Informatik 1
Codierung von Zahlen

Vorlesungsfolien

Zahlendarstellung

- Stellenwertsystem
- Signed Magnitude
- 1er-Komplement
- 2er-Komplement
- IEEE 754 Gleitkommaformat

Stellenwertsystem

- Repräsentation **positiver ganzer Zahlen**
  - Anzahl b von Ziffern gegeben, z.B. 0, 1, 2, 3, ..., 9
  - Eine ganze Zahl ist eine Folge a_i dieser Ziffern mit i = 0, 1, ..., n
  - Die Stelle i einer Ziffer in der Folge gibt die Wertigkeit b^ider Ziffer zu einer Basis an

$$a_n a_{n-1} \dots\ a_1 a_0 \coloneqq \sum_{i=0}^{n} \left(a_i \cdot b^i \right)$$

- b wird Basis der Zahlendarstellung genannt
- Die Basis wird oft tiefgestellt hinter die Zahl geschrieben
- Beispiel: Dezimalsystem
  - Basis 10 und arabischen Ziffern 0 bis 9

$$173_{10} = 1 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0$$

Stellenwertsystem

- Dualzahlen / Binärsystem
  - Basis 2, zwei Ziffern 0 und 1 (Bit, binary digit)

  $$1011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
  $$= 8 + 2 + 1 = 11_{10}$$

- Oktalsystem
  - Basis 8, Ziffern 0, 1, 2, ..., 7

  $$137_8 = 1 \cdot 8^2 + 3 \cdot 8^1 + 7 \cdot 8^0$$
  $$= 64 + 24 + 7 = 95_{10}$$

- Hexadezimalsystem
  - Basis 16, Ziffern 0, 1, ..., 9, A, B, ... , E, F (auch a, ..., f)

  $$21D_{16} = 2 \cdot 16^2 + 1 \cdot 16^1 + 13 \cdot 16^0$$
  $$= 512 + 16 + 13 = 541_{10}$$

Stellenwertsystem

Aufgabe

Gegeben:	333_8 und 101010_2
Gesucht:	Wert im Dezimalsystem

Stellenwertsystem

Hexadezimal	Binär	Oktal	Dezimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
A	1010	12	10
B	1011	13	11
C	1100	14	12
D	1101	15	13
E	1110	16	14
F	1111	17	15

Stellenwertsystem

Aufgabe

Gegeben:	Hexadezimalzahl 1C9
Gesucht:	Wert im Binär-, Oktal- und Dezimalsystem

- Einfache Umrechnung mit Hilfe der Tabelle bei einer Basis, die eine 2er-Potenz darstellt.

$$47_8 = 100\ 111 = 10\ 0111 = 27_{16}$$

Stellenwertsystem

- Grundrechenarten wie Schulmethode
  - **Ziffernweise** rechnen mit Dezimalziffern

$$\begin{array}{r} 273 \\ +\ \underline{854} \\ 1127 \end{array}$$

$$\begin{array}{r} \underline{126 \cdot 13} \\ 126 \\ {}_{1}378 \\ \hline 1638 \end{array}$$

$$\begin{array}{l} \ \ 475 : 12 = 39 \text{ Rest } 7 \\ -\underline{36} \\ \ \ \ 115 \\ \ \ -\underline{108} \\ \ \ \ \ \ \ 7 \end{array}$$

  - Analog mit **Binärziffern**

$$\begin{array}{r} 1111 \\ +\ \underline{0101} \\ 10100 \end{array}$$

$$\begin{array}{r} \underline{1011 \cdot 1001} \\ 1011 \\ \\ {}_{1}1011 \\ \hline 1100011 \end{array}$$

$$\begin{array}{l} \ \ 110100 : 101 = 1010 \text{ Rest } 10 \\ -\underline{101} \\ \ \ \ \ \ 110 \\ \ \ \ -\underline{101} \\ \ \ \ \ \ \ \ \ 10 \end{array}$$

