Computersysteme

Skript zur Vorlesung

Computersysteme
WS 2018/2019

Reinhard Koch

Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Vorwort

Dieses Skript, welches ursprünglich durch Prof. Gerhard Sommer entworfen und von mir für die Vorlesung Computersysteme angepasst wurde, soll einen Leitfaden zum Vorlesungsstoff bilden und Hintergrundinformationen für das Verständnis des Vorlesungsstoffes bereit stellen. Es ersetzt auf keinen Fall den Vorlesungsbesuch. Als weitere Informationsquelle dient das Skript, welches Prof. Schimmler in den vergangenen Jahren für diese Vorlesung herausgegeben hat. Allerdings ist das Schimmler-Skript knapper gehalten und kann eher für die Übungsaufgaben herangezogen werden. Beide Skripte ergänzen sich daher und werden parallel bereit gestellt.

Reinhard Koch, Oktober 2018

Inhaltsverzeichnis

1 Einführung in Computersysteme

In diesem Kapitel wird zunächst eine kurze Einführung in das Wesen der Informatik gegeben, danach betrachten wir die Geschichte des maschinellen Rechnens und führen in die Zahlendarstellung in Computern ein. Dieses Kapitel wird ergänzt durch das Kapitel 1 des Skriptes von Prof. Schimmler.

1.1 Was ist Informatik

Die Informatik ist eine im Fächerkanon wissenschaftlicher Hochschulen sehr junge Wissenschaft.

An deutschen Universitäten begann die Ausbildung von Informatikern Ende der 60er Jahre. Der Einsatz von Informatikern in der Praxis und die Anwendungen der Informatik im täglichen Leben und im Beruf haben in dieser kurzen Zeit zu einer revolutionären Dynamik geführt. So stehen mit den heutigen PCs Computer mit Leistungen zur Verfügung, die weit diejenigen von Großrechnern in den 70er Jahren übertreffen.

Der Begriff Informatik

Es ist schwer zu definieren, was das Wesen der Informatik ausmacht, im Gegensatz zu klassischen Wissenschaften (Physik, Mathematik, etc.). Der Begriff *Informatik* wurde 1968 durch den damaligen Wirtschaftsminister Stoltenberg eingeführt und stellt ein Kunstwort dar, das sich aus den Begriffen *Information* und *Mathematik* zusammensetzt.

Definition: (Informatik)

Informatik ist die Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung, Speicherung und Übermittlung von Informationen.

Informatik als Wissenschaft

Die Informatik beschäftigt sich mit

- Struktur, Wirkungsweise, Anwendung und Konstruktionsprinzipien von Informationsverarbeitungssystemen
- Strukturen, Eigenschaften und Beschreibungsmöglichkeiten von Information und Informationsverarbeitungsprogrammen
- Möglichkeiten der Strukturierung, Formalisierung und Mathematisierung von Anwendungsgebieten
- Modellbildung und Simulation.

Ist die Informatik eine Wissenschaft?

Wesentliches Moment der Definition einer Wissenschaft ist das Verfügen über eine theoretische Basis, aus der neue Erkenntnisse abgeleitet werden. Dies ist in der Informatik vielfältig unter Beweis gestellt. Als Beispiel sei die Entwicklung von Programmiersprachen und Rechnerarchitekturen aus theoretischen Erkenntnissen über Prozesse und Systeme der Informationsverarbeitung genannt.

Informatik ist eine Strukturwissenschaft

Carl Friedrich von Weizsäcker gibt in seinem Buch "Die Einheit der Natur" eine Unterteilung der Wissenschaften in

- Naturwissenschaften (Physik, Biologie)
- Ingenieurwissenschaften (Nachrichtentechnik)
- Geisteswissenschaften (Philosophie)
- Sozialwissenschaften (Soziologie, Politologie)
- Strukturwissenschaften (Mathematik, Informatik)

an und bezeichnet die Informatik als *Strukturwissenschaft*. Zitat:

> "... sie studiert Strukturen in abstracto, unabhängig davon, welche Dinge diese Strukturen haben, ja ob es überhaupt solche Dinge gibt."

Wurzeln der Informatik:

- Mathematik, Physik
- Nachrichtentechnik
- (Kybernetik: Wissenschaft von den Beziehungen zwischen dem Verhalten von Systemen, d.h. ihren Erscheinungsformen, und ihrer Struktur)

Es hat sich eine Entwicklung der Informatik zu einer selbständigen Wissenschaft mit vier wesentlichen Elementen vollzogen:

1. die formale Spezifikation von Systemen der Informationsverarbeitung
2. die Darstellung von Systemen durch Datenstrukturen und Algorithmen
3. die automatisierte Durchführung von Transformationen von Daten, um Information zu gewinnen
4. die Synthese und Analyse von Hardware- und Softwareinstrumenten als Basis der Implementierung von Algorithmen

Ambivalenz der Informatik:

1. Die Informatik erfordert

   das analytische Herangehen der Naturwissenschaften

   und

   die formalisierenden Methoden der Mathematik

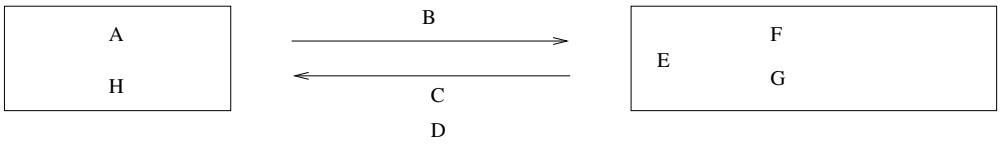
   zur Lösung eines Problems der Informationsverarbeitung.

   - Zuordnung eines konkreten Problems zu einer Problemklasse
   - Frage: Was ist Information, wie ist sie zu beschreiben und wie kann man sie aus Daten erhalten?
   - Frage: Wie sind informationsverarbeitende Systeme zu entwerfen?

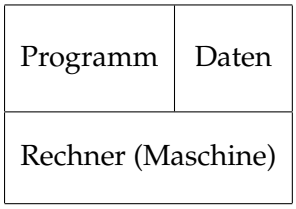
   Beispiel: Computer Vision $\sim$ Maschinensehen

   1. Wie ist die visuell wahrnehmbare Welt zu repräsentieren/strukturieren, um visuelle Wahrnehmung technisch zu realisieren?
   2. Wie sind Struktur und Dynamik von Systemen zu verstehen, um visuelle Wahrnehmung realisieren zu können?

•



- Rechenprozeß erfordert: $\longrightarrow$ Informatik beschäftigt sich mit
  - Rechnerstrukturen
  - Programmstrukturen
  - Datenstrukturen



2. Die Informatik lebt durch das konstruktive Element der Ingenieurwissenschaften: Ohne die Möglichkeit der Anwendung gäbe es die Informatik nicht!

   - Tätigkeiten eines Ingenieures [nach ENCYCLOPAEDIA BRITANNICA]:

     Die schöpferische Anwendung wissenschaftlicher Prinzipien auf den Entwurf und die Entwicklung von Strukturen, Maschinen, [...] im Hinblick auf eine gewünschte Funktion, Wirtschaftlichkeit und Sicherheit von Leben und Eigentum.

   - Wie lassen sich abstrakte Beschreibungen von Systemen informationsverarbeitender Prozesse auf konkrete Realisierungen abbilden?

     - Hardware $\sim$ "Rechner" $\rightarrow$ Rechnerarchitekturen, Computer Engineering

     - Software $\sim$ "Programme" $\rightarrow$ Software Engineering

     Auswahl aus unendlicher Zahl von Möglichkeiten unter Randbedingungen (Zwängen)

   - enge Benachbarung/Überschneidung von Informatik und Elektrotechnik
   - Gegenstand der Informatik ist vor allem anderen:
     - Analyse und (Re-)Organisation der Arbeit mit Hilfe informationstechnischer Mittel, ihre maschinelle Unterstützung oder ihre Ersetzung durch Maschinen und
     - die Entwicklung der Informationstechnik zu diesen Zwecken, insbesondere die Entwicklung des methodisch begründeten Entwurfes von Soft- und Hardware und der Integration informationstechnischer Komponenten zu Systemen.

Drei Paradigmen der Informatik:

1. Paradigma: *Theorie* (verwurzelt in Mathematik)
   Ziel: Schaffung einer kohärenten und validen Theorie nach 4 Schritten
   a) Definition: charakterisiere die zu untersuchenden Objekte
   b) Theorem: stelle Hypothese über mögliche Beziehungen unter diesen Objekten auf
   c) Beweis: untersuche, ob diese Beziehungen wahr sind
   d) Abstraktion: interpretiere die Ergebnisse
2. Paradigma: *Modellbildung* (verwurzelt in experimentellen Naturwissenschaften)
   Ziel: Abstraktion abgeleitet aus der Beobachtung von Phänomenen nach 4 Schritten
   a) bilde eine Hypothese
   b) konstruiere ein Modell und leite eine Vorhersage ab
   c) entwerfe ein Experiment und gewinne Daten
   d) analysiere die Ergebnisse
3. Paradigma: *Entwurf* (verwurzelt in Ingenieurswissenschaften)
   Ziel: Konstruktion eines Systems (Gerätes) nach 4 Schritten
   a) benenne die Erfordernisse
   b) gebe eine Spezifikation an
   c) entwerfe und implementiere das System
   d) teste das System

Folgerungen aus dem konstruktiven Element der Informatik:

- Die Informatik ist in bedeutendem Maße eine Ingenieurwissenschaft
- Die Informatik vermag durch ihre Ergebnisse die menschliche Gesellschaft enorm zu beeinflussen
  - Die technologische Entwicklung schreitet der gesellschaftlichen Entwicklung voran
  - Gefahr starker gesellschaftlicher Verwerfungen
    Muß das Machbare auch zu jeder Zeit realisiert werden?

- Der technologische Fortschritt treibt die interne Entwicklung der Informatik stark voran
  - Was gestern uninteressant war, weil technisch nicht realisierbar, muß heute entwickelt werden, weil morgen realisierbar
  - Softwarekrise: Die Entwicklung der Programmierkonzepte hinkt der Entwicklung der Hardware nach
- Die technische Realisierbarkeit bedeutet nicht die Durchsetzbarkeit auf dem Markt

---

Beispiel: Parallelrechner

Diktat des Faktischen: preisgünstige Universalrechner werden immer billiger, schneller, weiter verbreitet, einfacher zu bedienen/programmieren.
Die Programmierparadigmen der Parallelrechner sind noch nicht weit entwickelt: ihre Programmierung ist langwierig, teuer und kompliziert.

Gebietsgliederung der Informatik:

1. *Technische Informatik*
   - stellt die erforderlichen Gerätschaften (Hardware) bereit
   - beschäftigt sich mit der Konstruktion von Schaltwerken/Rechnern, Speicherchips, Prozessoren, Peripheriegeräten
   - sehr eng mit der Elektrotechnik verbunden
   - Hardware muß potentielle Anwendungen im Auge haben
   - gekennzeichnet durch Nutzung hardwarenaher Programmierung und Simulation der Hardware
2. *Praktische Informatik*
   - stellt im weitesten Sinne die Programme (Software) bereit
   - schlägt Brücke zwischen Hardware und Anwendungssoftware
   - beschäftigt sich mit der strukturierten Erstellung von Softwaresystemen: Informations-/Kommunikationssysteme, Betriebssysteme, Übersetzer
   - Künstliche Intelligenz: Wissensverarbeitung
   - gekennzeichnet durch hardwareferne Programmierung
3. *Theoretische Informatik*

- stellt im weitesten Sinne die abstrakten Strukturen bereit
- hat besonders enge Beziehung zur Algebra und Logik
- beschäftigt sich mit: Automatentheorie, formalen Sprachen, Komplexität von Algorithmen, Berechenbarkeit von Problemen

4. *Angewandte Informatik*

   - beschäftigt sich mit dem Einsatz von Rechnern in unterschiedlichsten Anwendungsgebieten (z.B. Textverarbeitungssysteme, Sprachverarbeitung, Bildverarbeitung, Handschriftenerkennung)
   - "Bindestrich-Informatik"
     Medizinische Informatik, Wirtschaftsinformatik, Rechtsinformatik

1.2 Die Geschichte des maschinellen Rechnens

Die Informatik ist die Wissenschaft, die den Menschen bei der Ausführung *geistiger* und *körperlicher* Tätigkeiten unterstützt oder von ihnen befreit.
Hierdurch ist ein inhärenter sozialer Sprengstoff im Wesen der Informatik begründet.

