Informatik I

— Grundlagen der Informatik

Fachhochschule Niederrhein

WS 2005/06

Prof. Dr. Rethmann

Literatur

Informatik allgemein

- Rechenberg: Was ist Informatik?
  Carl Hanser Verlag
- Gumm, Sommer: Einführung in die Informatik
  Oldenbourg Verlag

Programmiersprache C

- Kernighan, Ritchie: Programmieren in C
  Carl Hanser Verlag
- Zeiner: Programmieren lernen mit C
  Carl Hanser Verlag

Literatur (2)

Algorithmen

- Wirth: Algorithmen und Datenstrukturen
  Teubner Verlag
- Sedgewick: Algorithms, Addison-Wesley
- Ottmann, Widmayer: Algorithmen und Datenstrukturen
  BI Wissenschaftsverlag
- Cormen, Leiserson, Rivest: Introduction to Algorithms
  MIT Press
- Aho, Hopcroft, Ullman: Datastructures and Algorithms
  Addison-Wesley

Einleitung

Informatik

Was ist das?

- Kunstwort aus **Infor**mation und Mathe**matik**
- Informatik ist eng mit Computern verknüpft: solange es keine Computer gab, gab es auch keine Informatik
- elektronische Rechenmaschine entstand um 1940

Ursprung

- Rechnen galt bis Anfang der Neuzeit als **Kunst**
- heute kann jeder die vier Grundrechenarten ausführen
  * mechanisch ausführbares Verfahren, dass nicht verstanden werden muss, um es anwenden zu können
  * kann einer Maschine übertragen werden kann

Informatik (2)

Algorithmus

- mechanisch ausführbares Rechenverfahren
- bildet den Kern der Informatik
- nach dem persischen Mathematiker Al-Chowarizmi

Beispiel: Euklidischer Algorithmus

- berechne größten gemeinsamen Teiler zweier natürlicher Zahlen p und q
- Euklid lebte etwa 300 v.Chr.

Informatik (3)

Euklidischer Algorithmus

1. Man dividiere p ganzzahlig durch q. Dabei erhält man den Rest r, der zwischen 0 und $q-1$ liegt.
2. Wenn $r = 0$ ist, dann ist q der ggT. Wenn $r \neq 0$ ist, dann benenne das bisherige q in p um, das bisherige r in q und wiederhole ab Schritt 1.

p	q	$r = p \bmod q$
216	378	216
378	216	162
216	162	54
162	54	0

Informatik (4)

```
#include <stdio.h>
int main(void) {
    int a, b, p, q, r;
    scanf("%d, %d", &a, &b);
    p = a;
    q = b;
    r = p % q;
    while (r != 0) {
        p = q;
        q = r;
        r = p % q;
    }
    printf("ggT(%d,%d) = %d\n", a, b, q);
    return 0;
}
```

Informatik (5)

- Mittels `#include <stdio.h>` wird eine Bibliothek bereitgestellt, die Funktionen zur Ein- und Ausgabe enthält.
- Der Start eines Programms besteht im Ausführen der Funktion `main`.
- `int a` deklariert eine Variable `a`, die ganze Zahlen aufnehmen kann. Alle in einem C-Programm benutzten Variablen müssen explizit deklariert werden, wobei der Typ und der Name der Variablen festgelegt werden.
- Die Funktion `scanf()` liest Werte von der Tastatur ein, `printf()` gibt eine Zeichenkette auf dem Bildschirm aus. Solche Standardfunktionen sind übersetzte Funktionen, die zur C-Implementierung gehören.
- Alle Anweisungen werden mit einem Semikolon beendet.

Informatik (6)

- Anweisungsfolgen werden mit geschweiften Klammern zusammengefasst, der geklammerte Block gilt als eine Anweisung.
- Mittels `while()` kann eine Anweisung mehrmals durchlaufen werden. Ist die gegebene Bedingung nicht erfüllt, wird die Schleife verlassen. Die Bedingung wird vor der ersten und nach jeder Ausführung der Anweisung geprüft.

Informatik (7)

Algorithmus

- mechanisches Verfahren, das aus mehreren Schritten besteht
- Schritte werden sequentiell ausgeführt, bis das Ergebnis gefunden ist (es gibt auch parallele Algorithmen)
- einzelne Abschnitte des Verfahrens können mehrfach durchlaufen werden (Iteration, Schleife)

Entwurf von Algorithmen

- finde eine Problemlösung
- formuliere sie in kleinen, elementaren Schritten

Informatik (8)

Es gibt sehr alte, immer noch aktuelle Algorithmen:

- je zwei natürliche Zahlen haben einen ggT $\rightarrow$ Euklid
- eine Matrix ist invertierbar $\iff$ die Determinante ist ungleich Null $\rightarrow$ Gaußsches Eliminationsverfahren
- erst der Computer ermöglicht es, auch komplizierte Algorithmen mit tausenden von Schritten auszuführen

Technische Informatik

Aufbau und Konstruktion von Computern.

