Fixpunktdarstellung

- Fixed-point numbers
- Bsp. Dezimaldarstellung
  - Dezimalkomma (decimal point) rechts von Stelle mit Wertigkeit 100
  - nachfolgende Stellen haben Wertigkeit 10^{-1}, 10^{-2}, etc.
- Binärdarstellung
  - analog, Wertigkeiten rechts des (gedachten) "Dualpunktes": 2^{-1}, 2^{-2}, etc.

$$w(b_{n-1}b_{n-2}\dots b_0 b_{-1}b_{-2}\dots b_{-m}) = f(b_{n-1}, \sum_{i=-m}^{n-2} b_i 2^i)$$

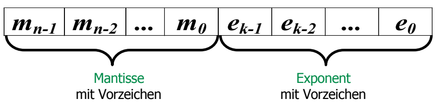
gedachter Dualpunkt

- Arithmetische Operationen
  - genau wie bei Darstellung ganzer Zahlen
  - vor Ausführung muß sichergestellt sein, daß der Dualpunkt bei allen Operanden an derselben Stelle steht
  - Danach mit Binärzahlen wie mit Integer-Binärzahlen rechnen
  - Zum Schluß evtl. Dualpunkt wieder an die richtige Position rücken
- Häufige Konvention: bei 32 Bit Darstellung 16.16 Bit
- Achtung bei Multiplikation: 32x32 Bit → 64 Bit, muß wieder auf 32 Bit "zurechtgestutzt" werden!

Floating-point Zahlen

- Probleme von Fixpunktzahlen
  - sehr große Zahlen, können nicht dargestellt werden, da Wertigkeit des höchstwertigen Bits festgelegt ist → Überlauf (overflow)
  - sehr kleine Zahlen, können nicht dargestellt werden, da Wertigk. des nieder-wertigsten Bits festgelegt ist → Unterlauf (underflow)
- wünschenswert
  - großes Intervall des Zahlenstrahls darstellbar
  - große Genauigkeit bei kleinen Zahlen, kleinere Genauigkeit bei großen Zahlen
- Lösung: FP-Zahlen ("Gleitpunktdarstellung")
  - entspricht Exponentialschreibweise: $0.4711 * 10^4$
  - Darstellung mit Hilfe von
    - Mantisse mit Vorzeichen
    - Exponent mit Vorzeichen

- Darstellung

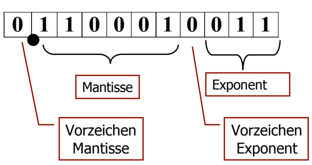


- Berechnung des Wertes

$$w(m_{n-1}\dots m_0 e_{k-1}\dots e_0) = w(m_{n-1}\dots m_0)\cdot 2^{w(e_{k-1}\dots e_0)}$$

Beispiel

- $(6.125)_{10}$ könnte dargestellt werden als



- Probe

$$(0.110001)_2 \cdot 2^3 = (110.001)_2 = 2^2 + 2^1 + 2^{-3} = 4 + 2 + 0.125$$

Normalisierte Gleitpunktdarstellung

- Darstellung bisher nicht eindeutig

$$1.101100 \cdot 2^5 = 0.110110 \cdot 2^6 = 0.011011 \cdot 2^7$$

- Definition Normierung:

> Eine FP-Zahl zur Basis 2 heißt normalisiert, falls gilt:
> $$1 \le |w(m)| < 2$$

  - höchstwertiges Bit der Mantisse = 1
  - höchstwertiges Bit ist damit redundant und kann weggelassen werden
    - diese Darstellung nennt man auch "Signifikand"

Interpretation der Darstellung

- Welche FP-Zahl ist:

0	1	1	0	0	0	1	0	0	1	1

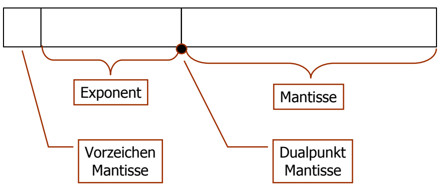
?

- Ist das überhaupt eine Gleitpunktzahl?
- Länge von Mantisse und Exponent?
- Zuerst Mantisse oder zuerst Exponent?
- Zahldarstellung für Mantisse?
- Zahldarstellung für Exponent?
- Normalisiert oder nicht?

IEEE 754

- Standardisierung sinnvoll, insbesondere bei der Datenkommunikation von Rechner zu Rechner
- Institute of Electrical and Electronics Engineers (IEEE)
  - begann 1979 mit der Erarbeitung eines Standards für Gleitpunktzahlen
  - veröffentlichte das Ergebnis 1985 als Standard "IEEE 754"
- wird seitdem in allen Computern benutzt
- Davor: heilloses Chaos
  - Insbesondere: Dasselbe Programm auf versch Plattformen hatte verschiedene numerische Stabilität und sonstige Eigenschaften

IEEE-Format



- Mantisse
  - Betrag und Vorzeichen
  - Darstellung als Signifikand, d.h.
    - normalisiert
    - führende 1 wird weggelassen (außer bei *extended precision*, s.u.)
    - Dualpunkt hinter führender 1 (vor dargestellten Bits)

• | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | $= (1.01000101)_2$

- Exponent
  - vorzeichenlose ganze Zahl mit *bias*
    - Bem.: $b_{n-1}2^{n-1}$ bei Zweierkomplement war ein Bias
- Definition "Bias":
  - *bias* muß subtrahiert werden, um wahren Exponenten zu erhalten