- Geistige Tätigkeiten:
  - Rechnen mit Zahlen und Symbolen
  - Bewerten von Daten und Aussagen
  - Verstehen von Text
  - Übersetzen von Text
  - Verstehen von natürlicher Sprache, Erzeugen natürlicher Sprache
  - Verstehen von Bildern, Maschinensehen
- Körperliche Tätigkeiten:
  - Regelung und Steuerung von Prozessen mittels Computer ("eingebettete Systeme")
  - Roboter
    - Industrierobotor: blind, "dumm", d.h. operieren nur unter vordefinierten Bedingungen
    - autonome Systeme: visuelle, sensorische Wahrnehmung, lernfähig, unter weniger eingeschränkten Bedingungen einsetzbar

Die Informatik hat ihre Wurzeln im Bedürfnis, geistige Tätigkeiten zu mechanisieren.

1.2.1 Das Rechnen mit Ziffern

Ziffern stellen Zahlzeichen eines Zahlwortes dar, z.B. "neun": IX oder 9

Die römischen Zahlen wurden nach einem *Additionssystem* gebildet,
z.B. 1996 $\sim$ MDCCCCLXXXVI oder MCMXCVI.
Die Zahlen wurden mit Zahlsteinchen, den "calculi" gelegt, hiervon stammt der Begriff "Kalkül" ab.

Die Ziffern von Eins bis Neun sowie die Null als Zeichen für leere Stellen wurden durch die Inder (ca. 800 n.Chr.) eingeführt.

Sie "erfanden" das Dezimalsystem, ein *Stellenwertsystem* zur Basis 10. Erst diese Einführung des Stellenwertsystemes der arabischen Ziffern erlaubte die Mechanisierung des

Rechnens mit Ziffern.
Eine Zahl im Dezimalsystem besitzt die Darstellung

$$z_{10} = \sum_{i=0}^{n-1} \alpha_i 10^i,\ \alpha_i \in \{0, 1, \ldots, 9\}.$$

Z.B. $2018_{10} = 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 8 \cdot 10^0$.
Auch heute existieren im täglichen Leben noch weitere Zahlensysteme, z.B:
$z_{12} \sim$ Dutzend, $z_{60} \sim$ zur Zeiteinteilung (von den Babyloniern).
Von Gottfried Wilhelm Leibniz (1679) stammte der Entwurf einer dual arbeitenden Rechenmaschine mit einem Stellenwertsystem zur Basis 2:

$$z_2 = \sum_{i=0}^{n-1} \alpha_i 2^i, \qquad \alpha_i \in \{0, 1\}.$$

Erst in diesem Jahrhundert erlangt das Dualsystem seine Bedeutung zum Rechnen: Konrad Zuse entwarf 1933 eine durch Relais gesteuerte Rechenmaschine.
Technische Realisierung des Dualsystems: 1/0-Codierung über Schalter, Relais, Röhren, Transistoren:

1: Spannung/Strom
0: keine Spannung/Strom

Hierdurch ist der Aufbau komplexer Rechenwerke für arithmetische Grundoperationen ermöglicht, deren effiziente Ausführung durch logische Schaltkreise erfolgt.

1.2.2 Das Rechnen mit Symbolen

Das "Buchstabenrechnen" wurde von der indischen Mathematik im frühen Mittelalter eingeführt. Hieraus leitet sich die algorithmische Lösung arithmetischer Aufgaben ab, z.B. die Division (Adam Ries, 1492-1559).
Damit kann eine erste, noch unscharfe Definition des Begriffes Algorithmus gegeben werden:

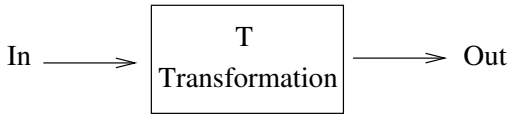
> Algorithmus: Umformung von gegebenen Größen auf Grund eines Systems von Regeln in andere Größen.

Z.B.: $(a + b)(a - b) = a^2 - b^2$ (Anwendung der Regeln der Algebra)
Das Rechnen mit Ziffern und mit Symbolen ist auf einer allgemeineren Ebene als gleichwertig anzusehen:

1. Das klassische Rechnen kann als Manipulation von Zeichenketten aufgefaßt werden.

2. Jede Manipulation von Zeichenketten nennt man *Rechnen* (Begriffserweiterung gegenüber der Umgangssprache!)

   Damit versteht man unter Rechnen eine regelhafte (algorithmische) Abbildung einer Eingangszeichenkette auf eine Ausgangszeichenkette:

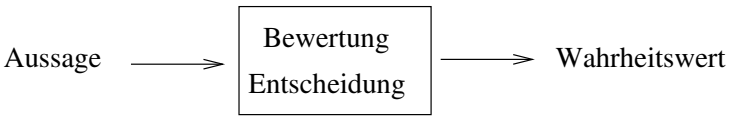


Unter Umständen kann für diese Abbildung (Transformation) ein geschlossener arithmetischer Ausdruck angegeben werden, generell ist sie aber durch eine abstraktere mathematische Funktion beschreibbar.
Beispiele für das Rechnen mit Symbolen:

- 1963 Programmsystem zur Formelmanipulation
- 1950 erster Textmanipulator: Editieren von Text
  1960 automatische Silbentrennung
- mechanische Sprachübersetzung (erste Ansätze 1933 Smirnov-Troyanski), machte Entwicklungen auf dem Gebiet der Linguistik erforderlich. Die Berücksichtigung des Kontextes ist notwendig:
  "HUND" $\rightarrow$ "DOG" aber "HUNDERT" $\rightarrow$ "HUNDRED"
- Schachspiel (N.Wiener, 1948; Shannon, 1950: grundlegende Theorie)
- Mathematische Beweise.

1.2.3 Logisches Rechnen

Die beim Rechnen mit Symbolen angegebene Transformation T kann auch die Zuordnung eines Wahrheitswertes $Out \in \{\text{true, false}\}$ zu einer Zeichenkette *In* erzeugen, wenn *In* eine Aussage darstellt:



Dabei sind *Aussagen* sprachliche Gebilde (Sätze), die zur Beschreibung und Mitteilung von Sachverhalten dienen, welche als wahr oder falsch bewertet werden können.

Es gilt das Prinzip der Zweiwertigkeit einer Aussage.
Die *Aussagenlogik* beschäftigt sich mit der Bewertung komplexer Aussagen.

- Leibniz (1672-76): Ansätze für ein logisches Kalkül
- Boole (1847): Algebraisierung der Logik
- Shannon (1938): erkannte Parallelen in der Arbeitsweise von Relaisschaltungen und Aussagenkalkül → Entwicklung einer Schalttheorie, z.B. systematisches Vereinfachen von Schaltfunktionen
- Abbot (1951): Bau einer digitalen Rechenanlage zur Durchführung logischer Verknüpfungen
- Frege (1879): Begründung der modernen Logik

Die *Prädikatenlogik* stellt eine Verallgemeinerung der Aussagenlogik dar:
Sie ermöglicht die Einführung einer symbolischen Sprache für die Beschreibung von Algorithmen und ihren Objekten. Die Geschichte der Algorithmen ist die Geschichte des logischen Rechnens mit Symbolen!
Heutiger Stand der Entwicklung:
Die Interpretation/Übersetzung von Anwenderprogrammen erfolgt nach logischen Regeln, die ihre Entsprechung in der Funktion der Schaltkreise des Rechners haben. Es existiert eine 1:1-Abbildung der Software auf die Hardware.

1.2.4 Das Rechnen mit Signalen

Die numerischen, symbolischen und logischen Berechnungsaufgaben sind äquivalent insofern, als daß sie sich auf diskrete Zeichen (Zahlen, Buchstaben, Operationssymbole) beziehen.
Symbole sind Objekte des menschlichen Geistes, sie sind Modellierungen von Phänomenen der Natur (z.B. interpretierte Signale).
Signale sind Phänomene der Natur. Sie haben unendlich viele Interpretationsmöglichkeiten. Welche hiervon gewählt wird, hängt von der Intention des Beobachters ab.
Albert Einstein zur Crux der Modellbildung:

> "Insofern sich die Sätze der Mathematik auf die Wirklichkeit beziehen, sind sie nicht sicher, und insofern sie sicher sind, beziehen sie sich nicht auf die Wirklichkeit."

Für die Symbolverarbeitung kann man von folgenden Annahmen ausgehen:

1. Symbole sind Abstraktionen
   Sie sind "sauber", d.h. nicht gestört. Sie geben die Wirklichkeit zwar nur näherungsweise wieder, enthalten aber das für die Aufgabe Wesentliche.
2. Symbole sind endlich, sie erfordern einen endlichen (vielleicht sogar minimalen) Aufwand zu ihrer Repräsentation und Beschreibung.
3. Typische Berechnungsaufgaben sind von begrenzter Komplexität, sie führen in endlicher Zeit zum gewünschten Ergebnis. Das trifft bei weitem nicht auf alle Probleme zu!
4. Typische Berechnungsaufgaben sind deterministisch, in der Auswahl der Bearbeitungsschritte besteht keine Freiheit.

Prinzipiell sollte man für die Signalverarbeitung von folgenden Annahmen ausgehen:

1. Signale sind die Wirklichkeit, sie enthalten relevante und nicht relevante Komponenten. Sie enthalten Störungen, die mit dem relevanten Signalanteil untrennbar verbunden sind. Signale sind "schmutzig".
2. Signale sind kontinuierlich, sie erfordern im Prinzip einen unendlichen Aufwand zur Repräsentation oder Beschreibung. Die Repräsentierung von Signalen im Rechner erfolgt in der Form von digitalen, d.h. diskretisierten und quantisierten Signalen.
3. Es gibt wesentliche Aufgabenklassen, die von sehr hoher Komplexität sind. Sie können mit deterministischen Algorithmen nur mit sehr hohem Aufwand (in sehr großer Zeit) oder überhaupt nicht exakt gelöst werden.
4. Typische Berechnungsaufgaben sind stochastisch (nicht deterministisch), in der Auswahl der Bearbeitungsschritte werden Freiheiten zugelassen (z.B. Heuristiken, Zufallsprozesse zur Steuerung).

Mit dem Rechnen mit Signalen beschäftigen sich folgende relativ junge Disziplinen:

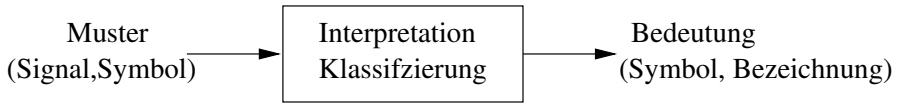
- Signaltheorie (ca. 1945)
- Nachrichtentheorie (ca. 1945)
- Mustererkennung (ca. 1960)
- Neuroinformatik (ca. 1980)
- maschinelles Lernen (ca. 2000)
- Data Science (ca. 2010)

Zur Erfassung des Zusammenhangs zwischen einem gestörten Signal und einem Symbol werden die Theorie stochastischer Prozesse (entwickelt aus Wahrscheinlichkeitstheorie), die Informationstheorie und Entscheidungstheorie herangezogen. Die Problemlösung, wie aus Umweltdaten auf den symbolischen Gehalt von gestörten Signalen geschlossen werden kann, hat sich in den letzten 5-10 Jahren durch maschinelles Lernen, und insbesondere durch den Einsatz von neuronalem Lernen, entscheidend verbessert. Der Einsatz von komplexen neuronalen Netzen (deep Neural Networks) in allen Bereichen der künstlichen Intelligenz hat daher in letzter Zeit enorm zugenommen und erzielt große Erfolge. Hier ist insbesondere das Zusammenspiel von graphischen Computereinheiten (GPUs) mit konfigurierbaren Softwarebibliotheken zu nennen, wie sie von den GPU-Herstellern wie NVIDIA und von großen datenverarbeitenden Konzernen wie Google zur Verfügung gestellt werden. Zusammen mit den enormen Datenmengen, die inzwischen von diesen Firmen gesammelt und zum Training der Netze eingesetzt werden, haben sich ganz neue Disziplinen der Data Science entwickelt. Damit soll die Zukunft des autonomen Fahrens, der automatischen Diagnose von Krankheiten, der smart Industry, des Internet of Things und anderer Gebiete vorangetrieben werden.