- Rechnerarchitektur
- Rechnerhardware
- Mikroprozessortechnik
- Rechnernetze

Praktische Informatik

Entwicklung und Erweiterung der Rechnereigenschaften. Programmierung und Nutzung von Computern.

- Betriebssysteme
- Benutzerschnittstellen
- Informationssysteme (Datenbanken)
- Programmiersprachen und Übersetzer
- Softwaretechnologie

Theoretische Informatik

Formale mathematische Grundlagen.

- Formale Sprachen
- Automatentheorie
- Berechenbarkeit
- Komplexitätstheorie
- Algorithmen & Datenstrukturen

Angewandte Informatik

Lösen spezieller Probleme in Anwendungsbereichen mittels Computer. Der Rechner wird als Werkzeug eingesetzt.

- Computergrafik
- Digitale Signalverarbeitung (Bild-/Spracherkennung)
- Simulation und Modellierung
- Künstliche Intelligenz
- Textverarbeitung

Anmerkungen

Praktische und Angewandte Informatik sind mitunter nur schwer abzugrenzen, weil in beiden die Programmierung im Mittelpunkt steht.

In letzter Zeit wird dies durch eine andere Art der Einteilung ergänzt: Wirtschafts-, Bio-, Geoinformatik usw.

Informatik ist nicht gleichzusetzen mit Programmierung.

Man lernt Informatik nicht aus Büchern wie

- „Word 7.0 für Fortgeschrittene“ oder
- „Die besten Tipps zum Surfen im Internet“ oder
- „Programmieren in C++“.

Inhalte der Vorlesung

- Zahlendarstellung im Rechner
  - Darstellung ganzer Zahlen
  - Darstellung von Gleitkommazahlen
  - Rechnerarithmetik
- Die Programmiersprache C
  - Top-down Entwicklung
  - Grundelemente der Sprache
  - Strukturierte Programmierung
  - Standardbibliotheken
  - Modulare Programmierung

Inhalte der Vorlesung (2)

- Algorithmen und Datenstrukturen
  - Aufwandsabschätzungen
  - Sortieralgorithmen
  - Suchalgorithmen
  - Graphalgorithmen
  - Hash-Verfahren
- Diverses
  - Formale Sprachen
  - Programmiersprachen
  - Modellbildung und Spezifikation
  - Rechnerarchitektur

Zahlendarstellung im Rechner

Allgemeines

Alphabet: In Daten vorkommende Zeichen gehören immer einer bestimmten Zeichenmenge an:

- Zahlen → Ziffern, Dezimalpunkt und Vorzeichen
- Texte → Buchstaben, Ziffern und Satzzeichen

Die einzelnen Zeichen werden **Symbole** genannt.

Das klassische Alphabet der indogermanischen Kultur ist

$$\Sigma_{10} = \{0,1,2,3,4,5,6,7,8,9\}$$

Wörter mit fester Länge werden durch Einfügen führender Nullen erreicht: 0123, 0974, 0007.

Allgemeines (2)

Weitere Alphabete:

$$\begin{aligned} \Sigma_2 &= \{0,1\} && \text{dual, binär} \\ \Sigma_8 &= \{0,1,2,3,\ldots,7\} && \text{oktal} \\ \Sigma_{16} &= \{0,1,2,3,\ldots,9,A,B,C,D,E,F\} && \text{hexadezimal} \end{aligned}$$

Σ_{16} ist strenggenommen das Alphabet $\{0,1,2,\ldots,14,15\}$. Anstelle der „Ziffern" $10, 11, \ldots$ werden generell neue Symbole $A, B, \ldots$ verwendet.

In der Informatik häufig zu finden: Basen $b = 2, 8, 16$

Von geringer Bedeutung: Basen $b = 12$ (Dutzend, Gros), $b = 20$ (franz. vingt) und $b = 60$ (Zeitrechnung).

Codierung

Die Symbole aller denkbaren Alphabete lassen sich durch **Gruppen von Binärzeichen** ausdrücken.

Beispiel: Das deutsche Alphabet der Kleinbuchstaben kann wie folgt dargestellt werden:

00000	a	00100	e	01000	i	
00001	b	00101	f	01001	j	
00010	c	00110	g	01010	k	
00011	d	00111	h	01011	l	...

Wichtig: Mit Gruppen aus n Binärzeichen lassen sich 2^n Symbole codieren.

Codierung (2)

Ein **Code** ist die Zuordnung einer Menge von Zeichenfolgen zu einer anderen Menge von Zeichenfolgen.

Die Zuordnung (besser Abbildung) erfolgt oft durch eine Tabelle, der **Codetabelle**.

Ziffern, Klein- und Großbuchstaben, Umlaute, Satzzeichen und einige mathematische Zeichen können mit 8 Binärzeichen codiert werden.

