Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

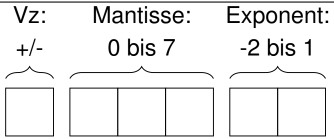
Festkommaarithmetik

Phillip Raffeck, Tim Rheinfels, Simon Schuster, Peter Wägemann

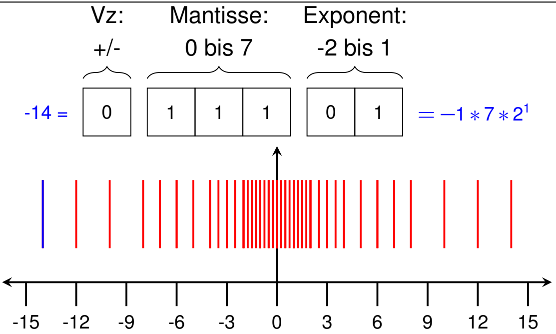
Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
`https://sys.cs.fau.de`

Wintersemester 2022

Fließkommazahlen

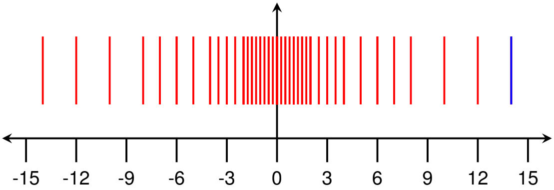


Fließkommazahlen



Fließkommazahlen

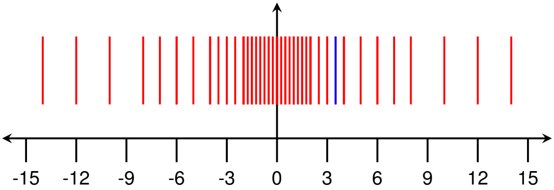
	Vz:	Mantisse:			Exponent:		
	+/-	0 bis 7			-2 bis 1		
14 =	1	1	1	1	0	1	$= 1 * 7 * 2^1$



Fließkommazahlen

	Vz:	Mantisse:			Exponent:	
	+/-	0 bis 7			-2 bis 1	
3.5 =	1	1	1	1	1	1

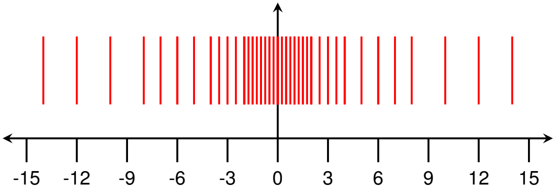
$= 1 * 7 * 2^{-1}$



Fließkommazahlen

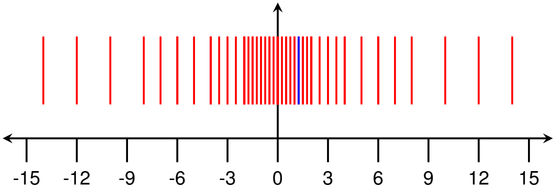
Vz:	Mantisse:	Exponent:
+/-	0 bis 7	-2 bis 1

$1.30 = \boxed{?}\ \boxed{?}\boxed{?}\boxed{?}\ \boxed{?}\boxed{?} = ? * ? * 2^{?}$



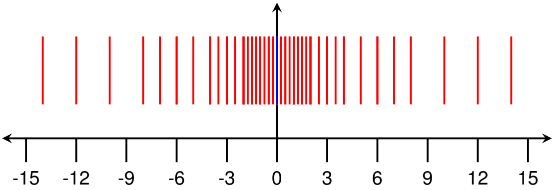
Fließkommazahlen

$$1.25 = \overbrace{\boxed{1}}^{\text{Vz: +/-}}\ \overbrace{\boxed{1}\boxed{0}\boxed{1}}^{\text{Mantisse: 0 bis 7}}\ \overbrace{\boxed{1}\boxed{0}}^{\text{Exponent: -2 bis 1}} = 1 * 5 * 2^{-2}$$