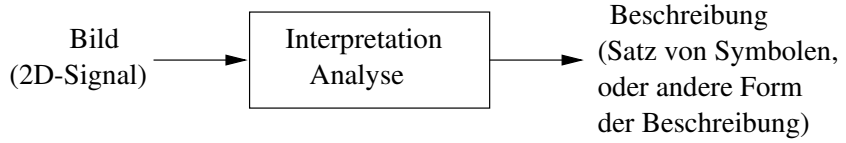
Diese Entwicklung wurde nur durch den enormen Zuwachs an verfügbarer Rechenleistung möglich, der sich in den letzten 40 Jahren durch eine exponentielle Erhöhung der Rechenkapazität (approximiert durch das Moor'sche "Gesetz") ausdrücken läßt: Alle zwei Jahre verdoppelt sich die Rechenleistung. Das bedeutet nach 40 Jahren eine Erhöhung um 10^6, also um eine Million!

Beispiele für das Rechnen mit Signalen sind

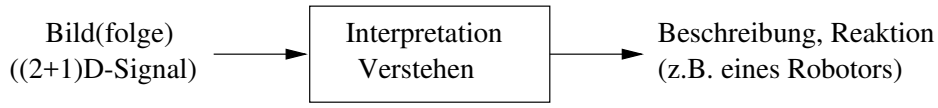
- *Mustererkennung* (Pattern Recognition, ca. 1960)
  Als *Muster* wird eine durch eine Menge von Messungen oder Beobachtungen erzeugte Struktur bezeichnet.



- *Bildverarbeitung* (Image Processing, ca. 1965)
  Unter einem *Bild* wird eine matrixförmige Anordnung von Helligkeitswerten verstanden.



- *Maschinensehen* (Computer Vision, ca. 1980)
  Eine *Szene* (durch Kamera beobachteter Weltausschnitt), gegeben durch ein Einzelbild oder eine Bildfolge, soll interpretiert werden, um Beschreibungen oder Reaktionen abzuleiten.



  Die Interpretation oder das Verstehen einer Bildfolge ist nicht als lineare Folge von Berechnungsprozessen realisierbar. Es stellt sich die Frage, ob sich die visuelle Wahrnehmung des Menschen technisch realisieren läßt. Das Symbolverarbeitungsparadigma der Kognitionswissenschaft führt nicht zur Lösung. Zu vielen Problemen der Wahrnehmung können keine Algorithmen explizit angegeben werden.
  In der *Neuroinformatik* sind in Form von neuronalen Netzen Ansätze zu Berechnungsverfahren begründet, die auf explizite Formulierungen von Algorithmen verzichten.

1.2.5 Die Entwicklungsgeschichte der Rechenmaschine

Die bis zu unseren heutigen Computern führende Entwicklung von Rechenmaschinen verlief in 3 Etappen:

1. 17. Jahrhundert: Mechanisierung des Rechnens mittels mechanischer Rechenwerke $\rightarrow$ einfache Ausführung der Grundrechenarten
2. 19. Jahrhundert: Automatisierung des Rechnens mittels Daten- oder Programmspeicher und Steuerwerk $\rightarrow$ komplexere Berechnungsabläufe
3. ca. 1940/1950: Einführung der frei programmierbaren elektronischen Universalrechenmaschine $\rightarrow$ Entwicklung bis zum heutigen Computer über 5 Generationen

Im folgenden werden einige Stationen der 3 Etappen skizziert:

- Leibniz(1673): 12-Dekaden-Rechenmaschine
  Schnelle Multiplikation mit Hilfe von Zehnerpotenzen, stufenweise verschiebbare Zahnradwalze (Staffelwalze), dieses Prinzip wurde von fast allen mechanischen Rechenmaschinen übernommen.
- Charles Babbage (1843): analytical engine
  Bahnbrechendes Konzept eines universellen Rechners mit

  - Rechenwerk
  - Steuereinheit für Iteration und bedingte Verzweigung
  - Zahlenspeicher
  - E/A über Lochkarten-Band nach Jacquard (1805)

  zur Berechnung ballistischer Tafeln für die britische Marine. Die analytical engine wurde jedoch nicht fertiggestellt, da nicht die nötige Präzision zur Fertigung der Zahnräder erreicht werden konnte.

- Hermann Hollerith (1886): Zähl- und Registriermaschine auf der Basis von Lochkarten zur Auswertung der amerikanischen Volkszählung 1890. Sie ermöglichte verschlüsselte Angaben von Personen auf Lochkarten und verfügte über eine elektrische Abtastapparatur, die über Fühler die Kodierung an magnetische Zählwerke weitergab.
  Hollerith war der Gründer der Firma International Business Machines (IBM).

- Konrad Zuse (Berliner Bauingenieur) war der Erfinder der ersten elektronischen Rechenmaschine.

  - Z1/Z2 (ca. 1938): Entwurf eines mechanischen Rechners. Die Realisierung der mechanischen Teile gelang nicht. Das Konzept war ähnlich der analytical engine von Babbage, es verwendete das Dualzahlsystem, realisierte Gleitkommazahlen und logische Operationen (Und, Oder, Negation)
  - Z3 (1941): erster frei programmierbarer, elektronischer, durch Relais gesteuerter Rechner
    Es wurden 2600 Fernmelderelais für das Rechenwerk und den Speicher verwendet. Die Z3 verfügte über 64 Speicherplätze für Zahlen, 22-stellige Dual- und 7-stellige Dezimalzahlen und beherrschte die 4 Grundrechenarten sowie das Radizieren. Es konnten 15-20 Additionsoperationen pro Sekunden durchgeführt werden, eine Multiplikation beanspruchte jedoch 4 Sekunden. Abbildung 1.1 skizziert den Rechenablauf.

- COLOSSUS (40er Jahre): erster Röhrenrechner
  Dechiffriermaschine unter Mitarbeit von Alan Turing, der in den 30er Jahren mit seiner Turing-Maschine ein Gedankenexperiment für einen maschinell deutbaren Algorithmenbegriff schuf.

- ENIAC (I.P.Eckert, J.W. Mauchly, 1946):

  - 18000 Röhren, 1500 Relais, 2000 mal schneller als elektromechanische Relais-Rechner
  - Additionen: 0.2 ms, Multiplikation zweier zehnstelliger Zahlen: 2.8 ms

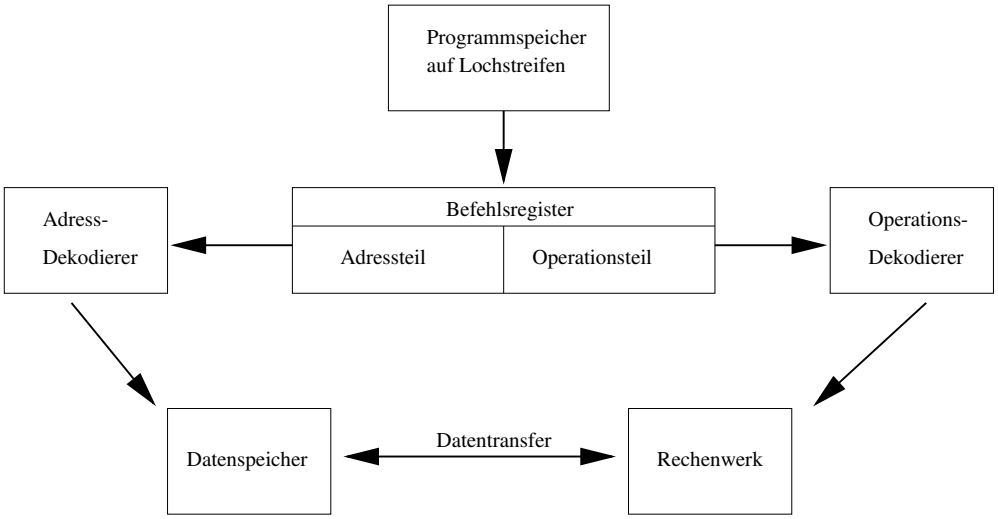


Abbildung 1.1: Rechenablauf der Z3.

  - Programmierung über Steckbretter
  - kein Programmspeicher

  ENIAC bildete den Ausgangspunkt des Rechnerbaus im Rahmen des Manhatten-Projekts.

- EDVAC (John von Neumann): Mit seinem ersten frei programmierbaren Allgemeinrechner setzt von Neumann den Beginn der modernen Rechnerarchitektur: "von-Neumann-Rechner"

  - binäre Kodierung
  - datenabhängige Programmverzweigung
  - sequentielle Verarbeitung
  - einheitlicher interner Daten- und Programmspeicher. Speicherung des Programms im internen Maschinenspeicher (zuvor wurden die Programme während der Ausführung eingegeben)
  - Realisierung datenabhängiger Programmschleifen ("While-Schleifen").

  Der Architektur liegt ein mathematisches Modell einer Berechnung (Turing-Maschine) zugrunde. Es wurde eine begriffliche Trennung zwischen Schaltlogikfunktionen und technischer Realisierung getroffen (McCulloch/Pitts-Zelle).

1.2.6 Die Generationen der elektronischen Rechenmaschine

Nach der Entwicklung von ENIAC und EDVAC setzte eine stürmische Entwicklung weiterer Unikate ein, die erst Anfang der 50er Jahre zu einer Standardisierung führte, durch die beginnende Kommerzialisierung durch IBM, UNIVAC u.a.

Jede Generation ist durch typische Entwicklungsstufen von Software und Hardware charakterisiert.

1. Generation: 1953-1958
   - etwa 250 Anlagen
   - Vakuumröhren, handverdrahtet, Externspeicher: Magnetbänder, Magnettrommel
   - Zugriffszeiten 10^{-3} sec
   - Betriebssystem u. Compiler gab es nicht
   - Programmierung in Maschinensprache
2. Generation: 1958-1966
   - Transistoren und magnetische Kernspeicher
   - Erstellung von Systemsoftware, Compiler, E/A-Hilfen
     am Ende der Periode:
     - universelle Betriebssysteme
     - Compiler für Cobol, Algol, Fortran
     - Zugriffszeiten bis 10^{-6} sec
     - verdrahtet
   - erste wissenschaftliche Rechnungen und Echtzeitinstallationen
   - Stapelbetrieb
   - Berufsbild des Informatikers entsteht
3. Generation: 1966-1974
   - integrierte Schaltungen: drastisch sinkende Hardwarepreise, Miniaturisierung, Zugriffszeiten bis 10^{-9} sec
   - Halbleiterspeicher, Bildschirm- und Dialogstationen.
     Am Ende der Periode: Mikroschaltkreistechnologie
   - sehr große und komplexe Rechenanlagen (Großrechner). "Softwarekrise" $\rightarrow$ Beginn verbesserter (systematischer) Softwareentwicklungsmethoden.