International: **ASCII-Code** (American Standard Code for Information Interchange)

ASCII-Tabelle (Auszug)

Oct	Dec	Hex	Char	Oct	Dec	Hex	Char
054	44	2C	,	070	56	38	8
055	45	2D	-	071	57	39	9
056	46	2E	.	072	58	3A	:
057	47	2F	/	073	59	3B	;
060	48	30	0	074	60	3C	$<$
061	49	31	1	075	61	3D	$=$
062	50	32	2	076	62	3E	$>$
063	51	33	3	077	63	3F	?
064	52	34	4	100	64	40	@
065	53	35	5	101	65	41	A
066	54	36	6	102	66	42	B
067	55	37	7	103	67	43	C

b-adische Darstellung natürlicher Zahlen

Satz: Sei $b \in \mathbb{N}$ und $b > 1$. Dann ist jede ganze Zahl z mit $0 \leq z \leq b^n - 1$ und $n \in \mathbb{N}$ eindeutig als Wort $z_{n-1}z_{n-2}\ldots z_0$ der Länge n über Σ_b dargestellt durch:

$$z = z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + z_1 \cdot b^1 + z_0 \cdot b^0$$

Vereinbarungen:

- In der Ziffernschreibweise geben wir in der Regel die Basis der Zahlendarstellung explizit an, außer wenn eindeutig klar ist, welche Basis gemeint ist.
- Die Basis selbst wird immer dezimal angegeben.

b-adische Darstellung natürlicher Zahlen (2)

Beispiel: Sei $b = 10$ (Dezimalsystem).
Die eindeutige Darstellung von $z = 4711$ lautet

$$\begin{aligned} z &= 4 \cdot 10^3 + 7 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0 \\ &= 4 \cdot 1000 + 7 \cdot 100 + 1 \cdot 10 + 1 \cdot 1 \\ &= 4000 + 700 + 10 + 1 \end{aligned}$$

und in Ziffernschreibweise $(4711)_{10}$.

Beispiel: Sei $b = 2$ (Dualsystem).
Die eindeutige Darstellung von $z = 42$ lautet

$$\begin{aligned} z &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 1 \cdot 32 + 1 \cdot 8 + 1 \cdot 2 \end{aligned}$$

und in Ziffernschreibweise $(101010)_2$.

Darstellung reeller Zahlen

Festpunktdarstellung: Das Komma wird an beliebiger, aber fester Stelle angenommen.

Satz: Sei $b \in \mathbb{N}$ und $x_{n-1}x_{n-2}\ldots x_0x_{-1}x_{-2}\ldots x_{-m}$ eine $n+m$-stellige Zahl mit $x_i \in \Sigma_b$ für $i = -m, -m+1, \ldots, n$, wobei das Komma rechts von x_0 angenommen wird. Dann stellt obiges Wort folgende Zahl dar:

$$\begin{aligned} x = {} & x_{n-1} \cdot b^{n-1} + x_{n-2} \cdot b^{n-2} + \ldots + x_1 \cdot b^1 + x_0 \cdot b^0 \\ & + x_{-1} \cdot b^{-1} + \ldots + x_{-m} \cdot b^{-m} \end{aligned}$$

Beispiel: 000010111011 ist bei 8 Vor- und 4 Nachkommastellen die Darstellung von

$$\begin{array}{ccccccccccccc} 2^3 & + & 2^1 & + & 2^0 & + & 2^{-1} & + & 2^{-3} & + & 2^{-4} & = & \\ 8 & + & 2 & + & 1 & + & \frac{1}{2} & + & \frac{1}{8} & + & \frac{1}{16} & = & 11.6875 \end{array}$$

Darstellung reeller Zahlen (2)

Frage: Zahlen sind Zeichenketten. Warum codiert man Zahlen nicht im ASCII?

Antwort:

- **hoher Speicherbedarf:** Jede Ziffer benötigt 8 Zeichen zur Darstellung. **Beispiel:** Darstellung von $(123)_{10}$

  ASCII: 00110001 00110010 00110011
  Dual: 01111011

- **komplizierte Arithmetik:** ASCII-Werte können nicht einfach summiert werden!

$$\begin{array}{r} 8 \\ +\ 9 \\ \hline 17 \end{array} \qquad \begin{array}{rcl} 00111000 & \widehat{=} & 8 \\ +\ 00111001 & \widehat{=} & 9 \\ \hline 01110001 & \widehat{=} & \text{q} \end{array}$$

Zahlenumwandlung
Dezimalsystem → andere Systeme

Beispiel: $(935.421875)_{10} = (3A7.6C)_{16}$.

1. Zahl aufteilen in Vor- und Nachkommateil.
2. Vorkommateil durch fortgesetzte Division umwandeln.

$$\begin{array}{rcrcrlcl} 935 & : & 16 & = & 58 & \text{Rest } 7 & \widehat{=} & 7 \\ 58 & : & 16 & = & 3 & \text{Rest } 10 & \widehat{=} & A \\ 3 & : & 16 & = & 0 & \text{Rest } 3 & \widehat{=} & 3 \end{array}$$

Die jeweiligen Divisionsreste ergeben von unten nach oben gelesen den Vorkommateil der gesuchten Zahl in der anderen Basis.

