

Verlässliche Systemsoftware

Übungen zur Vorlesung

Festkommaarithmetik

Phillip Raffeck, Simon Schuster, Peter Ulbrich

Technische Universität Dortmund

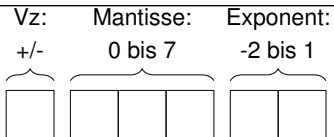
Lehrstuhl für Informatik 12 (Arbeitsgruppe Systemsoftware)

<https://sys.cs.tu-dortmund.de>

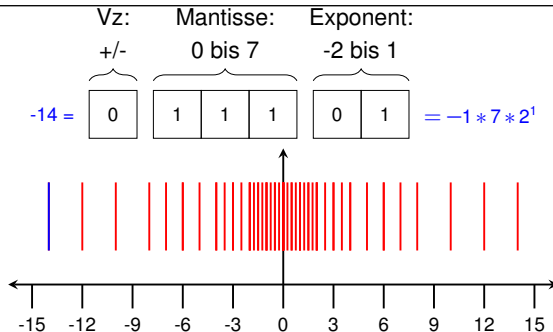
Wintersemester 2020



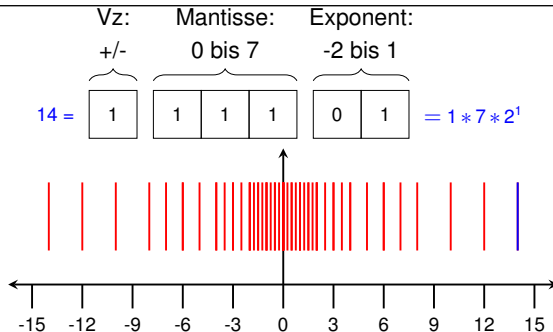
Fließkommazahlen



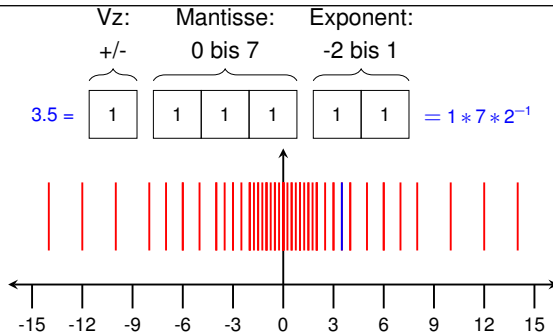
Fließkommazahlen



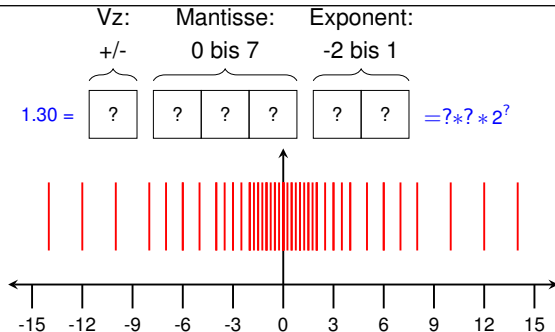
Fließkommazahlen



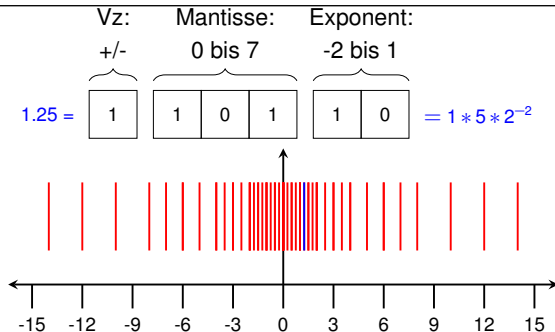
Fließkommazahlen



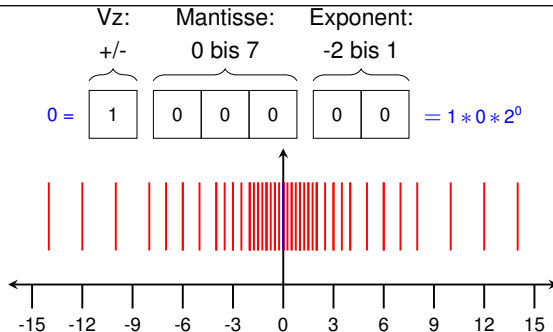
Fließkommazahlen



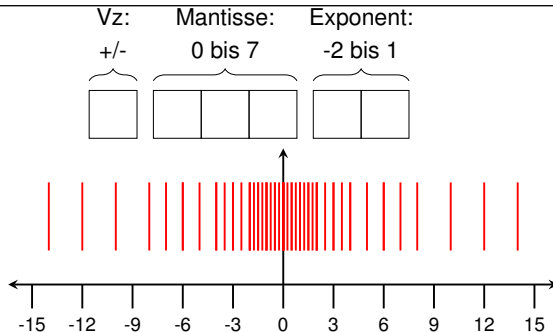
Fließkommazahlen



Fließkommazahlen



Fließkommazahlen



IEEE 754

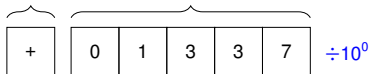
- Noch komplexer:
 - normalisierte/denormalisierte Darstellung
 - Rundung, Fehlersemantik, ...
 - NaN, ∞ , ...
- <https://ieeexplore.ieee.org/document/4610935/>

Festkommazahlen: Grundlagen

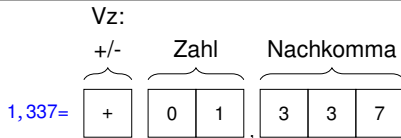
Vz:

+/-

Zahl



Festkommazahlen: Grundlagen



Festkommazahlen: Grundlagen

Vz:

+/-

Zahl

Nachkomma

+	

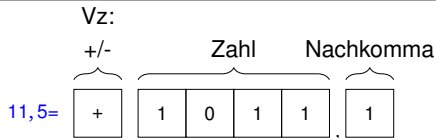
0	1

3	3	7

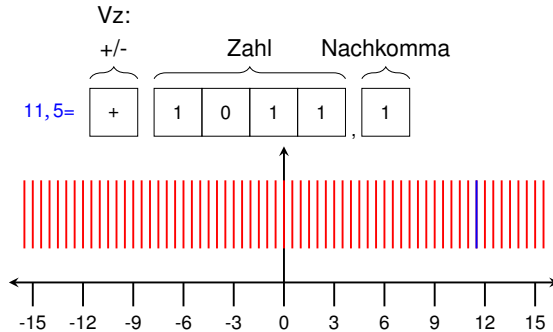
$\div 10^3$



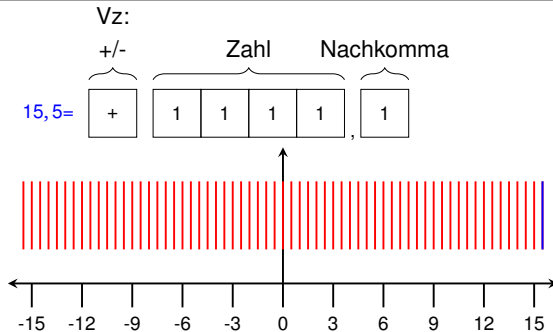
Festkommazahlen: Grundlagen



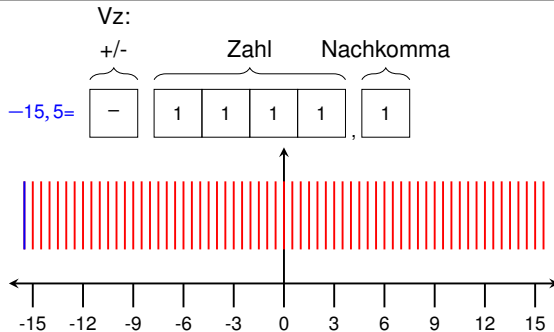
Festkommazahlen: Grundlagen



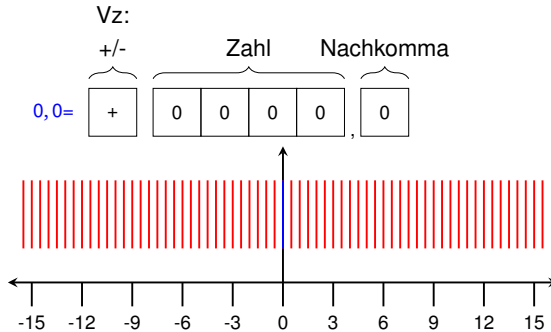
Festkommazahlen: Grundlagen



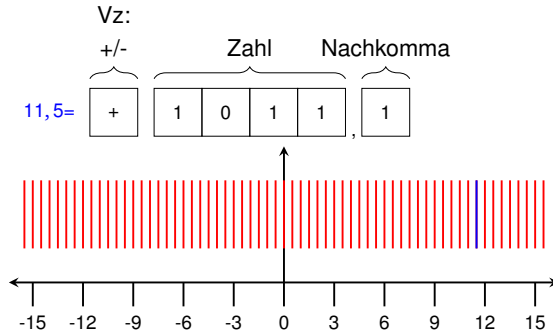
Festkommazahlen: Grundlagen



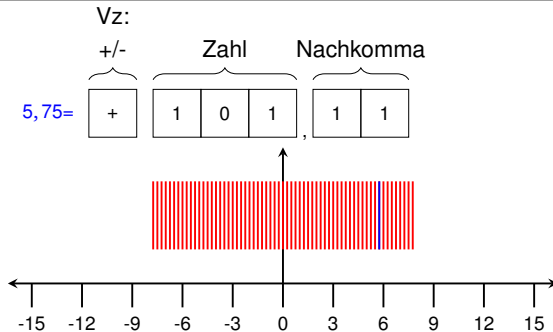
Festkommazahlen: Grundlagen



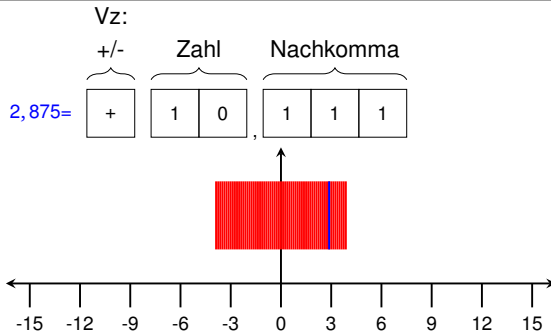
Festkommazahlen: Grundlagen



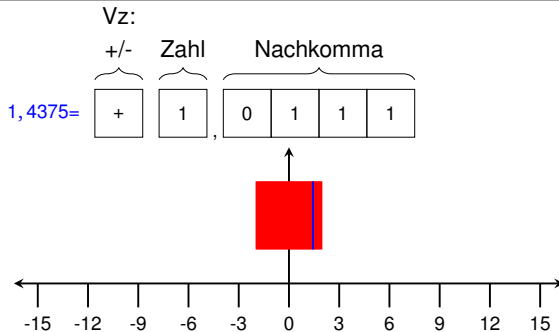
Festkommazahlen: Grundlagen



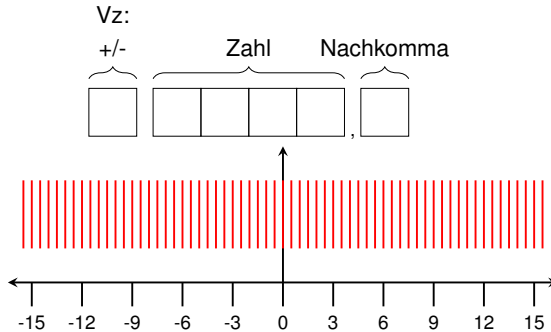
Festkommazahlen: Grundlagen



Festkommazahlen: Grundlagen



Festkommazahlen: Grundlagen



C-Standard und Zahlendarstellung

- Zahlendarstellung im Standard nicht festgelegt:
 - Einerkomplement
 - Vorzeichen und Magnitude
 - Zweierkomplement
- Heute meist Zweierkomplement $\leadsto$ kein dediziertes Vorzeichenbit

```
float func(void){  
    volatile float a = 23.42;  
    volatile float b = 12.34;  
    return a * b;  
}
```

```
func:  
    push    {r7, lr}  
    sub     sp, #8  
    add     r7, sp, #0  
    ldr     r3, [pc, #28] ; float a  
    str     r3, [r7, #4]  