- stürmische Entwicklung auf dem Gebiet der Echtzeitrechner, Prozeßrechner zur Steuerung werden immer kleiner (Kleinrechner). Echtzeit-Betriebssysteme
- Beginn einer Dezentralisierung der Datenverarbeitungsaufgaben

4. Generation: 1974-1982

- Miniaturisierung der Schaltkreise in der Mikroelektronik (VLSI-Technologie, $> 10^5$ Gatter pro Chip). Mikrorechner: alle Einheiten eines Rechners auf einem Schaltkreis zusammengefaßt.
- Schaltkreise für CPU: 130 000 Transistoren
- 64KBit-Speicher auf einem Schaltkreis
- Superrechner, Personalcomputer, Homecomputer
- Konsolidierung: Hardware- und Softwaresysteme wachsen zu einer Einheit zusammen
- teilweise Überwindung der Softwarekrise

5. Generation: seit 1982

- rasante Miniaturisierung der VLSI-Technologie, Beispiel Speicher:
  1980: 64KBit RAM
  1983: 256KBit RAM
  1986: 1MBit RAM
  1992: 16MBit RAM
  2000: 128MBit RAM, 0.18 μ Technologie
  2010: Core-i7 mit 1 Mrd. Transistoren
  2018: Multiprozessoren mit 10 Mrd. Transistoren und 10nm Technologie
- Einbau von neuronalen Netzen für biometrische Datenerfassung (bionic chip) bereits in den i-phones
- rasante Zunahme der Rechengeschwindigkeit (Taktfrequenzen bis 4 GHz), Abnahme der Volumina, Zunahme der internen Speicher
- rasante Entwicklung massiv paralleler Rechner
- im Dreijahreszyklus: Verdopplung der Taktfrequenz, Vervierfachung der Zahl der Transistoren pro Chip
  im Jahreszyklus: Verdopplung der Prozessorleistung
  seit 90er Jahren: Vervierfachung des Integrationsgrades im Zweijahreszyklus
- generelle Informatik-Durchdringung aller Lebensbereiche

1.2.7 Ein Resumé: Wohin geht die Informatik?

1. "Quantensprünge" in der Hardware
   - Weitere Integration von Speicher- und Prozessor-Chips bis an die physikalischen Grenzen (bereits erreicht auf mikroelektronischem Weg: 100 Millionen Transistoren pro cm^2)
   - Weiteres Ausreifen des Parallelverarbeitungs-Paradigmas
   - Photonik (Photonen als Informationsträger)
     - optische Prozessoren (z.Zt. 1000 Schaltelemente pro cm^2)
     - Vernetzung über Glasfaser
       - ▷ Bandbreite einer Glasfaser: 85 Terahertz (Billionen Schwingungen pro Sekunde)
       - ▷ Übermittlung durch eine Glasfaser: 300 Billionen Bit/s (theoretisch, bisher erreicht: 400 Gigabit/s)
       - ▷ Bandbreite in der Nachrichtentechnik z.Zt. 85 Gigahertz
     - erste optische Computer in ca. 10 Jahren
2. Nano-Technologie
   Integration informationsverarbeitender und mechatronischer Systeme auf dem Chip (Micro-Motoren, Micro-Spiegel Projektoren)
3. Globale und lokale Vernetzung
   Die gegenwärtige Internet-Euphorie ist nur ein Schatten künftiger Entwicklung.
4. Fortschreitende Theorienbildung/Mathematisierung
   - Automatisierung von Softwareentwurf, -entwicklung und -prüfung
   - Informationsverarbeitende Modelle für Wahrnehmungs- und kognitive Leistungen
   - Softwareentwicklung für Parallelverarbeitung
5. neue Paradigmen
   - Entwicklung von Künstlicher Intelligenz und Neuroinformatik
   - Quanten-Computer
6. Innovative Anwendungen und fortschreitende Computerisierung der "technologischen Zivilisation"
   - Informatik-Wissen gehört zu beliebigen anderen Disziplinen (wie Mathematik und Beherrschung einer Schriftsprache)
   - gesellschaftliche Konsequenzen müssen auch gesellschaftlich gesteuert werden

1.3 Darstellung von Zeichen und Zahlen

Zeichen und Zahlen müssen repräsentiert werden.

Daten unterschiedlichen Typs (unterschiedlicher Sorten) kommen zur Anwendung. Der Typ eines Datums bestimmt, welche Operationen mit diesem Datum ausführbar sind, denn nicht jedes Datum ist als Operand für jeden Operator geeignet.

Beispiele:

int	Menge der ganzen Zahlen
char	Menge der durch einzelne Zeichen bezeichneten Textsymbole
bool	Menge der Wahrheitswerte 'L'=≪wahr≫, '0'=≪falsch≫

Die formale Verkoppelung der Datentypen (Sorten) mit den auf diesen ausführbaren Grundoperationen in Form von Rechenstrukturen stellt die Basis einer jeden Programmiersprache dar.

Hier interessiert der Typ insofern, als er die unterschiedlich zu repräsentierenden Daten zu unterscheiden gestattet.

Die Anzahl der Objekte eines bestimmten Typs kann unendlich sein. Ein Objekt muß aber stets durch endlich viele Zeichen aus einem Zeichenvorrat wiederzugeben sein. Tatsächlich können in Rechnern nur endlich viele verschiedene Objekte eines bestimmten Typs repräsentiert werden.

Mit n Bits kann man bei Binärcodierung $N = 2^n$ verschiedene Zustände repräsentieren, die den Dezimalzahlen aus dem Bereich $0, \ldots, (2^N - 1)$ zugeordnet werden können (vgl. Tabelle 1.1).

Definition: (Binärzahl)

Eine Dualzahl $z_2 \in \mathbb{B}^n$ der Länge n Bits heißt Binärzahl, wenn sie im Stellenwertsystem erklärt ist,

$$z_2 = \sum_{i=0}^{n-1} \alpha_i 2^i, \;\; \alpha_i \in \mathbb{B}.$$

Sie vermag $N = 2^n$ verschiedene Zustände eines Sachverhaltes zu repräsentieren.

Diese Zustände können Daten unterschiedlichen Typs oder auch Adressen der Daten im Arbeitsspeicher bzw. Operationen über diesen Daten sein.

Bits	Kombinationen	darstellbare Dezimalzahlen	möglicher Bereich
1	0,1	$2 = 2^1$	$0 \cdots 1$
2	00,01,10,11	$4 = 2^2$	$0 \ldots 3$
3	000,001,010,011, 100,101,110,111	$8 = 2^3$	$0 \ldots 7$
⋮			
8		$256 = 2^8$	$0 \ldots 255$
⋮			
16		65536=2^{16}	$0 \ldots 65535$
⋮			
32		4.294.967.296 $= 2^{32}$	$0 \ldots$4.294.967.295
⋮			
64		$\cdots = 2^{64}$	

Tabelle 1.1: Mit n Bits darstellbare Dezimalzahlen.

Aus Gründen der Ökonomie des Rechnerbaues gilt:

- Es wird stets eine Codierung in fester Wortlänge n benutzt
- n ist stets eine Zweierpotenz

  n=8, 16, 32 oder 64

  Jeder Rechner hat seine spezifische Datenwortlänge.
- Der Zugriff auf die Daten erfolgt meist parallel in Gruppen von mindestens 4 Bit Breite.
  Üblich sind folgende Bezeichnungen (für 32-Bit-Rechner):

  4 Bit: Nibble
  8 Bit: Byte
  16 Bit: Halbword
  32 Bit: Word
  64 Bit: Double Word

Die Verwendung von Codes der Basis 8 (*Oktalcode*) und 16 (*Hexadezimalcode*) führt zu einer komprimierten Darstellung:

Beispiel:

$$\underbrace{11\ldots1}_{16}{}_2 = 177777_8 = \mathrm{FFFF}_{16}.$$

Definition: (b-adische Zahlendarstellung)

Seien $b \in \mathbb{N}, b > 1$ eine Basis und $n_b \in \mathbb{N}$ die Stelligkeit einer Zahlendarstellung. Dann ist für $<\alpha_{n_b-1}\ldots\alpha_0>_b,\ \alpha_i \in \sigma_b, i = 0, 1, \ldots, n_b - 1,$ $n_b \in \mathbb{N}_+$, durch

$$z_b = \sum_{i=0}^{n_b-1} \alpha_i b^i = \alpha_{n_b-1} b^{n_b-1} + \alpha_{n_b-2} b^{n_b-2} + \ldots \alpha_1 b + \alpha_0$$

eine Zahlendarstellung zur *Basis* b gegeben.

Dabei sei σ_b der Zeichenvorrat (Ziffern α_i) der Zahlendarstellung. Es sind zwei Fälle möglich:

1. $<\alpha_{n_b-1}\ldots\alpha_0>_b$ ist ein Wort aus σ_b^*: Darstellung mit variabler Länge ($\alpha_{n_b-1} \neq 0$, d.h. keine führende Nullen, $n_b^{\min} = \lfloor \log_b z_{10} \rfloor + 1$)

   oder

2. $<\alpha_{n_b-1}\ldots\alpha_0>_b$ ist ein Wort aus $\sigma_b^{n_b}$: Darstellung mit fester Länge (n_b fest vorgegeben, $n_b \geq n_b^{\min}$, führende Nullen zugelassen)

- b= 2 : Binärzahlsystem $\sigma_2 = \{0, 1\}$
  b= 8 : Oktalzahlsystem $\sigma_8 = \{0, 1, \ldots, 7\}$
  b=10 : Dezimalzahlsystem $\sigma_{10} = \{0, \ldots, 9\}$
  b=16 : Hexadezimalzahlsystem $\sigma_{16} = \{0, \ldots, 9, \mathrm{A}, \ldots, \mathrm{F}\}$

- Mit der Wahl der Stellenzahl n_b einer b-adischen Zahl

$$z_b = \sum_{i=1}^{n_b-1} \alpha_i b^i$$

  trifft man eine Entscheidung für den Umfang darstellbarer Zustände

$$N_b = b^{n_b}$$

Dezimal $b = 10$	Binär $b = 2$	Oktal $b = 8$	Hexadez. $b = 16$	
0	0000	00	0	1 Bit
1	0001	01	1	
2	0010	02	2	2 Bits
3	0011	03	3	
4	0100	04	4	3 Bits
5	0101	05	5	
6	0110	06	6	
7	0111	07	7	
8	1000	10	8	4 Bits
9	1001	11	9	
10	1010	12	A	
11	1011	13	B	
12	1100	14	C	
13	1101	15	D	
14	1110	16	E	
15	1111	17	F	

Tabelle 1.2: Zahlen $0, \ldots, 15$ im Dezimal-, Binär-, Oktal- und Hexadezimalzahlsystem

- Sind diese Zustände abzählbar (normalerweise gegeben), entspricht dies der Möglichkeit der Codierung von N_b Zahlen z_b, $N_b \in \mathbb{N}$, $N_b = |\{z_{10}\}|$, $z_{10} \in \mathbb{N}$

$$C : z_{10} \longrightarrow z_b \qquad , z_{10}, z_b \in \mathbb{N}$$

  Z.B.: $b = 10$, $n_{10} = 2 \rightsquigarrow N_{10} = 10^2 \ : \ z_{10} \in \{00, 01, \ldots, 99\}$
  Z.B.: $z_{10} = 32 \ = \ 3 \cdot 10^1 + 2 \cdot 10^0$

- Für jede Zahl $x \in \mathbb{Z}$ und jede Zahl $y \in \mathbb{N}$, $y{>}0$, gilt auf der Grundlage der Operation "*Division mit Rest*" die folgende Identität: $x = qy + r$ mit q als Quotient und r als Rest für alle b-adischen Zahlendarstellungen

$$x = (x \text{ div } y)\, y + x \bmod y$$

  - *ganzzahlige Division:* $q = x \text{ div } y = \left\lfloor \frac{x}{y} \right\rfloor$
    mit $\lfloor \alpha \rfloor = \max\{z \in \mathbb{Z} | \, z \leq \alpha, \alpha \in \mathbb{R}\}$
    (Untere Gauß-Klammer $\sim$ ganzzahliger Anteil der Division)
  - *Rest der ganzzahligen Division:* $r = x \bmod y = x - \left\lfloor \frac{x}{y} \right\rfloor y, \qquad 0 \leq x \bmod y \ < \ y$

1. Es gilt $z_b = z_b \bmod N_b$
   $\Rightarrow$ Jede n_b-stellige b-adische Zahl ist gleich dem Rest ihrer ganzzahligen Division durch N_b.
   $\hookrightarrow$ Eine Zahl mit $n{>}n_b$ Stellen wird nur bezgl. der letzten n_b Stellen darstellbar sein. Das entspricht der Darstellung auf einem *Zahlenkreis* mit N_b Positionen.
   Z.B. $b = 10, \quad n_{10} = 2, \quad$ aber $n = 3$
   $432 \to 32 \quad \sim$ Die Zahl 432 wird auf die Zahl 32 abgebildet.

2. Es gilt $\alpha_i = (z_b \text{ div } b^i) \bmod b$
   Die Koeffizienten α_i jeder b-adischen Zahl ergeben sich als Rest der ganzzahligen Division $\lfloor \frac{z_b}{b^i} \rfloor$ bzgl. der Basis b.

$$\alpha_i = \left\lfloor \frac{z_b}{b^i} \right\rfloor - \left\lfloor \frac{\lfloor \frac{z_b}{b^i} \rfloor}{b} \right\rfloor b$$

   Z.B.: $z_{10} = 32 = 3 \cdot 10^1 + 2 \cdot 10^0$
   $\alpha_0 = \lfloor \frac{32}{1} \rfloor \bmod 10 = 32 - \lfloor \frac{32}{10} \rfloor \cdot 10 = 32 - 3 \cdot 10 = 32 - 30 = 2$
   $\alpha_1 = \lfloor \frac{32}{10} \rfloor \bmod 10 = 3 - \lfloor \frac{3}{10} \rfloor \cdot 10 = 3 - 0 \cdot 10 = 3 - 0 = 3$

Hieraus leitet sich ein besonders einfaches Verfahren der Konvertierung von Codes zwischen Dual-, Oktal- und Hexadezimalsystem ab.