Zahlenumwandlung
Dezimalsystem → andere Systeme (2)

3. Nachkommateil durch fortgesetzte Multiplikation umwandeln.

$$\begin{array}{lcccrcll} 0.421875 & \cdot & 16 & = & 6 & + & 0.75 & \rightarrow 6 \\ 0.75 & \cdot & 16 & = & 12 & + & 0 & \rightarrow C \end{array}$$

Die jeweiligen ganzen Teile ergeben von oben nach unten gelesen den Nachkommateil der gesuchten Zahl in der anderen Basis.

Korrektheit des Verfahrens: ...

Zahlenumwandlung
Dezimalsystem → andere Systeme (3)

Beispiel: $(978.127685546875)_{10} = (3D2.20B)_{16}$

Vorkommateil:

$$\begin{array}{rcrcrlcl} 978 & : & 16 & = & 61 & \text{Rest } 2 & \widehat{=} & 2 \\ 61 & : & 16 & = & 3 & \text{Rest } 13 & \widehat{=} & D \\ 3 & : & 16 & = & 0 & \text{Rest } 3 & \widehat{=} & 3 \end{array}$$

Nachkommateil:

$$\begin{array}{lcccrcll} 0.127685546875 & \cdot & 16 & = & 2 & + & 0.04296875 & \rightarrow 2 \\ 0.04296875 & \cdot & 16 & = & 0 & + & 0.6875 & \rightarrow 0 \\ 0.6875 & \cdot & 16 & = & 11 & + & 0 & \rightarrow B \end{array}$$

Zahlenumwandlung
Dezimalsystem → andere Systeme (4)

Beispiel: $(122.1)_{10} = (172.0\overline{6314})_8$
Vorkommateil:

$$\begin{array}{rcrcrl} 122 & : & 8 & = & 15 & \text{Rest } 2 \\ 15 & : & 8 & = & 1 & \text{Rest } 7 \\ 1 & : & 8 & = & 0 & \text{Rest } 1 \end{array}$$

Nachkommateil:

$$\begin{array}{lcccccl} 0.1 & \cdot & 8 & = & 0 & + & 0.8 \\ 0.8 & \cdot & 8 & = & 6 & + & 0.4 \\ 0.4 & \cdot & 8 & = & 3 & + & 0.2 \\ 0.2 & \cdot & 8 & = & 1 & + & 0.6 \\ 0.6 & \cdot & 8 & = & 4 & + & 0.8 \ \ldots \end{array}$$

Periodische Dualbrüche

Bei der Umwandlung vom Dezimal- ins Dualsystem ergeben sich oft periodische Dualbrüche: $(0.1)_{10} = (0.0\overline{0011})_2$.

Im Rechner:

- Länge der Zahlen ist beschränkt → periodische Dualbrüche können nur näherungsweise dargestellt werden
- bei n Nachkommastellen ist der durch das Weglassen weiterer Dualstellen entstehende Fehler im Mittel die Hälfte der letzten dargestellten Ziffer → $0.5 \cdot 2^{-n}$

Zahlenumwandlung
beliebige Systeme → Dezimalsystem

Berechnen der b-adischen Darstellung: **Horner-Schema**

Beispiel: $(63D2)_{16} = (25554)_{10}$

$$\begin{aligned} (63D2)_{16} &= 6 \cdot 16^3 + 3 \cdot 16^2 + 13 \cdot 16^1 + 2 \cdot 16^0 \\ &= (6 \cdot 16 + 3) \cdot 16^2 + 13 \cdot 16^1 + 2 \cdot 16^0 \\ &= [(6 \cdot 16 + 3) \cdot 16 + 13] \cdot 16 + 2 \end{aligned}$$

Durch Anwenden des Horner-Algorithmus reduziert sich die Anzahl der durchzuführenden Multiplikationen.

Beispiel: $(1736)_8 = (990)_{10}$

$$\begin{aligned} (1736)_8 &= 1 \cdot 8^3 + 7 \cdot 8^2 + 3 \cdot 8^1 + 6 \cdot 8^0 \\ &= [(1 \cdot 8 + 7) \cdot 8 + 3] \cdot 8 + 6 \end{aligned}$$

Umwandlung artverwandter Systeme

Bei zwei Basen $b, b' \in \mathbb{N}$ mit $b' = b^n$ für ein $n \in \mathbb{N}$ kann die Zahlenumwandlung vereinfacht werden.