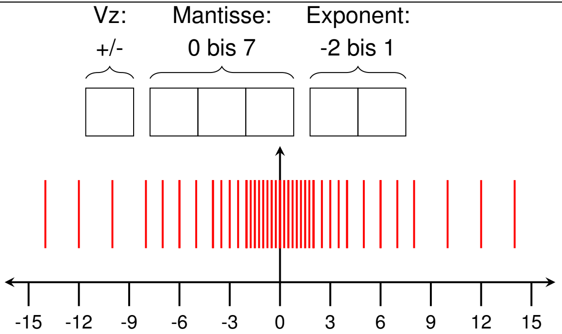


Fließkommazahlen

	Vz:	Mantisse:			Exponent:		
	+/-	0 bis 7			-2 bis 1		
0 =	1	0	0	0	0	0	$= 1 * 0 * 2^0$



Fließkommazahlen



IEEE 754

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

Festkommazahlen: Grundlagen

Vz:

$$\overbrace{\boxed{+}}^{+/-} \quad \overbrace{\boxed{0}\boxed{1}\boxed{3}\boxed{3}\boxed{7}}^{\text{Zahl}} \quad \div 10^{0}$$

Festkommazahlen: Grundlagen

$$1{,}337 = \overbrace{\boxed{+}}^{\text{Vz: } +/-}\ \overbrace{\boxed{0}\boxed{1}}^{\text{Zahl}}\,,\,\overbrace{\boxed{3}\boxed{3}\boxed{7}}^{\text{Nachkomma}}$$

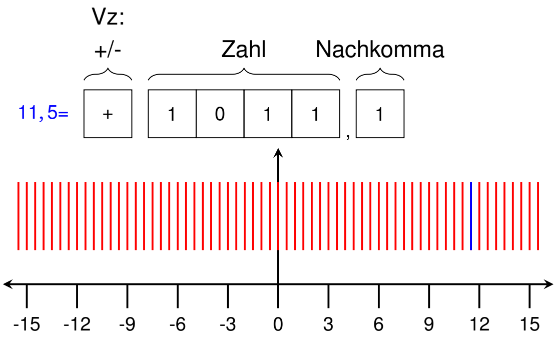
Festkommazahlen: Grundlagen

$$\overbrace{\boxed{+}}^{\substack{\text{Vz:}\\ +/-}} \quad \overbrace{\boxed{0}\boxed{1}}^{\text{Zahl}} \quad \overbrace{\boxed{3}\boxed{3}\boxed{7}}^{\text{Nachkomma}} \quad \div 10^3$$

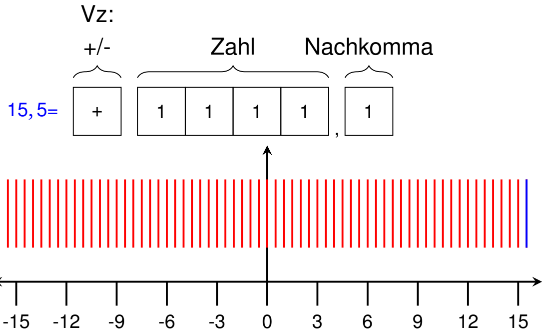
Festkommazahlen: Grundlagen

$$11{,}5 = \overbrace{\boxed{+}}^{\text{Vz: +/-}}\ \overbrace{\boxed{1}\boxed{0}\boxed{1}\boxed{1}}^{\text{Zahl}}\ ,\ \overbrace{\boxed{1}}^{\text{Nachkomma}}$$