1. $z_8, z_{16} \longrightarrow z_2$: jede Oktal-/Hexziffer α_i^b wird für sich in die entsprechende Dualzahl umgewandelt
2. $z_2 \longrightarrow z_8, z_{16}$: Zusammenfassen der Dualziffern α_i^2 zu Gruppen von 3 bzw. 4 Bit Breite und Konvertierung dieser Gruppe in den Zielcode

(Das Verfahren gilt allgemein, wenn Quell- und Zielbasis zueinander durch eine Potenz im Verhältnis stehen: $q = p^{\pm n}$, z.B. 10 und 100).

Beispiele:

zu 1. $123_8 \longrightarrow 001\,010\,011_2$
$4A8_{16} \longrightarrow 0100\,1010\,1000_2$

zu 2. $10000101_2 \longrightarrow 205_8$ oder 85_{16}

1.3.1 Zeichencodes

Zeichencodes werden zur Repräsentation von Texten benötigt:

- Eingaben über die Tastatur
- Programme (“Quellprogramme”)
- Datenbanken.

Zeichen werden ebenfalls binär repräsentiert.
Da der Umfang des Alphabets der Schriftsprache gering ist (Deutsch $\lesssim$ 30 Zeichen), enthalten Zeichencodes mehr Information:

1. Buchstaben (groß + klein)
2. Ziffern (1...9)
3. mathematische Operationssymbole (Klammern, Vergleiche, arithm. Operationen)
4. nicht darstellbare Steuerzeichen, z.B. dient CR (carriage return - Wagenrücklauf) der Formatierung (Zeilenabschluß) einer Eingabe über Tastatur

Es genügen 7 Bit zur Codierung des Zeichenvorrats einer Sprache. Es existieren verschiedene standardisierte Codes:

- ”ISO 7-Bit Codes”:

  ISO 8859-1: westeuropäische Sprachen
  ISO 8859-2...4: nord-, ostmittel, südosteur. Sprachen
  ISO 8859-5...9: kyrillisch, griechisch, hebräisch, arabisch

- ASCII-Code (American Standard Code for Information Interchange) $\hat{=}$ DIN 66003

b8	b7	b6	b5		b4	b3	b2	b1		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX
				0	0	0	0	0	0			SP	0	@	P	`	p								ð	
				1	0	0	0	1	1			!	1	A	Q	a	q		‘	¡	±	À	Ð	à	ñ	
				2	0	0	1	0	2			"	2	B	R	b	r		’	¢	²	Á	Ñ	á	ò	
				3	0	0	1	1	3			#	3	C	S	c	s		“	£	³	Â	Ò	â	ó	
				4	0	1	0	0	4			$	4	D	T	d	t		”	¤	´	Ã	Ó	ã	ô	
				5	0	1	0	1	5			%	5	E	U	e	u		•	¥	µ	Ä	Ô	ä	õ	
				6	0	1	1	0	6			&	6	F	V	f	v		–	¦	¶	Å	Õ	å	ö	
				7	0	1	1	1	7			'	7	G	W	g	w		—	§	·	Æ	Ö	æ	÷	
				8	1	0	0	0	8			(	8	H	X	h	x		~	¨	¸	Ç	×	ç	ø	
				9	1	0	0	1	9			)	9	I	Y	i	y		™	©	¹	È	Ø	è	ù	
				10	1	0	1	0	10			*	:	J	Z	j	z		š	ª	º	É	Ù	é	ú	
				11	1	0	1	1	11			+	;	K	[	k	{		›	«	»	Ê	Ú	ê	û	
				12	1	1	0	0	12			,	<	L	\	l	|		œ	¬	¼	Ë	Ü	ë	ü	
				13	1	1	0	1	13			-	=	M	]	m	}			-	½	Ì	Ý	ì	ý	
				14	1	1	1	0	14			.	>	N	^	n	~			®	¾	Í	Þ	î	þ	
				15	1	1	1	1	15			/	?	O	_	o			Ÿ	-	¿	Î	ß	ï	ÿ	
										0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

SONDERZEICHEN

Beispiele: ASCII-Code

'G' = 0100 $0111_2 = 47_{16} = 107_8$
'H␣A' = 0100 1000 0010 0000 0100 0001_2
= 48 20 41_{16}= 22020101_8
'32' =0011 0011 0011 0010_2=33 32_{16}=31462_8

Texte werden byteweise orientiert gespeichert: 1 Byte pro Zeichen
Zeichensequenzen werden als Zeichenketten in logisch zusammenhängenden Strukturen (String) gespeichert:

Beispiele:

Darstellung mit fester Länge (8 Bytes):

H	A	L	L	O	!	␣	␣

H	A	L	L	O	␣	P	E

Darstellung mit variabler Länge:

H	A	L	L	O	␣	P	E	T	E	R	!	NUL			

mit <NUL> als eindeutig festgelegtem Abschlußzeichen.

1.3.2 Darstellung von Zahlen

In der Mathematik kennt man unterschiedliche Zahlbereiche, die durch eine Enthaltenseinsrelation in Beziehung stehen:

$\mathbb{N}$	natürliche Zahlen	$\mathbb{N} = \{0, 1, 2, \ldots\}$
$\cap$		
$\mathbb{Z}$	ganze Zahlen	$\mathbb{Z} = \{\ldots, -1, 0, 1, 2, \ldots\}$
$\cap$		
$\mathbb{Q}$	rationale Zahlen	z.B. $\frac{3}{2} = 1.5$, $\frac{82}{7} = 11.\overline{714285}$
$\cap$		
$\mathbb{R}$	reelle Zahlen	z.B. $\sqrt{2},\ \pi,\ e$
$\cap$		
$\mathbb{C}$	komplexe Zahlen	$c = a + jb,\ a, b \in \mathbb{R},\ j := \sqrt{-1}$

Sie gestatten, die Klassen zu lösender arithmetischer Gleichungen zu erweitern, z.B.:

$\mathbb{N}$	$a + x = b,\ a \leq b$
$\mathbb{Z}$:	$a + x = b$
$\mathbb{Q}$:	$a \cdot x = b,\ a \neq 0$
$\mathbb{R}$:	$a + x^2 = b,\ a \leq b$
$\mathbb{C}$:	$a + x^2 = b$

Die beschränkte Stellenzahl der Computer hat grundlegende Konsequenzen für die in der Informatik zugelassenen Zahlbereiche:

- natürliche Zahlen sind bis zu einer Obergrenze exakt darstellbar
- ganze Zahlen halbieren diesen Darstellungsbereich
- rationale Zahlen sind mit endlicher Genauigkeit darstellbar
- reelle Zahlen sind prinzipiell nur durch Approximationen darstellbar
- komplexe Zahlen werden als Paare reeller Zahlen dargestellt

1.3.2.1 Darstellung natürlicher Zahlen

Die Vorgabe von n Bits für die Binärcodierung einer natürlichen Zahl schränkt den repräsentierbaren Bereich auf das Intervall $[0, \ldots, 2^n - 1] \subset \mathbb{N}$ ein. Zur Konvertierung der Darstellung natürlicher Zahlen ist im allgemeinen Fall, daß Quell- und Zielbasis q bzw.

p nicht durch eine Potenz im Verhältnis stehen ($q \neq p^{\pm m}$), die Darstellung natürlicher Zahlen nach dem *Horner-Schema* von Vorteil:

$$z_b = \sum_{j=0}^{n-1} \alpha_j b^j = (\ldots((\alpha_{n-1}b + \alpha_{n-2})b + \alpha_{n-3})b + \ldots + \alpha_1)b + \alpha_0$$

Ausgehend von der Identität

$$z_b = (z_b \text{ div } b)b + z_b \bmod b$$

beruht das Horner-Schema auf der rekursiven Anwendung der Division durch die Basis b und der Abspaltung der Koeffizienten α_i zu jedem Rekursionsschritt i. Auf diese Art und Weise wird, beginnend mit der niedrigsten Potenz (Null) in der Stellenwertdarstellung der Zahl z_b, deren Umwandlung in das Horner-Schema durchgeführt.

Initialisierung:

$$\begin{aligned} z_b &= z_{(0)} = (z_{(0)} \text{ div } b)b + z_{(0)} \bmod b = z_{(1)}b + \alpha_0 \\ &= \sum_{j=0}^{n-1} \alpha_j b^j = (\sum_{j=1}^{n-1} \alpha_j b^{j-1})b + \alpha_0 \\ \alpha_0 &= z_{(0)} \bmod b = z_{(0)} - z_{(1)}b \end{aligned}$$

Rekursion: für $0 < i \leq n-1$ ist das Ergebnis der ganzzahligen Division

$$\begin{aligned} z_{(i)} &= z_{(i-1)} \text{ div } b = (z_{(i)} \text{ div } b)b + z_{(i)} \bmod b = z_{(i+1)}b + \alpha_i \\ &= \sum_{j=i}^{n-1} \alpha_j b^{j-i} = (\sum_{j=i+1}^{n-1} \alpha_j b^{j-i-1})b + \alpha_i \\ z_{(n)} &= 0 \\ \alpha_i &= z_{(i)} \bmod b = z_{(i)} - z_{(i+1)}b = z_{(i-1)} \text{ div } b - z_{(i+1)}b \end{aligned}$$

also:

$$\begin{aligned} \alpha_1 &= z_{(1)} \bmod b = (z_{(0)} \text{ div } b) \bmod b \\ \alpha_2 &= z_{(2)} \bmod b = (z_{(1)} \text{ div } b) \bmod b = ((z_{(0)} \text{ div } b) \text{ div } b) \bmod b \\ &\vdots \\ \alpha_i &= z_{(i)} \bmod b = (\ldots((z_{(0)} \text{ div } b) \text{ div } b)\ldots \text{ div } b) \bmod b \end{aligned}$$

Beispiel:

$$
\begin{aligned}
z_{10} &= 2345 \\
&= 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 \\
z_{(0)} &= (2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0)10 + \quad \underline{5} \\
z_{(1)} &= 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 \\
&= (2 \cdot 10^1 + 3 \cdot 10^0)10 + \quad \underline{4} \\
z_{(2)} &= 2 \cdot 10^1 + 3 \cdot 10^0 \\
&= (2 \cdot 10^0)10 + \quad \underline{3} \\
z_{(3)} &= 2 \cdot 10^0 = \quad \underline{2}
\end{aligned}
$$

$$
z_{10} = ((2 \cdot 10 + 3)10 + 4)10 + 5
$$

Für die Konvertierung aus einer Quellsystem-Darstellung in eine Zielsystem-Darstellung gibt es verschiedene Verfahren: Rechnen im *Quellsystem* oder im *Zielsystem*, je nachdem welches Verfahren einfacher ist.

A) Divisionsmethode (Quellverfahren)

Notationen:

$z_q = \sum_{i=0}^{n_q-1} \alpha_i q^i$ ist Zahl im Quellsystem-Darstellung

$z_p = \sum_{i=0}^{n_p-1} \beta_i p^i$ ist Zahl im Zielsystem-Darstellung

$z_{(i)}$ Ergebnis des i-ten Schrittes der ganzzahligen Division von z_q durch die Basis des Zielsystems p_q, d.h. in Quellsystemdarstellung.

Aus dem Horner-Schema leitet sich folgendes Schema für die Berechnung der Koeffizienten β_i der Zielsystemdarstellung ab:

$$
\begin{array}{llll}
z_{(0)} = z_q & & \beta_0 = z_{(0)} \bmod p_q = z_q \bmod p_q \\
z_{(1)} = z_{(0)} \operatorname{div} p_q & & \beta_1 = z_{(1)} \bmod p_q = (z_q \operatorname{div} p_q) \bmod p_q \\
z_{(2)} = z_{(1)} \operatorname{div} p_q & & \beta_2 = z_{(2)} \bmod p_q = ((z_q \operatorname{div} p_q) \operatorname{div} p_q) \bmod p_q \\
\vdots & & \vdots \\
z_{(i)} = z_{(i-1)} \operatorname{div} p_q & & \beta_i = z_{(i)} \bmod p_q
\end{array}
$$

Das heißt, sukzessive Division von $z_{(i)}$ durch p liefert die Koeffizienten β_i als Divisionsreste. Die Division erfolgt solange, bis $z_{(i)} = 0$, d.h. bis die Zahl in Quellsystem-Darstellung durch fortgesetzte Division völlig aufgebraucht ist.