    ldr     r3, [pc, #28] ; float b  
    str     r3, [r7, #0]  
    ldr     r2, [r7, #4]  
    ldr     r3, [r7, #0]  
    adds    r0, r2, #0 ; Param 1  
    adds    r1, r3, #0 ; Param 2  
    bl      3a6c <__aeabi_fmul>  
    adds    r3, r0, #0  
    adds    r0, r3, #0  
    mov     sp, r7  
    add     sp, #8  
    pop     {r7, pc}
```

- Setup
 - Plattform: ARM Cortex-M0+
 - Compiler: arm-gcc
- Funktion `__aeabi_fmul` : 300 Zeilen Assembler
- Keine Fließkommaeinheit (engl. floating-point unit, FPU) vorhanden
- Emulation der **Fließkommaarithmetik in Software**



- Mikrocontroller ohne *Fließkommaeinheit*
- *Kein EAN* für Fließkommazahlen
 - ↪ *Festkommaarithmetik* mit Ganzzahlen
- Zahlenformat häufig in Q-Notation [1] angegeben
- $Qm.n \leadsto$ Festkommazahl mit
 - m Bit vor dem Komma, n nach dem Komma, ein Vorzeichenbit
 - Wertebereich: $[-2^m, 2^m - 2^{-n}]$
 - Auflösung: 2^{-n}
- Implementierung für Übungsaufgabe *vorgegeben*

Implementierung als Integer

↪ passendes Q-Format ist **anwendungsspezifisch**



von Fließkomma nach Qm.n

- 1 Multiplikation mit 2^n
- 2 Runden auf die nächste Ganzzahl

von Qm.n nach Fließkomma

- 1 Umwandlung in Fließkommazahl $\leadsto$ cast
- 2 Multiplikation mit 2^{-n}