Beispiel: $(21121,1)_3 = (247,3)_9$
Die Ziffern der Zahl $(21121,1)_3$ werden von rechts nach links paarweise zusammengefasst, da $9 = 3^2$:

$$\begin{aligned} (02)_3 &= (2)_9 \\ (11)_3 &= (4)_9 \\ (21)_3 &= (7)_9 \\ (10)_3 &= (3)_9 \end{aligned}$$

Umwandlung artverwandter Systeme (2)

Beispiel: $(32132)_4 = (39E)_{16}$
Die Ziffern der Zahl $(32132)_4$ werden von rechts nach links paarweise zusammengefasst, da $16 = 4^2$.

$$\begin{aligned}(03)_4 &= (3)_{16}\\ (21)_4 &= (9)_{16}\\ (32)_4 &= (E)_{16}\end{aligned}$$

Beispiel: $(2A7)_{16} = (0010\ 1010\ 0111)_2$
Die Ziffern der Zahl $(2A7)_{16}$ werden jeweils als 4-stellige Dualzahl geschrieben, da $16 = 2^4$:

$$\begin{aligned}(2)_{16} &= (0010)_2\\ (A)_{16} &= (1010)_2\\ (7)_{16} &= (0111)_2\end{aligned}$$

Arithmetik: Addition

Addition einstelliger Dualzahlen:

$$\begin{array}{r}0\\+\,0\\\hline 0\end{array}\qquad\begin{array}{r}0\\+\,1\\\hline 1\end{array}\qquad\begin{array}{r}1\\+\,0\\\hline 1\end{array}\qquad\begin{array}{r}1\\+\,1\\\hline 10\end{array}$$

Beispiel: Addition mehrstelliger Zahlen

$$\text{Dezimal}\quad\begin{array}{r}37\\+49\\\hline 86\end{array}\qquad\text{Dual}\quad\begin{array}{r}00100101\\+00110001\\\hline 01010110\end{array}$$

Arithmetik: Multiplikation

Multiplikation einstelliger Dualzahlen:

$$\begin{array}{ccccc}0 & * & 0 & = & 0\\1 & * & 0 & = & 0\\0 & * & 1 & = & 0\\1 & * & 1 & = & 1\end{array}$$

Beispiel: Multiplikation mehrstelliger Zahlen

$$\text{Dezimal}\quad\begin{array}{r}37\cdot 21\\\hline 37\\740\\\hline 777\end{array}\qquad\text{Dual}\quad\begin{array}{r}00100101\cdot 00010101\\\hline 100101\\10010100\\1001010000\\\hline 1100001001\end{array}$$

Arithmetik: Division

Dezimal

$$\begin{array}{lll}5\ 0\ 4 & : 42 = 12\\ \underline{4\ 2} & \\ \ \ \ 8\ 4 & \\ \ \ \ \underline{8\ 4} & \\ \ \ \ \ \ \ 0 & \end{array}$$

Dual

$$\begin{array}{l}1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ : 101010 = 1100\\ \underline{1\ 0\ 1\ 0\ 1\ 0}\\ \ \ 1\ 0\ 1\ 0\ 1\ 0\\ \ \ \underline{1\ 0\ 1\ 0\ 1\ 0}\\ \ \ \ \ \ \ \ \ \ \ \ \ 0\end{array}$$

Zahlendarstellung im Rechner

Codierung ganzer Zahlen

- Vorzeichen und Betrag
- Einer-Komplement
- Zweier-Komplement
- Exzess-Darstellung

Vorzeichen- und Betragdarstellung

Voraussetzung: feste Wortlänge

ganz links stehendes Bit: Vorzeichen ($0/1 \hat{=} +/-$)
restliche Bits: Darstellung des Betrags

Beispiel: Bei einer Wortlänge von 16 Bit können Zahlen zwischen $+2^{15}$ und -2^{15} dargestellt werden.

$$\begin{aligned} +92 &= 0000000001011100 \\ -92 &= 1000000001011100 \end{aligned}$$

Nachteile:

- zwei Nullen: -0 und +0
- Addierer und Subtrahierer nötig
- Entscheidungslogik nötig: addieren oder subtrahieren?

Vorzeichen- und Betragdarstellung (2)

Entscheidungslogik: vier Fälle sind zu unterscheiden:

Operanden	Operation	Beispiel
$+x, +y$	$x + y$	5 + 2 = 5 + 2
$-x, -y$	$-(x + y)$	-5 - 2 = -(5 + 2)
$+x, -y$ mit $\|x\| \geq \|y\|$	$x - y$	5 - 2 = 5 - 2
$-x, +y$ mit $\|y\| \geq \|x\|$	$y - x$	-2 + 5 = 5 - 2
$+x, -y$ mit $\|x\| < \|y\|$	$-(y - x)$	2 - 5 = -(5-2)
$-x, +y$ mit $\|y\| < \|x\|$	$-(x - y)$	-5 + 2 = -(5-2)

Wir wollen die Subtraktion vermeiden, indem wir die Subtraktion auf die Addition zurückführen.