> Engl. *bias*: Hang, Neigung, Vorliebe, Vorurteil.
> bezeichnet oft "Verschiebung um additive Konstante"

- die Werte 0...0 und 1...1 sind reserviert

- Beispiel (8 Bit Exponent, *bias* 127)
  - wahrer Exponent: $(12)_{10}$
  - Darstellung: $(12)_{10} + (127)_{10} = (139)_{10}$

±	1	0	0	0	1	0	1	1	Mantisse

  - wahrer Exponent: $(-2)_{10}$
  - Darstellung: $(-2)_{10} + (127)_{10} = (125)_{10}$

±	0	1	1	1	1	1	0	1	Mantisse

- Exponent 0...0
  - nicht normalisierte Mantisse
  - führendes (weggelassenes) Bit ist 0
  - noch kleinere Zahlen darstellbar
  - Ist Mantisse auch 0...0: Zahl 0 ("+0" oder "-0")
- Exponent 1...1
  - Mantisse = 0..0
    - unendlich (z.B. x/0) ("+∞" oder "-∞")
  - Mantisse ≠ 0..0
    - NaN (*not a number*) undefiniertes Resultat (z.B. ∞/ ∞)

IEEE 754 standardisiert drei Genauigkeiten

- einfache Genauigkeit (*single precision*): 32 Bit
  - Genauigkeit: ca. 7 Dezimalstellen
- doppelte Genauigkeit (*double precision*): 64 Bit
  - Genauigkeit: ca. 15 Dezimalstellen
- erweiterte Genauigkeit (*extended precision*): 80 Bit
  - Genauigkeit: ca. 19 Dezimalstellen
  - wird nur innerhalb FPU zur Reduzierung von Rechenungenauigkeiten benutzt!
  - Default bei allen aktuellen CPUs
  - Läßt sich abschalten

Item	Single precision	Double precision
Bits in sign	1	1
Bits in exponent	8	11
Bits in fraction	23	52
Bits, total	32	64
Exponent system	Excess 127	Excess 1023
Exponent range	−126 to +127	−1022 to +1023
Smallest, normalized	2^{-126}	2^{-1022}
Largest, normalized	approx. 2^{+128}	approx. 2^{+1024}
Decimal range	approx. 10^{-38} to 10^{+38}	approx. 10^{-308} to 10^{+308}
Smallest, denormalized	approx. 10^{-45}	approx. 10^{-324}

Normalized	±	0 < Exp < Max	Any bit pattern
Denormalized	±	0	Any nonzero bit pattern
Zero	±	0	0
Infinity	±	1 1 1 . . . 1	0
Not a number	±	1 1 1 . . . 1	Any nonzero bit pattern

Sign bit

- Arithmetische Operationen
  - erheblich aufwendiger als bei ganzen Zahlen (in 2er-Komplement)
  - alle modernen Prozessoren verfügen über FPU (*floating point unit*)
    - sonst zeitraubende Berechnung in länglichen Unterprogrammen
  - nicht jede FPU ist 100% kompatibel zum Standard (i. Allg. aber gut genug)

Arbeiten mit FP-Zahlen

- auch "einfache" Dezimalzahlen sind nicht exakt darstellbar (Rundungsfehler)
  - Endlicher Dezimalbruch kann unendlicher Dualbruch sein
  - Beispiel: $0.1_{10}=0.00011001100110011..._2$
- bei arithmetischen Operationen entstehen weitere Ungenauigkeiten
- Vergleich zweier Gleitkommazahlen ist problematisch

  0.1*5 == 0.5 ?!?
- Man muß beim Programmieren mit FP-Zahlen immer mit Rundungsfehlern rechnen!

Verfahren zur Konvertierung dezimal → dual

- Ann.: z < 1 ("Normierung")
1. Schreibe eine 0 als Vorkommastelle
2. Falls z ≥ 1, ziehe 1 von z ab und nenne das Ergebnis wieder z.
3. Multipliziere z mit 2 und nenne das Ergebnis wieder z.
4. Die Vorkommastelle von z ist nun die nächste duale Nachkommastelle.
5. Weiter bei 2.

Beispiel

- z = 0.6

z_i	Dualzahl (mit Genauigkeit *i* Bits)
0.6	0.
1.2	0.1
0.4	0.10
0.8	0.100
1.6	0.1001
1.2	0.10011

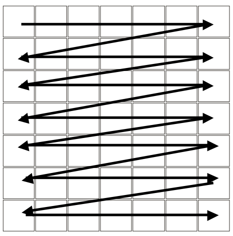
- Offensichtlich wiederholt sich das Ganze jetzt periodisch, also $0.6_{10} = 0.\overline{1001}_2$

Darstellung von Programmen

- Anweisungen an den Computer, bestimmte Dinge zu tun
- Erstellung eines Programmes als Text (ASCII)
- Übersetzung in Maschinensprache
- Viele andere (Zwischen-)Repräsentationen
  - *Annotated Syntax-Tree*
  - Byte-Code
  - Assembler
  - (s. Compiler-Bau)

Grafiken, Bilder

- Bilder werden als Folge von Rasterpunkten dargestellt



Bitmap
(Rastergrafik)