (Instruction Set Architecture)
 - Maschinenbefehle, Register
 - Adressierungsarten, Interruptbehandlung
 - 1-/2-/3-Adress-Maschinen bzw. -Befehle
 - Spezialregister (IP, SP, Status)
- Load, Store, Push, Pop, arithm. Operationen
→ RISC, CISC

- Pipelining
 - 5-stufige RISC-Pipeline (Fetch, Decode, Execute, Memory Access, Write-back)
 - 6-stufige CISC-Pipeline (Fetch, Decode, Calculate Operands, Fetch Operands, Execute, Write-back)
 - Pipeline-Hemmnisse
 - strukturell (Speicherzugriff bei **M**, **F**)
 - Datenabhängigkeiten (RAW-Konflikt)
 - ablauf-bedingt (bedingter Sprung; Sprungvorhersage)

- Superskalare Architekturen
 - mehrere Execute-Einheiten, nach Aufgaben getrennt (z. B. FPU, Int, MUL)
 - abhängige vs. unabhängige Pipelines
 - Datenabhängigkeiten (RAW, WAR, WAW; RAR)
 - Abhängigkeitsgraph (keine RARs, keine transitiven Abhängigkeiten)
 - Reorder-Buffer
 - Platz im Buffer, Anzahl der Ausführ-Einheiten (pro Kategorie), Anzahl der pro Takt zuteilbaren (issue) und bestätigten (commit) Befehle

- Parallele Systeme
- Programm parallelisieren, Speed-up
→ Amdahls Gesetz: $S(n) \rightarrow 1 / \alpha$
- Arten der Parallelität (Bit-Level, Instruction-Level, Daten/Schleifen-Parall., Task-Parall.)
- Verteilte Systeme
- Client / Server, RPC, Reverse Proxy, Asynchronous RPC
- Cluster, Super Computer (HPC, MPI), Grid

- Cloud Computing
 - skalierbar, elastisch, abrechenbar, zentralisiert
 - Virtualisierung
 - public / private / community / hybrid cloud
 - Cloud-Vorteile: pay as you go, keine Investitionen
 - Varianten:
 - Software as a Service (SaaS),
 - Platform as a Service (PaaS),
 - Infrastructure as a Service (IaaS)

- Technische Umsetzung
 - Verteilte Konfiguration mit ZooKeeper
 - Map und Fold (funktionale Programmierung)
 - Software-Entwicklung mit MapReduce
 - Verteilte Dateisysteme (GPFS, GoogleFS statt NFS)