Beispiel: Konvertierung 4711_{10} **in** 1001001100111_2

$$
\begin{array}{rclclrclcl}
z_{(0)} & = & 4711_{10} & & & \beta_0 & = & z_{(0)} \text{ mod } 2_{10} & = & 1_{10} = 1_2 \\
z_{(1)} & = & z_{(0)} \text{ div } 2_{10} & = & 2355_{10} & \beta_1 & = & z_{(1)} \text{ mod } 2_{10} & = & 1_{10} = 1_2 \\
z_{(2)} & = & z_{(1)} \text{ div } 2_{10} & = & 1177_{10} & \beta_2 & = & z_{(2)} \text{ mod } 2_{10} & = & 1_{10} = 1_2 \\
z_{(3)} & & & = & 588_{10} & \beta_3 & & & & = 0_2 \\
z_{(4)} & & & = & 294_{10} & \beta_4 & & & & = 0_2 \\
z_{(5)} & & & = & 147_{10} & \beta_5 & & & & = 1_2 \\
z_{(6)} & & & = & 73_{10} & \beta_6 & & & & = 1_2 \\
z_{(7)} & & & = & 36_{10} & \beta_7 & & & & = 0_2 \\
z_{(8)} & & & = & 18_{10} & \beta_8 & & & & = 0_2 \\
z_{(9)} & & & = & 9_{10} & \beta_9 & & & & = 1_2 \\
z_{(10)} & & & = & 4_{10} & \beta_{10} & & & & = 0_2 \\
z_{(11)} & & & = & 2_{10} & \beta_{11} & & & & = 0_2 \\
z_{(12)} & & & = & 1_{10} & \beta_{12} & & & & = 1_2 \\
z_{(13)} & & & = & 0_{10} & & & & &
\end{array}
$$

Die Konvertierung vom Dezimal- in das Dualsystem erfolgt durch fortgesetzte Division der $z_{(i)}$ durch 2, bis das Divisionsergebnis Null erreicht ist. Die Koeffizienten im Dualsystem ergeben sich durch die Prüfung, ob das jeweilige Divionsergebnis gerade oder ungerade ist. Der Code im Zielsystem ergibt sich als Folge $<\beta_{n_b-1} \cdots \beta_1\beta_0>$. Das Verfahren läßt sich kompakt darstellen:

Beispiel: $49_{10} = 110001_2$

$$
\begin{array}{rrcl}
49 \text{ div } 2 = & 24 & \text{R} & 1 \\
24 \text{ div } 2 = & 12 & \text{R} & 0 \\
12 \text{ div } 2 = & 6 & \text{R} & 0 \\
6 \text{ div } 2 = & 3 & \text{R} & 0 \\
3 \text{ div } 2 = & 1 & \text{R} & 1 \\
1 \text{ div } 2 = & 0 & \text{R} & 1
\end{array}
$$

B) Multiplikationsmethode (Zielverfahren)

Die Quellziffern werden in ihrer Zielsystem-Darstellung entsprechend dem Hornerschema sukzessive von links nach rechts mit der Quellbasis (in der Zielsystemdarstel-

lung) multipliziert und die jeweils nächste Quellziffer der folgenden Stelle hinzuaddiert.
D.h., in der Hornerschema-Darstellung des Quellsystems werden die Klammerausdrücke von innen nach außen gleichzeitig in das Zielsystem konvertiert und aufgelöst.

Beispiel: Konvertierung BCDEF$_{16}$ in 773615_{10}

$$\begin{aligned} \text{BCDEF}_{16} = z_q &= (((\text{B}_{16} \cdot 10_{16} + \text{C}_{16}) \cdot 10_{16} + \text{D}_{16}) \cdot 10_{16} + \text{E}_{16}) \cdot 10_{16} + \text{F}_{16} \\ &= (((11_{10} \cdot 16_{10} + 12_{10}) \cdot 16_{10} + 13_{10}) \cdot 16_{10} + 14_{10}) \cdot 16_{10} + 15_{10} \\ &= 773615_{10} = z_p \end{aligned}$$

1.3.2.2 Darstellung ganzer Zahlen

Zur Darstellung ganzer Zahlen $z \in \mathbb{Z}$ durch ein Binärwort der Breite n Bit wird der darstellbare Bereich von 2^n Zahlen symmetrisch um die Null verteilt. Damit halbiert sich etwa für gegebenes n die maximale darstellbare natürliche Zahl $z_{\max} \in \mathbb{N}$, so daß der frei werdende Darstellungsbereich für die negativen Zahlen verwendet wird.
Negative Zahlen werden durch die *Komplementbildung*

$$K(z) := -z$$

für n-stellige Binärzahlen $z_2 \in \mathbb{B}^n$, $0 \leq |z_{10}| < N = 2^n$, gebildet. Es kommen zwei Methoden der Komplementbildung zur Anwendung:

1. Stellen- oder *Einerkomplement*

$$K_1(z) = (2^n - 1) - z$$

2. echtes oder *Zweierkomplement*

$$K_2(z) = 2^n - z.$$

Beide Methoden orientieren sich an der Zielstellung, die Architektur des Rechenwerkes einfach zu halten.
Eigenschaften des Einerkomplements:

- Zahlbereich: $|z| \leq 2^{n-1} - 1$ (symmetrisch)
- wegen
$$-z = (2^n - 1) - z = \underbrace{11\ldots1}_{n}{}_2 - z = (-0) - z = -z$$
erhält man das Einerkomplement durch stellenweises Invertieren aller Bits von z.

- es exisitieren zwei Darstellungen der Null:

$$\begin{aligned} -0_{10} &= 11\ldots1_2 = <1_{n-1}1_{n-2}\ldots1_0>_2 \\ +0_{10} &= 00\ldots0_2 = <0_{n-1}0_{n-2}\ldots0_0>_2 \end{aligned}$$

- Ursprung des Namens:

$$z + (-z) = \langle\alpha_{n-1}\alpha_{n-2}\ldots\alpha_0\rangle_2 \ + \ \langle\overline{\alpha_{n-1}}\,\overline{a_{n-2}}\ \ldots\overline{\alpha_0}\rangle_2 = 11\ldots1_2$$

  d.h. für die einzelnen Bits gilt

$$\alpha_i + \overline{\alpha_i} = 1$$

Eigenschaften des Zweierkomplements:

- Zahlbereich: $-2^{n-1} \leq z \leq 2^{n-1} - 1$ (asymmetrisch)

- das Zweierkomplement erhält man aus dem Einerkomplement durch Addieren einer Eins (modulo 2^n)

$$K_2(z) = K_1(z) + 1$$

- es existiert eine eindeutige Darstellung der Null:

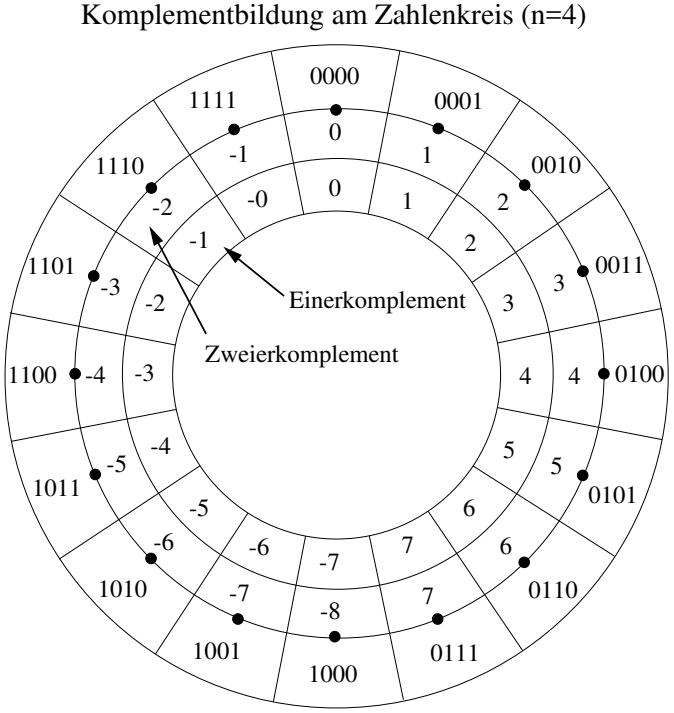
$$0_{10} = 00\ldots0_2 = <0_{n-1}0_{n-2}\ldots0_0>_2$$

  Deshalb wird das Zweierkomplement am häufigsten angewendet.

Es gilt folgender Zusammenhang zwischen den beiden Komplementbildungen:

$$\begin{aligned} K_2(z) = 2^n - z &= 2^n - \sum_{i=0}^{n-1}\alpha_i 2^i = \underbrace{1 + \sum_{i=0}^{n-1} 2^i}_{=2^n} - \sum_{i=0}^{n-1}\alpha_i 2^i \\ &= 1 + \sum_{i=0}^{n-1}(1-\alpha_i)2^i = 1 + K_1(z) \end{aligned}$$

Beispiel: Asymmetrische 2er-Komplement-Verteilung der Binärzahlen auf dem Zahlenkreis:



Die Komplementdarstellung einer Zahl ist mehrdeutig.

Beispiel:

$$\begin{aligned} n = 4 \quad z_{10} &\in \{0, \ldots, 7\},\ N = 2^4 = 16 \\ z_2 &\in \{0000, \ldots, 0111\} \end{aligned}$$

$5_{10} = 0101_2$
$K_2(5) = K_1(5) + 1 = 1010_2 + 1 = 1011_2 = 11_{10}$
die Dualdarstellung von -5 stimmt also mit der Darstellung von 16-5=11 überein!

Die Mehrdeutigkeit kann behoben werden durch folgende Feststellung:

$$\begin{aligned} \alpha_{n-1} = 0 \quad &\leftrightarrow \quad z \text{ ist positive Zahl} \\ \alpha_{n-1} = 1 \quad &\leftrightarrow \quad z \text{ ist negative Zahl} \end{aligned}$$

Deshalb kann die Zahl 11_{10} im vorigen Beispiel nicht mit 4 Bit dargestellt werden.

In jedem b-adischen Zahlensystem sind "$(b-1)$-Komplement" und "b-Komplement" berechenbar, vorausgesetzt, daß sie sich auf eine beliebige, aber feste Stellenzahl beziehen.

Die vier Grundrechenarten im b-adischen Zahlensystem werden prinzipiell wie im Dezimalsystem ausgeführt. Eine Besonderheit entsteht aus der Beschränkung auf eine feste Wortlänge.

Zu beachten ist, daß bei der stellenweisen Verrechnung (Addition, Multiplikation, Subtraktion wird auf die Addition mit dem Komplement zurückgeführt) eine Überschreitung des Wertebereiches der Koeffizienten (modulo b) oder der darstellbaren Zahlenbereiche (modulo N bei natürlichen Zahlen, modulo $2^{n-1}-1$ bei ganzen Zahlen) auftreten kann.

Addition zweier b-adischer Zahlen:
$z = x + y$ mit $x = \alpha_{n-1}\alpha_{n-2}\ldots\alpha_0,\ y = \beta_{n-1}\ldots\beta_0,\ z = \gamma_{n-1}\ldots\gamma_0,\ \alpha_i, \beta_i, \gamma_i \in \sigma_b$ für $i = 0, \ldots, n-1$. Die Addition erfolgt durch sukzessives Addieren der Koeffizienten α_i und β_i, beginnend bei $i = 0$. Bei der Addition der Koeffizienten an der Stelle i ist der ggf. an der Stelle $i-1$ aufgetretene Übertrag u_i zu berücksichtigen. Die Addition der Komponenten erfolgt nach dem Schema

$$\gamma_i = (\alpha_i + \beta_i + u_i) \bmod b,$$

für den Übertrag gilt

$$u_i = (\alpha_{i-1} + \beta_{i-1} + u_{i-1}) \operatorname{div} b, \text{ mit } u_0 = 0.$$

Beispiele:

Dualsystem:

	1	0	0	1	1	α_i
	0	1	0	1	1	β_i
+	0	0	1	1	0	u_i
	1	1	1	1	0	γ_i

Hexadezimalsystem:

	3	7	F
	0	C	9
+	1	1	0
	4	4	8

Definition: (Überlauf)

Der Überlauf stellt ein Überschreiten des darstellbaren Zahlenbereiches dar.