Stellenwertsystem

- Ganze Zahlen sind im Computer binär codiert
- Fast immer als Dualzahl
- Die Codierung ist limitiert auf **endliche vielen Bits**
- Bit: Binärziffer, Binärstelle
- z.B. 8 (8-Bit CPUs), 16, 32, 64 (heutige 64-Bit-CPUs)
- Wort: Kleinste natürliche Dateneinheit eines Computers
- Byte: Anzahl Bits zur Codierung von Zeichen
- IBM System/360: Byte = 8 Bits, Wort = Byte
- Java Virtual Machine: Wort = 32 Bits
- Java: Vier ganzzahlige Codierungen:
  - 8, 16, 32, 64 Bits
  - Datentyp byte in Java: 8 Bits = 1 Oktett

Negative ganze Zahlen

- Drei Varianten
  - signed magnitude
  - 1er-Komplement
  - 2er-Komplement (häufigste Variante)
- Das höherwertigste Bit gibt das Vorzeichen an:
  0 = +   1 = -

Beispiel 8-Bit signed magnitud	**Vorzeichen**
1000 0100	Negativ
0000 0100	Positiv

Signed Magnitude

- Höherwertigste Bit (Bit mit „höchsten“ Wertigkeit) ist das Vorzeichenbit
- Restlichen Bits codieren den Betrag der Zahl

Beispiel 8-Bit signed magnitud	Wert
0000 0001	1
1000 0001	- 1
1001 0100	- 20

- Probleme:
  1. Vorzeichen und Subtraktion ist gesondert in der Hardware zu implementieren
  2. Repräsentation der Null nicht mehr eindeutig:
     1000 0000 = -0   0000 0000 = +0

1er-Komplement

- Positive Zahlen als Binärzahl mit 0 als Vorzeichenbit
- Eine Negative Zahl ist codiert als Komplement des Absolutbetrags der Zahl
- Komplement einer Binärzahl:
  - 1 wird durch 0, 0 durch 1 ersetzt.
  - ~ wird oftmals als Komplementoperator verwendet, wie ein Vorzeichen

Beispiel 8-Bit 1er-Komplement	Wert
1111 1100	-3

- Rechnung:

  Absolutbetrag: ~1111 1100 = 000 0011 = 3

  Dezimaler Wert ist -3

1er-Komplement

- Binäre Addition mit negativer Zahl, ergibt immer einen Fehler von 1
- Übertrag ins 9. Bit wird hinzuaddiert (Carry-Bit)
- Beispiel: 8-Bit 1er-Komplement

```
   0001 0111      (  23  )
 + 1111 1100      (+ -3  )
 1 0001 0011      (19, es fehlt 1 )
 + 0000 0001      (1 hinzuaddieren)
   0001 0100   (20)
```

- Beispiel: Addition negativer 8-Bit 1er-Komplement Zahlen

```
   1111 1100  (-3)
 + 1111 1100  (+ -3)
 1 1111 1000  (-7)
 + 0000 0001
   1111 1001  (-6)
```

1er-Komplement

- Hardwareimplementierung
  - Binäre Addition
  - Komplement (einfache Schaltung)
  - Carry-Bit addieren (sehr einfache Schaltung)
- Immer noch zwei Darstellungen der 0

2er-Komplement

- Positive Zahlen als Binärzahl mit 0 als Vorzeichenbit
- Eine negative Zahl ist codiert als Komplement des Absolutbetrags der Zahl **plus 1**
- Beispiel: 8-Bits 2er-Komplement, -3

  ~0000 0011 (3)
  1111 1100 (1er-Komplement der 3)
  \+ 0000 0001
  1111 1101 (-3 im 2er-Komplement)

- Absolutbetrag einer negativen 2er-Komplementzahl
  – Komplement bilden und dann 1 addieren

  (-3) 1111 1101 → 0000 0010 → 0000 0011 (3)

2er-Komplement

- Nur noch eine Repräsentation der 0
- Binäre Addition funktioniert auch mit negativen Zahlen (ohne Addition Carry-Bit)

  Beispiel Addition 3 + -3

  ```
    1111 1101   (-3)
  + 0000 0011   (3)
  1 0000 0000   (0)  (Carry-Bit wird verworfen)
  ```
- Hardwareimplementierung für Negation
  - Komplement (einfach)
  - Addition mit 1 (sehr einfach)

2er-Komplement

Aufgaben
Geben Sie 18 und -18 im 8-Bit 2er-Komplement an!
Bestimmen Sie den Dezimalwert des 8-Bit 2er-Komplements von 1111 1110 !