Komplementdarstellung

Sei $x = x_{n-1}x_{n-2}\dots x_0$ eine n-stellige Dualzahl. Sei

$$\overline{x_i} = \begin{cases} 1 & \text{falls } x_i = 0 \\ 0 & \text{falls } x_i = 1 \end{cases}$$

Einer-Komplement: komplementiere die einzelnen Ziffern der Dualzahl:

$$\overline{x}^1 = \overline{x_{n-1}}\ \overline{x_{n-2}} \dots \overline{x_0}$$

Zweier-Komplement: bilde erst das Einer-Komplement und addiere dann eine Eins (modulo 2^n):

$$\overline{x}^2 = \overline{x_{n-1}}\ \overline{x_{n-2}} \dots \overline{x_0} \ + \ 1 \text{ (modulo } 2^n).$$

Komplementdarstellung (2)

Beispiel:

$$\begin{aligned} x &= 10110100 \\ \overline{x}^1 &= 01001011 \\ \overline{x}^2 &= 01001100 \end{aligned}$$

Beispiel:

$$\begin{aligned} x &= 0001101101011000 \\ \overline{x}^1 &= 1110010010100111 \\ \overline{x}^2 &= 1110010010101000 \end{aligned}$$

Komplementdarstellung (3)

Für jede b-adische Zahlendarstellung kann das $(b-1)$- und das b-Komplement definiert werden.

Eine Komplementdarstellung ist auf eine beliebige, aber **fest vorgegebene Stellenzahl** bezogen.

Komplementdarstellung: eine negative Zahl $-x$ wird durch die Differenz $N-x$ dargestellt (N = Anzahl darstellbarer Zahlen)

Beispiel: Für $b = 10$ und $n = 3$ gilt $N = 10^3 = 1000$. Im Zehner-Komplement: -23 entspricht $N - 23 = 977$

$$\begin{array}{r} 568 \\ -\ 23 \\ \hline 545 \end{array} \qquad \begin{array}{r} 568 \\ +\ 977 \\ \hline 1545 \end{array}$$

Komplementdarstellung (4)

Dualzahl	Einer-Komplement	Zweier-Komplement
000 = 0	000 = 0	000 = 0
001 = 1	001 = 1	001 = 1
010 = 2	010 = 2	010 = 2
011 = 3	011 = 3	011 = 3
100 = 4	100 = -3	100 = -4
101 = 5	101 = -2	101 = -3
110 = 6	110 = -1	110 = -2
111 = 7	111 = -0	111 = -1

Subtraktion im Einer-Komplement

Voraussetzung: x und y zwei n-stellige Dualzahlen

1. anstelle von $x - y$ berechne $x + \overline{y}^1$
2. ggf. Übertrag zur niederwertigsten Stelle addieren

Beispiel: Sei $n = 8$. Aus der Rechnung

$$\begin{array}{rrcr} x & 01110111 & \widehat{=} & 119 \\ -y & -00111011 & \widehat{=} & 59 \\ \hline & 00111100 & \widehat{=} & 60 \end{array}$$

wird im Einer-Komplement

$$\begin{array}{rr} x & 01110111 \\ +\overline{y}^1 & +11000100 \\ \hline & 100111011 \end{array} \qquad \begin{array}{r} 00111011 \\ +1 \\ \hline 00111100 \end{array}$$

Subtraktion im Einer-Komplement (2)

Beispiel: Aus der Rechnung

$$\begin{array}{rr} 27 & 00011011 \\ -38 & -00100110 \\ \hline -11 & 00001011 \end{array}$$

wird im Einer-Komplement

$$\begin{array}{rcr} 00011011 & \widehat{=} & 27 \\ +11011001 & \widehat{=} & -38 \\ \hline 11110100 & \widehat{=} & -11 \end{array}$$

Subtraktion im Zweier-Komplement

Voraussetzung: x und y zwei n-stellige Dualzahlen

1. anstelle von $x - y$ berechne $x + \overline{y}^2$
2. ignoriere eventuell auftretenden Übertrag

Beispiel: Sei $n = 8$. Aus der Rechnung

$$\begin{array}{rrcr} x & 01110111 & \widehat{=} & 119 \\ -y & -00111011 & \widehat{=} & 59 \\ \hline & 00111100 & \widehat{=} & 60 \end{array}$$

wird im Zweier-Komplement

$$\begin{array}{rr} x & 01110111 \\ +\overline{y}^2 & +11000101 \\ \hline & 100111100 \end{array}$$

Subtraktion im Zweier-Komplement (2)

Beispiel: Aus der Rechnung

$$\begin{array}{rr} 27 & 00011011 \\ -38 & -00100110 \\ \hline -11 & 00001011 \end{array}$$

wird im Zweier-Komplement

$$\begin{array}{rcr} 00011011 & \widehat{=} & 27 \\ +11011010 & \widehat{=} & -38 \\ \hline 11110101 & \widehat{=} & -11 \end{array}$$

Exzess-Darstellung

- zur Darstellung des Exponenten bei Gleitpunktzahlen
- zum Wert einer Zahl x wird eine positive Zahl q addiert, so dass das Ergebnis nicht negativ ist
- Exzess q gleich Betrag der größten negativen Zahl

Beispiel: Anzahl Bits gleich 4 $\Rightarrow q = 8$

x	$x+q$	Code	x	$x+q$	Code	x	$x+q$	Code
-8	0	0000	-3	5	0101	2	10	1010
-7	1	0001	-2	6	0110	3	11	1011
-6	2	0010	-1	7	0111	4	12	1100
-5	3	0011	0	8	1000	..	..	..
-4	4	0100	1	9	1001	7	15	1111

Darstellung von Gleitkommazahlen

Operationen auf Festkomma-Zahlen: Das Komma muss bei allen Operanden an der gleichen Stelle stehen.