■ Addition und Subtraktion wie bei Ganzzahlen

Addition

```
1 int32_t      a = ...;  
2 int32_t      b = ...;  
3 int32_t result = a + b;
```

$$\begin{array}{r} 2,80 \\ + 13,37 \\ \hline = 16,17 \end{array}$$

Subtraktion

```
1 int32_t      a = ...;  
2 int32_t      b = ...;  
3 int32_t result = a - b;
```

$$\begin{array}{r} 16,17 \\ - 2,80 \\ \hline = 13,37 \end{array}$$



- Braucht Zwischenergebnis von doppelter Bitbreite

Multiplikation

```
1 #define K      (1 << (n - 1))
2 int32_t      a = ...;
3 int32_t      b = ...;
4 int64_t temp = (int64_t) a * (int64_t) b;
5             temp += K;
6 int32_t result= temp >> n;
```

$$a \cdot b$$

$$\stackrel{Q.n}{=} (a \cdot 10^n) \cdot (b \cdot 10^n)$$

$$= (a \cdot b) \cdot 10^{2n}$$

$$\neq (a \cdot b) \cdot 10^n$$

Division

```
1 int32_t      a = ...;
2 int32_t      b = ...;
3 int64_t      temp = (int64_t) a << n;
4             temp += b / 2;
5 int32_t result = temp / b;
```

$$\frac{a}{b} \stackrel{Q.n}{=} \frac{a \cdot 10^n}{b \cdot 10^n}$$

$$= \frac{a}{b} \neq \frac{a}{b} \cdot 10^n$$

- Siehe Implementierung in fixedpoint.c
- **Vorsicht: Rundungsfehler durch Transformationen**





Erick L. Oberstar.

Fixed-point representation & fractional math.

Technical report, Oberstar Consulting, August 2007.