2er-Komplement

- Wertebereich 8-Bit 2er-Komplement

  -128 bis 127
- Probleme bei Addition, Multiplikation, … (generell für Codierung mit fester Anzahl Bits):

  Überlauf: Ergebnis ist zu groß für die Codierung

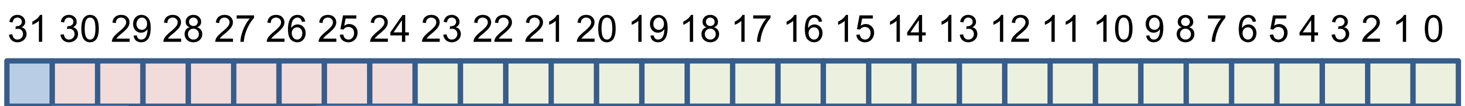
  Unterlauf: Ergebnis ist zu klein für die Codierung
- 0111 1111 + 0000 0001 (= 127 + 1)
  1000 0000 (= -128)
- 1000 0000 + 1111 1111 (= -128 + -1)
  1 0111 1111 (= 127) Carry-Bit wird ignoriert
- Ein Über- oder Unterlauf wird in Java nicht erkannt
- Programmierer ist verantwortlich, dass dies nicht passiert!

IEEE 754 Gleitkommaformat

Idee (Dezimal):

$$123{,}45 = 1{,}2345 \cdot 10^{3}$$
$$0{,}00123 = 1{,}23 \cdot 10^{-3}$$

Java	IEEE 754 (1985)
float	32-bit single precision
double	64-bit double precision



Vorzeichen
1 Bit

Exponent
8 Bits

Mantisse
23 Bits

$$binär: \quad V \cdot M \cdot 2^{E}$$

E so wählen, dass vor dem Komma nur eine 1 ist (normalisierte Zahl)
V = 0 : positive Zahl   Exponent = 127 + E   127 heißt Charakteristik
V = 1 : negative Zahl   M : Nachkommaanteil ohne die 1 vor dem Komma

double: Vorzeichen 1 Bit, Exponent 11 Bit, Mantisse 52, Charakteristik 1023

IEEE 754 Gleitkommaformat

- Zwei Codierungen der Null:

+0	0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0	1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- Exponent nur von -126 (0000 0001) bis 127 (1111 1111)
- (Not a Number) z.B Wurzel einer negativen Zahl

NaN	0	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Unendlich z.B. bei Division durch 0

+Infinity	0	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-Infinity	1	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- Kleinste positive Zahl

0	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$2^{1-127} \cdot 1$ ungefähr $1{,}17549435 \cdot 10^{-38}$

IEEE 754 Gleitkommaformat

- Beispiel: 10,625 in float umrechnen
- Stellenwertsystem in Nachkommabereich erweitern mit Kehrwerten der 2er-Potentenzen

$$10{,}625_{10} = 8 + 2 + 0{,}5 + 0{,}125 = 1010{,}101$$

$$= 1{,}010101 \cdot 2^3$$

3 = <u>130</u> – 127, 1000 0010 ist Exponent

0	1 0 0 0 0 0 1 0	0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

IEEE 754 Gleitkommaformat

- Beispiel: 0,1 Dezimal in float umrechnen

0,1 * 2 = 0,2
0,2 * 2 = 0,4
0,4 * 2 = 0,8
0,8 * 2 = 1,6
0,6 * 2 = 1,2
0,2 * 2 = 0,4 etc.

- Die Zahl hat eine periodische Darstellung im Binärsystem:

$$0{,}0001\;1001\;1001\;1001\;\ldots\; = \;1{,}1001\;1001\;1001\;\ldots \cdot\; 2^{-4}$$

-4 +127 = <u>123</u> = 0111 1011

0	0 1 1 1 1 0 1 1	1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0

Die Zahl ist abgerundet und **kleiner** 0,1

- In Java wird zur Gleitkommazahl mit geringsten Abstand gerundet

IEEE 754 Gleitkommaformat

- Der Standard beschreibt auch die Ergebnisse von Rechenoperationen
  - Addition, Multiplikation, …
  - Wurzel
  - Rundungen
  - Konvertierung
  - Ausnahmen
  - Größenvergleiche