Beispiel: Addition der Zahlen 101.01 und 11.101:

$$\begin{array}{rrcl} & 101 & . & 01 \\ + & 11 & . & 101 \\ \hline & 1000 & . & 111 \end{array}$$

Also: Zahlen müssen eventuell transformiert werden
$\Rightarrow$ signifikante Stellen gehen verloren

Beispiel: Bei 4 Vor- und 4 Nachkommastellen muss die Zahl 0.000111101 durch 0000.0001 abgerundet dargestellt werden $\longrightarrow$ fünf signifikante Stellen gehen verloren

Halblogarithmische Darstellung

Bei der **Gleitpunktdarstellung** wird jede Zahl z dargestellt in der Form:

$$z = \pm m \cdot b^{\pm d}$$

m : **Mantisse**
d : **Exponent**
b : **Basis des Exponenten**

Beispiele:

$$\begin{array}{lcll} 3.14159 & = & 0.314159 & \cdot\ 10^{1} \\ 0.000021 & = & 0.21 & \cdot\ 10^{-4} \\ 12340000 & = & 0.1234 & \cdot\ 10^{8} \end{array}$$

Halblogarithmische Darstellung (2)

Die halblogarithmische Darstellung ist nicht eindeutig:

$$\begin{array}{rcll} 3.14159 & = & 0.0314159 & \cdot\ 10^{2} \\ & = & 0.314159 & \cdot\ 10^{1} \\ & = & 31.4159 & \cdot\ 10^{-1} \\ & = & 314.159 & \cdot\ 10^{-2} \end{array}$$

Mehrdeutigkeiten vermeiden $\longrightarrow$ normalisierte Darstellung

Eine Gleitkommazahl der Form $\pm m \cdot b^{\pm d}$ heißt **normalisiert**, wenn gilt:

$$\frac{1}{b} \leq |m| < 1$$

Halblogarithmische Darstellung (3)

Beispiele:

- $(0.000011101)_2 \longrightarrow (0.11101)_2 \cdot 2^{-4}$
- $(1001.101)_2 \cdot 2^{10} \longrightarrow (0.1001101)_2 \cdot 2^{14}$
- $3.14159 \longrightarrow 0.314159 \cdot 10^{1}$
- $47.11 \cdot 10^{2} \longrightarrow 0.4711 \cdot 10^{4}$
- $0.0815 \cdot 10^{-3} \longrightarrow 0.815 \cdot 10^{-4}$

Gleitpunktzahlen im Rechner

Speichern einer Gleitkommazahl im Computer: aufteilen der 32 Bit wie folgt:

1. **Mantisse:** 23 Bit für den Betrag plus ein Bit für das Vorzeichen der Mantisse (Vorzeichen-/Betragdarstellung)
2. **Exponent:** 8 Bit (Zweier-Komplement Darstellung)
3. erste Stelle der Mantisse ist immer Null und wird in der Rechnerdarstellung ignoriert

analog für 64 Bit-Darstellung ...

Gleitpunktzahlen im Rechner (2)

Beispiel:

$$\begin{aligned}(5031.1875)_{10} &= (1001110100111.0011)_2 \cdot 2^0 \\ &= (0.10011101001110011)_2 \cdot 2^{13} \\ &= (0.10011101001110011)_2 \cdot 2^{(00001101)_2}\end{aligned}$$

Vorzeichen	:	0
Mantisse	:	10011101001110011000000
Exponent	:	00001101

Gleitpunktzahlen im Rechner (3)

Beispiel:

$$\begin{aligned}(-0.078125)_{10} &= (-0.000101)_2 \cdot 2^0 \\ &= (-0.101)_2 \cdot 2^{-3} \\ &= (-0.101)_2 \cdot 2^{(11111101)_2}\end{aligned}$$

Vorzeichen	:	1
Mantisse	:	10100000000000000000000
Exponent	:	11111101

Gleitpunktzahlen im Rechner (4)

Null im darstellbaren Bereich nicht enthalten $\Rightarrow$ abweichende Darstellung: Vorzeichen 0, Mantisse 0, Exponent 0

Ist die Basis b des Exponenten 2, so ist das erste Bit der Mantisse immer 1:

- erstes Bit der Mantisse nicht speichern (**hidden bit**)
- Verwechselung zwischen $\frac{1}{2}$ und 0 ausschließen

Gleitkommazahlen: Einbußen hinsichtlich Genauigkeit

- größte Gleitkommazahl bei 32 Bit: etwa 2^{127}
- größte Zahl in Dualdarstellung bei 32 Bit: $2^{31} - 1$

Gleitpunktzahlen: Arithmetik

Seien $x = m_x \cdot 2^{d_x}$ und $y = m_y \cdot 2^{d_y}$.