Bei der Addition natürlicher Zahlen wird ein Überlauf durch den Übertrag $u_n = 1$ erzeugt. Er wird gewöhnlich als Berechnungsfehler signalisiert.

1.3.2.3 Darstellung rationaler Zahlen

Rationale Zahlen sind Bruchzahlen. Eine rationale Zahl steht für die Gesamtheit aller durch Erweiterung eines Bruches erzeugbaren Brüche. Insbesondere kann eine rationale Zahl auch bezüglich einer Basis b als b-adische Bruchzahl geschrieben werden.
Bei der Festlegung der Länge n dieser Repräsentation ist zwangsläufig damit zu rechnen, daß rationale Zahlen nur approximiert werden können.

Beispiele:

$\left.\begin{array}{l} \frac{1}{12} = 0.08\overline{3}_{10} \\ \frac{82}{7} = 11.\overline{714285}_{10} \end{array}\right\}$ Dezimalbrüche

$\frac{1}{16} = 2_{10}^{-4} = 0.0001_2$ Dualbruch

$\frac{5}{16} = 5 \cdot 16_{10}^{-1} = 0.5_{16}$ Hexadezimalbruch

$\frac{5}{16} = 0.3125_{10} = 0.0101_2 = 0.24_8 = 0.5_{16}$

Definition: (b-adische Bruchzahl)

Eine b-adische rationale Zahl $z \in \sigma_b^s.\sigma_b^{-r} \subset \mathbb{Q},\ |z| < b^s$,

$$z = \sum_{i=0}^{s-1} \alpha_i b^i . \sum_{i=-r}^{-1} \alpha_i b^i,$$

$\alpha_i \in \sigma_b,$ entsteht aus der Konkatenation (angezeigt durch einen Punkt) zweier b-adischer Zahlen mit positivem bzw. negativem Exponenten. Hierbei steht der Punkt rechts neben dem Koeffizienten α_0.

Die Umkehrung der Divisionsmethode ergibt ein einfaches Verfahren zur Konvertierung b-adischer Bruchzahlen:

Der ganzzahlige Teil der Zahl wird unabhängig vom gebrochenen Teil konvertiert (z.B. Divisionsmethode).
Der gebrochene Teil der Zahl wird konvertiert, indem folgende Schritte iterativ ausgeführt werden:

1. Multiplikation des gebrochenen Zahlteils mit der Zielbasis

2. Auslesen des hierdurch entstehenden ganzzahligen Zahlteils und Verwendung als Codeziffer des gebrochenen Anteils im Zielcode (in direkter Reihenfolge)

Das Verfahren terminiert, wenn der gebrochene Zahlteil Null ergibt.

Beispiele:

a: $\frac{5}{16} \rightarrow (0.3125_{10} - 0) \cdot 2 \rightarrow (0.625_{10} - 0) \cdot 2 \rightarrow (1.25_{10} - 1) \cdot 2 \rightarrow (0.5_{10} - 0) \cdot 2 \rightarrow 1.0_{10}$

$\qquad 0. \qquad\qquad 0 \qquad\qquad 1 \qquad\qquad 0 \qquad\qquad 1$

$\rightarrow 0.3125_{10} = 0.0101_2$

b: $0.3125_{10} \rightarrow (0.3125_{10} - 0) \cdot 8 \rightarrow (2.5_{10} - 2_{10}) \cdot 8 \rightarrow 4.0_{10}$

$\qquad 0. \qquad\qquad 2 \qquad\qquad 4$

$\rightarrow 0.3125_{10} = 0.24_8$

c: $0.3125_{10} \rightarrow (0.3125_{10} - 0) \cdot 16 \rightarrow 5.0_{10}$

$\qquad 0. \qquad\qquad 5$

$\rightarrow 0.3125_{10} = 0.5_{16}$

d: $16.125_{10} = 10000.001_2$

Die Struktur der Darstellung einer rationalen Zahl muß vereinbart werden. Gebräuchlich sind zwei Darstellungsformen:

1. Festpunkt-Darstellung

2. Gleitpunkt-Darstellung

Definition: (Festpunkt-Darstellung)

Bei der Festpunkt-Darstellung einer b-adischen rationalen Zahl der Länge $n = s + r$ führt die Konvention

$$z = \sum_{i=0}^{s-1} \alpha_i b^i . \sum_{i=-r}^{-1} \alpha_i b^i = z' \cdot b^{-r}$$

mit

$$z' = \sum_{i=0}^{n-1} \alpha_i b^i$$

zu einer Standard-Darstellung durch das Paar (z', r).

Beispiele:

$16\frac{1}{8} = 16.125_{10}$

$$\begin{array}{llllll} b = 10, & n = 5 & , & r = 3 & : & z' = 16125_{10} \quad , z_{10} = 16.125_{10} \\ b = 2, & n = 8 & , & r = 3 & : & z' = 10000001_2 \quad , z_2 = 10000.001_2 \\ b = 8, & n = 3 & , & r = 1 & : & z' = 201_8 \quad , z_8 = 20.1_8 \\ b = 16, & n = 3 & , & r = 1 & : & z' = 102_{16} \quad , z_{16} = 10.2_{16} \end{array}$$

Die Festpunkt-Darstellung wird z.B. in der Prozeßsteuerung (hohe Genauigkeit, wenige komplexe arithmetische Operationen) und im Finanzwesen (Kontrolle der Rundungsfehler) angewandt.

Als Nachteil der Festpunkt-Darstellung ist der kleine darstellbare Zahlenbereich zu nennen. Insbesondere für wissenschaftliche Rechnungen wird ein wesentlich größerer darstellbarer Zahlenbereich bei einer relativ kleinen Zahl signifikanter Stellen benötigt.

Definition: (Gleitpunkt-Darstellung)

Die Gleitpunktdarstellung einer rationalen Zahl

$$z = \pm m \cdot B^e,$$

besteht aus Mantisse m, Basis der Darstellung B und Exponent e. Die Speicherung der Zahl unterscheidet die normalisierte und die denormalisierte Form.

Normalisiert: Die Mantisse wird derart verschoben, dass die erste führende 1 vor dem Komma steht. Die Mantisse repräsentiert somit eine Zahl $1 \leq m < 2$. Man bezeichnet sie auch als *Signifikant m*, während der Nachkommaanteil als *Fraktion M* bezeichnet wird. Die führende 1 (hidden one) wird aber nicht abgespeichert sondern nur bei der Rechnung berücksichtigt. So wird ein Bit gespart. Nur die Fraktion M wird im Speicher abgelegt und auch dort als Mantisse bezeichnet. Es gilt $m = 1 + M$. Dadurch kann allerdings die Null nicht dargestellt werden.

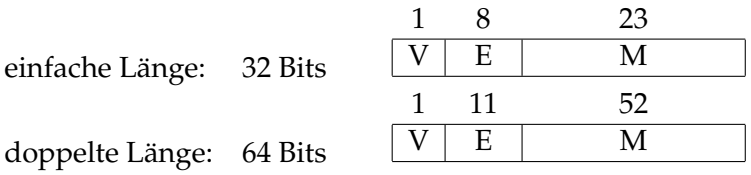
Denormalisiert: Der Signifikant liegt im Wertebereich

$$0 \leq M < 1$$

es wird keine führende 1 verwendet, somit ist $m = M$. Diese Form erlaubt die Darstellung von betragsmäßig sehr kleinen Zahlen.

Die Gleitpunkt-Darstellung wurde von K. Zuse als halblogarithmische Darstellung eingeführt. Rechnerintern kann als Basis B 2, 8 oder 16 verwendet werden. Der Anwender programmiert meist in der Basis 10, die Wandlung bezüglich der vom Rechner verwendeten Basis und die Normalisierung erfolgen intern.

Eine Möglichkeit zur rechnerinternen Darstellung von Gleitpunktzahlen ist im IEEE-Standard 754-1985 (ergänzt im Standard von 2008) festgelegt. Dort wird einerseits ein Standardfall festgelegt (normalisierter Fall, deckt die meisten Fälle ab), aber auch Spezialfälle sind berücksichtigt (denormalisiert, z.B. exakte Darstellung der Null, Zahlen sehr nahe bei Null, etc.).



IEEE Datenformate: 32 Bit (float, single):
1 Vorzeichenbit
8 Exponentenbits (MSB first)
23 Mantissenbits (MSB first)
Der Wert w einer in diesem Format dargestellten Zahl berechnet sich als:
$w = (-1)^V \cdot (1, M) \cdot 2^{E-127}$, falls $E > 0$ und $E < 255$ (normalisiert)
$w = (-1)^V \cdot (0, M) \cdot 2^{-126}$, falls $E = 0$ und $M \neq 0$ (denormalisiert)
$w = (-1)^V \cdot 0$, falls $E = 0$ und $M = 0$
$w = (-1)^V \cdot$ Infinity (∞), falls $E = 255$ und $M = 0$
w=NaN (not a number), falls $E = 255$ und $M \neq 0$
Darstellbarer Bereich: $\pm(1 - 2^{-24}) \cdot 2^1 28$, ca. $\pm 10^{38}$
Betragsmäßig kleinste Zahl: denormalisiert $\pm 2^- 149$, normalisiert $\pm 2^{-126}$

64 Bit (double):
1 Vorzeichenbit
11 Exponentenbits (MSB first)
52 Mantissenbits (MSB first)
Der Wert w einer in diesem Format dargestellten Zahl berechnet sich als:
$w = (-1)^V \cdot (1, M) \cdot 2^{E-1023}$, falls $E > 0$ und $E < 2047$
$w = (-1)^V \cdot (0, M) \cdot 2^{-1022}$, falls $E = 0$ und $M \neq 0$
$w = (-1)^V \cdot 0$, falls $E = 0$ und $M = 0$
$w = (-1)^V \cdot$ Infinity (∞), falls $E = 2047$ und $M = 0$
w=NaN (not a number), falls $E = 2047$ und $M \neq 0$
Darstellbarer Bereich: $\pm(1 - 2^{-53}) \cdot 2^1 024$, ca. $\pm 10^{300}$
Betragsmäßig kleinste Zahl: denormalisiert $\pm 2^- 1074$, normalisiert $\pm 2^{-1022}$

80 Bit (extended):
1 Vorzeichenbit
15 Exponentenbits (MSB first)
64 Mantissenbits (MSB first)
Der Wert w einer in diesem Format dargestellten Zahl berechnet sich als:
$w = (-1)^V \cdot (0, M) \cdot 2^{<E-16383}$, falls $E > 0$ und $E < 32767$
$w = (-1)^V \cdot$Infinity (∞), falls $E = 32767$ und $M = 0$
w=NaN (not a number), falls $E = 32767$ und $M \neq 0$
Darstellbarer Bereich ca. $\pm 10^{5000}$

Dabei ist zu beachten: Die Zahlen sind in normalisierter Form, d.h. vor dem Komma steht eine 1. Diese führende 1 wird nicht mit dargestellt (hidden one). Der Exponent ist mit einem Offset (Bias) versehen. Bei Single-Zahlen 127, bei Double-Zahlen 1023 und bei Extended-Zahlen 32767. D. h. die dargestellten Exponenten sind um diesen Bias zu

vermindern um ihre *normalen* Dualdarstellungen zu bekommen. Man erreicht dadurch, dass keine negativen Exponenten im IEEE-Format auftauchen können.

Folgende Sonderfälle werden unterschieden: Eine 0 wird dargestellt durch Vorzeichen=0, Mantisse = 000...0, Exponent = 000...0. Eine ∞ wird dargestellt durch Mantisse = 000...0, Exponent = 111...1. Eine nicht darstellbare Zahl (NaN, not a number) wird dargestellt als Mantisse $\neq 0$ und Exponent = 111...1. Wenn der Exponent 000...0 ist und die Mantisse $\neq$ 000...0, so wird die versteckte 1 nicht einkopiert. Dadurch werden Zahlen dargestellt, die sehr nah an Null sind.

Der gegenüber der Festpunktdarstellung erheblich höhere Darstellungsbereich wird mit der reduzierten Genauigkeit erkauft: 23 Bit entsprechen 7 signifikanten Dezimalstellen.

Echte reelle Zahlen (z.B. π, $\sqrt{2}$) werden nur als bestmögliche Approximation durch Gleitpunktzahlen repräsentiert.

