Bits, Bytes und Zahlen

Informatik Grundlagen
Kantonsschule am Burggraben

Ivo Blöchliger

Zahlen im Computer

- Darstellung der Zahlen im Stellwertsystem
- Beste Basis? 2, 10, 16?
- Ziffern müssen zuverlässig unterschieden werden.
  - Zwei Ziffern (Zustände) ist am einfachsten
  - Konkret: Elektrische Spannung an oder aus.
    - 0V oder 1.5V bei modernen CPUs, höhere Werte für kleine CPUs
- Es gibt aber heute Speicher- und Übertragungsprotokolle mit mehr als 2 Zuständen (z.B. SD-Karten mit bis zu 8 Zuständen oder Ethernet mit 3 oder mehr Zuständen).

Bit

- Binary Digit (Binäre Ziffer)
- 0 oder 1
- Falsch oder wahr (False / True)
- 1 Bit → Zahlen 0 und 1
- 2 Bits → Zahlen von 0b00=0 bis 0b11=3 (4 Zustände)
- 3 Bits → Zahlen von 0b000=0 bis 0b111=7 (8 Zustände)
- 8 Bits → ?

Pause

0b 0000'0000 = 0 bis
0b 1111'1111 = 255

$2^8 = 256$
Zustände

Byte: 8 (geordnete!) Bits

- 1 Byte: 8 Bits: 2**8 = 256 Zustände.
- 2 Bytes: 2**16 = 65’536 Zustände.
- 4 Bytes: 2**32 ca. 4 Milliarden Zustände.
- 8 Bytes (64 Bits) können von heutigen CPUs auf einmal verarbeitet werden: ca. 18 Trillionen Zustände.
  - Und das ca. 1 Milliarde Mal pro Sekunde!

Zahlen und Bytes

- 1 Byte: 256 Zustände (z.B. Basis 256, jede Stelle ein Byte)
- Daten im Computer immer in Vielfachen von Bytes
- 1 Byte → 8 Bits → 2 Vierergruppen von Bits
- 4 Bits → 16 Zustände
- **Eine Vierergruppe in binär ist genau eine Hex-Ziffer!**

0x0 bis 0xf

0b0000 bis 0b1111

Binär ↔ Hexadezimal

```
0b1111’1010’1100’1110
      15   10   12   14
0x     f    a    c    e


0x  c   0   1   d   c   a   f   e
ob1100’0000’0001’1101’1100’1010’1111’1110
```

8 4

4 Bytes

Negative ganze Zahlen?

- Bis jetzt nur natürliche Zahlen (mit der Null, natürlich!)
- Idee 1: Erstes Bit ist Vorzeichen, restliche Bits für Betrag
  - Damit ist immer Fallunterscheidung nötig, ungünstig
- Idee 2: Negative Zahlen so darstellen, dass die normale Addition das richtige Resultat liefert.

Darstellung von -1 (in 8 Bits)

- Bei der Addition von 1 muss 0 herauskommen:

```
  0b 0000’0001         1
+ 0b ????’????        -1

----------------------
  0b 0000’0000         0
```

Darstellung von -1 (in 8 Bits)

- Bei der Addition von 1 muss 0 herauskommen:

```
    0b 0000'0001          1
  + 0b 1111'1111         -1
       1 1111 111
  -----------------------
    0b 0000'0000          0
```

9. Bit

2 = 0b10

10'1010
+ 1111 1111
= 0010 1001 = 41

- Das höchstwertige Bit wird ignoriert (nur 8 Bits).

Darstellung von -10

- Finden Sie diese selbständig.

Pause

Darstellung von -10 (in 8 Bits)

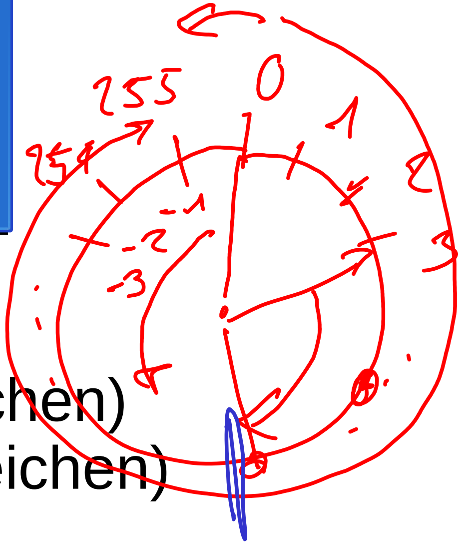
```
  0b 0000'1010         10
+ 0b 1111'0110        -10
    11111 11
------------------------
  0b 0000'0000          0
```

$-10 = 0b1111'0110$

Negative Zahlen (8 Bits)

- Feststellung:

  -1 = 0b1111’1111, (also 255=256-1 ohne Vorzeichen)
  -10 = 0b1111’0110, (also 246=256-10 ohne Vorzeichen)



- Anordnung der Zahlen von 0 bis 255 im Kreis.
- Addition: Vorwärtsgehen im Kreis (nach 255 wieder 0)
- Gleiches Resultat, ob man 255 addiert oder 1 subtrahiert (wenn der Überlauf ignoriert wird)

Höchstwertiges Bit als Vorzeichen

- Konvention:
  - Positive Zahlen 0 bis 127 = 0b0111’1111
  - Negative Zahlen -1 bis -128 = 0b1000’0000

Umrechnungen für -n

-42

- Betrag n von 256 abziehen, oder

+42 - 1

- Das Komplement (alle Bits umdrehen) von (n-1) bilden.
- Beispiel mit -10: 8 1
  10-1=9 = 0b0000’1001, alle Bits invertieren liefert
  -9 = 0b1111’0110
- Rechnen Sie die Darstellung von -100 aus.

Pause!

-100 in 8 Bits

- 256 – 100 = 156 = 0b1001’1100

  oder

- 100-1 = 99 = 0b0110’0011
  Invertieren liefert -100 = 0b1001’1100

Dezimal- und Binärbrüche

- 3.147 heisst

  10^{-1} 10^{-2} 10^{-3}

  3 Einer + 1 Zehntel + 4 Hunderstel + 7 Tausendstel

- 0b11.011 heisst

  1 Zweier + 1 Einer + 1 Viertel + 1 Achtel

$$3.375 = 0b11.011$$

1/10 in Binär?

- Entweder umrechnen mit 1/2, 1/4, 1/8, etc.
- Oder schriftliche Division im Binärsystem

1/10 in binär

1/2, 1/4, 1/8 zu gross, als erstes passt 1/16:

0b0.0001, die Differenz ist 1/10-1/16 = $\boxed{3/80}$ > 1/32 = 3/96

0b0.00011, es bleibt noch 3/80-1/32 = $\boxed{1/160}$ > $\boxed{1/256}$ $= 2^{-8}$

$1/32$

- Also 1/10 ~ 0b 0.00011001…

$1/128$

$\frac{1}{3} = 0.3333\ldots$

- Hört der Binärbruch einmal auf?

1/10 mit Binärdivision

Divisionsalgorithmus (hier ohne 0b-Prefix)

1. : 1010 = 0,0001100110011001100....

10000
1010
11
001100
− 1010
1
0010000

1/10 mit Binärdivision

Divisionsalgorithmus (hier ohne 0b-Prefix)

```
1  : 1010 = 0.00011001100110011...
10000
 1010
-----
  1100
  1010
  ----
    10000
     1010
    -----
     1100 ...etc
```

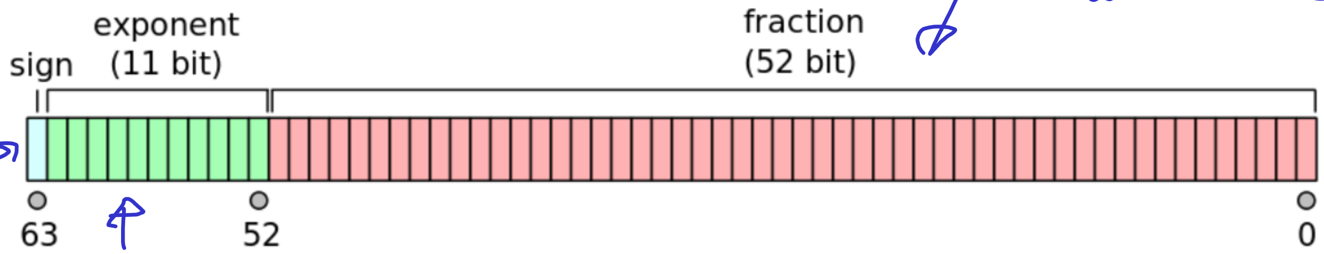
0.1 ist binär nicht exakt darstellbar

- Wie z.B. 1/3 = 0.333333 etc.
- Der Computer kann nur eine endliche Anzahl Stellen speichern! Immer potentiell kleine Rundungsfehler.
- Z.B. ist 1.2-1.1 nicht gleich 0.1
- Immer aufgepasst, wenn mit Floats (Kommazahlen) gearbeitet wird!

Darstellung von Binärbrüchen

- 64 Bit, aufgeteilt in

Mantisse



The real value assumed by a given 64-bit double-precision datum with a given biased exponent e and a 52-bit fraction is

$$(-1)^{\text{sign}}(1.b_{51}b_{50}\ldots b_0)_2 \times 2^{e-1023}$$

- Genauigkeit ca. 17 Dezimalstellen (53 Binärstellen)

$3{,}0 \cdot 10^8 \, m/s$

1