Multiplikation:
Mantissen multiplizieren, Exponenten addieren

$$x \cdot y = (m_x \cdot m_y) \cdot 2^{d_x + d_y}$$

Division:
Mantissen dividieren, Exponenten subtrahieren

$$x : y = (m_x : m_y) \cdot 2^{d_x - d_y}$$

Gleitpunktzahlen: Arithmetik (2)

Beispiel: 12.18 · 3.7

$$\begin{aligned} 0.1218 \cdot 10^2 \quad \cdot \quad 0.37 \cdot 10^1 &= 0.045066 \cdot 10^3 \\ &= 45.066 \end{aligned}$$

Beispiel: 45.066 : 3.7

$$\begin{aligned} 0.45066 \cdot 10^2 \quad : \quad 0.37 \cdot 10^1 &= 1.218 \cdot 10^1 \\ &= 12.18 \end{aligned}$$

Gleitpunktzahlen: Arithmetik (3)

Seien $x = m_x \cdot 2^{d_x}$ und $y = m_y \cdot 2^{d_y}$.

Addition:

$$x + y = (m_x \cdot 2^{d_x - d_y} + m_y) \cdot 2^{d_y} \quad \text{falls} \quad d_x \leq d_y$$

Subtraktion:

$$x - y = (m_x \cdot 2^{d_x - d_y} - m_y) \cdot 2^{d_y} \quad \text{falls} \quad d_x \leq d_y$$

kleiner Exponent muss großem Exponenten angeglichen werden $\Rightarrow$ Rundungsfehler durch Denormalisierung

Rundungsfehler

Seien x, y, z wie folgt gegeben:

$$\begin{aligned} x &= +0.1235 \cdot 10^3 \\ y &= +0.5512 \cdot 10^5 \\ z &= -0.5511 \cdot 10^5 \end{aligned}$$

Dann gilt

$$\begin{aligned} x + y &= +0.1235 \cdot 10^3 + 0.5512 \cdot 10^5 \\ &= +0.0012 \cdot 10^5 + 0.5512 \cdot 10^5 \\ &= +0.5524 \cdot 10^5 \end{aligned}$$

$$\begin{aligned} (x + y) + z &= +0.5524 \cdot 10^5 - 0.5511 \cdot 10^5 \\ &= +0.0013 \cdot 10^5 \\ &= +0.1300 \cdot 10^3 \end{aligned}$$

Rundungsfehler (2)

Andererseits gilt

$$\begin{aligned} y + z &= +0.5512 \cdot 10^5 - 0.5511 \cdot 10^5 \\ &= +0.0001 \cdot 10^5 \\ &= +0.1000 \cdot 10^2 \end{aligned}$$

$$\begin{aligned} x + (y + z) &= +0.1235 \cdot 10^3 + 0.1000 \cdot 10^2 \\ &= +0.1235 \cdot 10^3 + 0.0100 \cdot 10^3 \\ &= +0.1335 \cdot 10^3 \\ &\neq +0.1300 \cdot 10^3 \end{aligned}$$

Ungenauigkeiten bei Gleitpunktzahlen

Ungenauigkeiten im Umgang mit Gleitpunktzahlen

- bei der Umwandlung vom Dezimal- ins Dualsystem
- und bei den arithmetischen Operationen.

In der Regel spielen kleine Abweichungen keine große Rolle. Im Rechner werden oft tausende von Rechenoperationen hintereinander ausgeführt: kleine Rundungsfehler addieren sich, Resultat wird völlig unbrauchbar!

IEEE-754: Gleitpunktzahlen

- Exponent in Exessdarstellung speichern
- normalisiert wird auf 1.xxxxxx
- die führende Eins wird nicht abgespeichert
- einfache Genauigkeit (32Bit)
  - * Exzess: $2^7 - 1 = 127$
  - * 1Bit Vorzeichen, 8 Bit Exponent, 23 Bit Mantisse
- doppelte Genauigkeit (64Bit)
  - * Exzess: $2^{10} - 1 = 1023$
  - * 1Bit Vorzeichen, 11 Bit Exponent, 52 Bit Mantisse

IEEE-754: Gleitpunktzahlen (2)

Beispiel: einfache Genauigkeit

$$(-0.078125)_{10} = (-0.000101)_2 \cdot 2^0 = (-1.01)_2 \cdot 2^{-4}$$

Vorzeichen	:	1	
Exponent	:	01111011	($\hat{=}$ -4 + 127)
Mantisse	:	01000000000000000000000	

Ergänzende Literatur:
Walter Oberschelp und Gottfried Vossen: Rechneraufbau und Rechnerstrukturen, Oldenbourg Verlag