Die kleinste / größte Gleitpunktzahl ist $z_{\min} = B^{e_{\min}-1}$ bzw. $z_{\max} = (1 - B^{-r})B^{e_{\max}}$. Zwischen diesen sind $N = 2(B-1)B^{r-1}(e_{\max}-e_{\min}+1)$ normalisierte Gleitpunktzahlen im Standardformat darstellbar. Im Falle einfacher Genauigkeit gilt $z_{\min} = 1.18 \cdot 10^{-38}$, $z_{\max} = 3.40 \cdot 10^{38}$ und $N = 2^{24} \cdot 254 \approx 4.26 \cdot 10^{9}$.

Die Verfügbarkeit so weniger Zahlen in einem derart großen Zahlenbereich legt nahe, daß Rundung eine wichtige Rolle spielt, damit Operationen über Gleitpunktzahlen selbst wieder eine Gleitpunktzahl ergeben. Wegen der enormen Konsequenzen für das Rechnen im Gleitpunktformat werden die Zusammenhänge kurz dargestellt. In der Numerischen Mathematik erfährt man hierzu mehr Einzelheiten.

1.3.2.4 Arithmetische Operationen mit Gleitpunktzahlen

Seien $z_1 = (m_1, e_1) = m_1 \cdot B^{e_1}$ und $z_2 = (m_2, e_2) = m_2 \cdot B^{e_2}$ zwei Gleitpunktzahlen. Die Rechenregeln der Exponentialdarstellung ergeben, dass bei Addition und Subtraktion zunächst die Exponenten angeglichen werden müssen, danach kann gerechnet werden:

Addition: $z_3' = (m_1 + m_2 \cdot B^{e_2-e_1}) \cdot B^{e_1}$

Subtraktion: $z_3' = (m_1 - m_2 \cdot B^{e_2-e_1}) \cdot B^{e_1}$

Die Multiplikation und Division sind einfacher durchzuführen:

Multiplikation: $z_3' = (M_1 \cdot M_2) \cdot B^{e_1+e_2}$

Division: $z_3' = (M_1/M_2) \cdot B^{e_1-e_2}$

In der normalisierten IEEE-Form muss zudem der Bias und die Normierung berücksichtigt werden. Für die Ausführung der Addition ergeben sich die drei Verarbeitungsschritte:

1. Exponentenausgleich mit Verschieben der Mantisse wegen der Wahl einer Referenz (hier: B^{e_1}). Achtung: bei der Berechnung der Exponenten ist auf den Bias zu achten, er muss zunächst bei der Berechnung nach dem Lesen aus dem IEEE-Format hinzugefügt werden (e= E+Bias), um den korrekten Exponenten zu bekommen, und vor der Abspeicherung im IEEE-Format wieder entfernt werden (E=e-Bias).
2. Addition der Mantissen
3. Normalisierung.

Beispiel: Gleitpunktzahl-Addition

$z_1 = 1.00144 \cdot 10^1$, $z_2 = -8.33333 \cdot 10^{-2}$, $B = 10$
Wähle Referenz 10^1 bzgl. Zahl $z_1, e_2 - e_1 = -3$
$z = (1.00144 - 8.3333 \cdot 10^{-3}) \cdot 10^1 = (1.00144 - 0.0083333)10^1$
$z = 0.9931067 \cdot 10^1$
Normalisierung: $z = 9.931067 \cdot 10^0$

1.3.2.5 Rundung von Gleitpunktzahlen

Das Ergebnis einer zweistelligen arithmetischen Operation über Gleitpunktzahlen ist im allgemeinen keine Gleitpunktzahl. Damit dieser Umstand überwunden werden kann, ist eine Änderung der arithmetischen Grundoperationen über Gleitpunktzahlen erforderlich, die zur Folge hat, daß wichtige Eigenschaften der Algebra reeller Zahlen verloren gehen.
Sei $\circ$ eine zweistellige arithmetische Operation reller Zahlen,

$$\circ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}; z_1^* \circ z_2^* = z_3^* \qquad \text{mit } z_1^*, z_2^*, z_3^* \in \mathbb{R}.$$

Sei $\mathbb{F} \subset \mathbb{R}$ die Menge der darstellbaren Gleitpunktzahlen. Dann gilt für die oben behandelten arithmetischen Operationen

$$\circ' : \mathbb{F} \times \mathbb{F} \to \mathbb{R}; z_1 \circ' z_2 = z_3^{*\prime} \qquad \text{mit } z_1, z_2 \in \mathbb{F}, z_3^{*\prime} \in \mathbb{R}.$$

Gesucht ist eine arithmetische Operation, deren Ergebnis ebenfalls eine Gleitpunktzahl ist.

$$\circ_{\text{rnd}} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}; z_1 \circ_{\text{rnd}} z_2 = z_3 \qquad \text{mit } z_1, z_2, z_3 \in \mathbb{F}.$$

Für jedes $z^* \in \mathbb{R}$ gilt

$$|z_{\text{lower}}| \leq |z^*| \leq |z_{\text{higher}}| \qquad \text{mit } z_{\text{lower}}, z_{\text{higher}} \in \mathbb{F}.$$

Also ist die Forderung durch Einführen einer Rundungsfunktion $\text{rnd}(z^*)$ erfüllbar, so daß

$$\text{rnd}(z^*) = z, \quad z \in \mathbb{F}$$

und

$$z_1 \circ_{\text{rnd}} z_2 = \text{rnd}(z_1 \circ z_2).$$

Folgende Varianten sind gebräuchlich:

1. *Abschneiden:* Runden auf absolut kleinere der benachbarten Gleitpunktzahlen.

$$\text{rnd}(z^*) = z^*_{\text{rnd}} = z_{\text{lower}}$$

2. *optimale Rundung:* Runden auf nächstgelegenen Wert $z \in \mathbb{F}$.

$$\text{rnd}(z^*) = z^*_{\text{rnd}} = \begin{cases} z_{\text{higher}} & \text{, wenn } |z_{\text{higher}} - z^*| \leq |z_{\text{lower}} - z^*| \\ z_{\text{lower}} & \text{, sonst} \end{cases}$$

Der *absolute Rundungsfehler* ist

$$\varepsilon(z) = \text{rnd}(z^*) - z^*$$

und der *relative Rundungsfehler* ist

$$\rho(z) = \frac{\varepsilon(z)}{z}.$$

Als Schranke des relativen Rundungsfehlers kann man die *relative Maschinengenauigkeit* $\Delta(B, r)$ angeben.

$$\Delta(B, r) = \begin{cases} B^{1-r} & \text{für Abschneiden} \\ 0.5 \cdot B^{1-r} & \text{für optimale Rundung.} \end{cases}$$

Deren Werte für einfache Genauigkeit sind

$$\Delta(2, 24) = \begin{cases} 2^{-23} \approx 1.19 \cdot 10^{-7} & \text{für Abschneiden} \\ 2^{-24} \approx 5.96 \cdot 10^{-8} & \text{für optimale Rundung.} \end{cases}$$

Dies entspricht also 7 bzw. 8 Dezimalstellen. Bei doppelter Genauigkeit erhöht sich die relative Maschinengenauigkeit auf 16 bzw. 17 Dezimalstellen.

Also gilt

$$z_1 \circ_{\text{rnd}} z_2 = (z_1 \circ z_2)(1 + \rho) \qquad \text{mit } |\rho| \leq \Delta.$$

Die Fehlerfortpflanzung kann zu erheblichem Gesamtfehler bei längeren Berechnungsfolgen führen. Aber bereits einfache zweistellige Operationen können zu absurden Ergebnissen (bzgl. der Algebra der reellen Zahlen) führen.

Bei Ausführung der Operation $\circ_{\text{rnd}}$ gehen wichtige Eigenschaften der Algebra der reellen Zahlen verloren:

1. Verlust der Assoziativität von Addition und Multiplikation

$$z_1 +_{\text{rnd}} (z_2 +_{\text{rnd}} z_3) \neq (z_1 +_{\text{rnd}} z_2) +_{\text{rnd}} z_3 \qquad \text{für } z_1, z_2, z_3 \in \mathbb{F}$$

$$z_1 \cdot_{\text{rnd}} (z_2 \cdot_{\text{rnd}} z_3) \neq (z_1 \cdot_{\text{rnd}} z_2) \cdot_{\text{rnd}} z_3$$

2. Verlust der Distributivität zwischen Addition und Multiplikation

$$z_1 \cdot_{\text{rnd}} (z_2 +_{\text{rnd}} z_3) \neq (z_1 \cdot_{\text{rnd}} z_2) +_{\text{rnd}} (z_1 \cdot_{\text{rnd}} z_3).$$

Beispiele: extreme Rundungsfehler

1. $z_1 = 10^{23}, z_2 = 2 \rightarrow z_3 = z_1 +_{\text{rnd}} z_2 = 10^{23}$
2. Auslöschung: Bei Subtraktion fast gleich großer Zahlen gehen die meisten relevanten Stellen verloren.
   $z_1 = 0.746841 \cdot 10^2, z_2 = 0.746832 \cdot 10^2$
   Annahme: letzte beide Ziffern wegen Rundung unzuverlässig
   $z_3 = z_1 -_{\text{rnd}} z_2 = 0.000009 \cdot 10^2 = 0.9 \cdot 10^{-3}$
   Also sind alle Ziffern der Mantisse des Ergebnisses unzuverlässig!

Literaturverzeichnis

[1] F.L. Bauer, G. Goos. *Informatik. Band1: Eine einführende Übersicht*. Springer, 1991.

[2] F.L. Bauer, G. Goos. *Informatik. Band2: Eine einführende Übersicht*. Springer, 1992.

[3] W. Bauer et al. *Studien- u. Forschungsführer Informatik*. Springer, 1989.

[4] R. Berghammer. *Informatik I und II. Vorlesungsskript 1995/96*. Inst. f. Informatik u. Prakt. Math., CAU Kiel, 1996.

[5] M. Broy. *Informatik I: Problemnahe Programmierung*. Springer, 1992.

[6] M. Broy. *Informatik II: Rechnerstrukturen und maschinennahe Programmierung*. Springer, 1993.

[7] M. Broy. *Informatik III: Systemstrukturen und systemnahe Programmierung*. Springer, 1994.

[8] W. Coy. *Informatik-Spektrum 12*, 1989.

[9] W. Coy. *Aufbau und Arbeitsweise von Rechenanlagen*. Vieweg, 1992.

[10] P.J. Denning et al. *IEEE Computer Magazine*, Febr. 1989.

[11] G. Goos. *Vorlesungen über Informatik. Band I: Grundlagen und funktionales Programmieren*. Springer, 1995.

[12] G. Goos. *Vorlesungen über Informatik. Band II: Objektorientiertes Programmieren und Algorithmen*. Springer, 1996.

[13] B.W. Kernighan, D.M. Ritchie. *Programmieren in C*. C. Hanser, 1990.

[14] H. Klaeren. *Vom Problem zum Programm*. Teubner, Stuttgart, 1991.

[15] W. Kluge. *Informatik II. Vorlesungsskript 1987*. Inst. f. Informatik. u. Prakt. Math., CAU Kiel, 1987.

[16] W. Kluge, C. Aßmann. *Informatik für Ingenieure II. Vorlesungsskript 1996*. Inst. f. Informatik u. Prakt. Math., CAU Kiel, 1996.

[17] B.O. Küppers. *Der Ursprung biologischer Information*. Piper, 1990.

[18] H. Liebig, T. Fink. *Rechnerorganisation*. Springer, 1993.

[19] H. Neumann, H.S. Stiehl. *Einführung in die Informatik für Mathematiker und Naturwissenschaftler. Vorlesungsskript 1992*. Fachbereich Informatik, Universität Hamburg, 1992.

[20] W. Oberschelp, G. Vossen. *Rechneraufbau und Rechnerstrukturen*. Oldenburg, 1994.

[21] R. P. Paul. *SPARC Architecture, Assembly Language Programming, & C*. Prentice Hall, 1994.

[22] P. Pepper. *Grundlagen der Informatik*. Oldenburg, 1995.

[23] *Spektrum der Wissenschaft*, Juli 1996.

[24] C.L. Tomdo, S.E. Gimpel. *Das C-Lösungsbuch*. C. Hanser, 1990. (zu Kernigham, Ritchie: Programmieren in C).

[25] C.F. v. Weizsäcker. *Die Einheit der Natur*. 1971.

[26] P.H. Winston. *Artificial Intelligence*. Addison-Wesley, 1992.

Index

Unterstrichene Seitenzahlen verweisen auf die Definitionen der zugehörigen Begriffe.