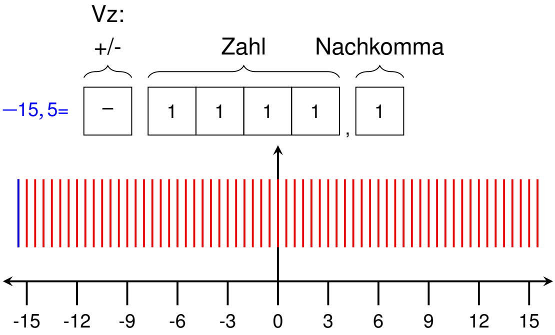
Festkommazahlen: Grundlagen



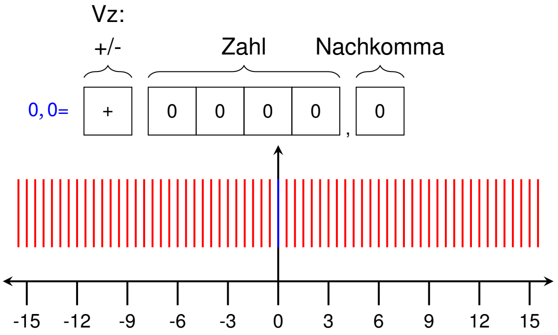
Festkommazahlen: Grundlagen



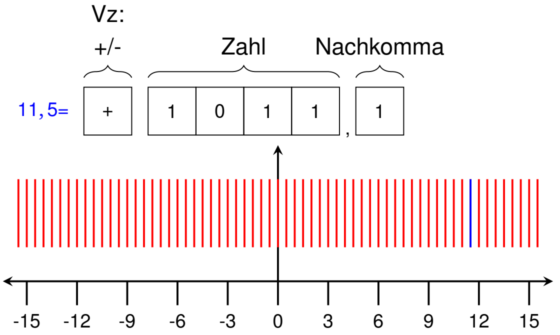
Festkommazahlen: Grundlagen



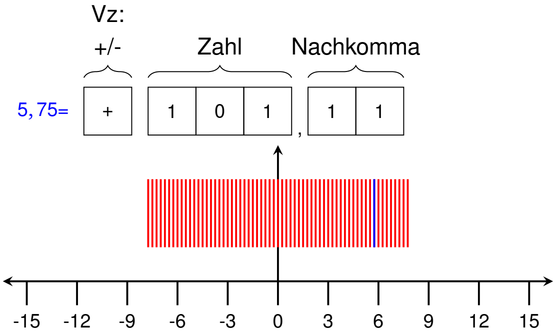
Festkommazahlen: Grundlagen



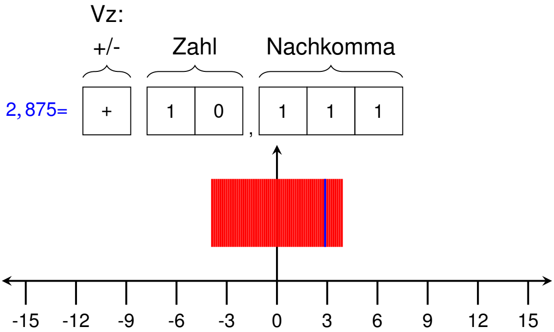
Festkommazahlen: Grundlagen



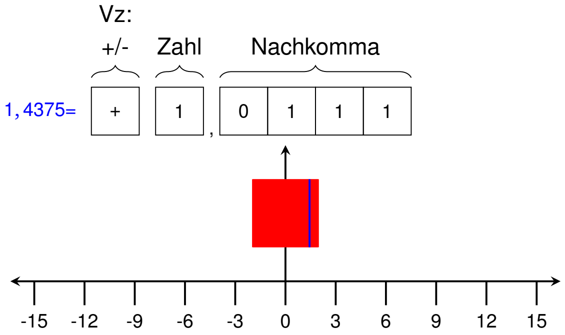
Festkommazahlen: Grundlagen



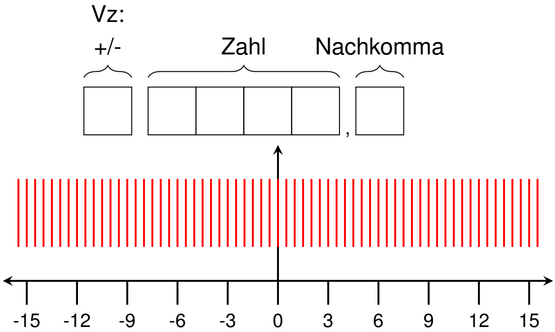
Festkommazahlen: Grundlagen



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

Festkommaarithmethik – Motivation

```c
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**

Festkommaarithmetik – Q-Notation

- Mikrocontroller ohne *Fließkommaeinheit*
- *Kein EAN* für Fließkommazahlen

$\leadsto$ *Festkommaarithmethik* 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: $\left[-2^m, 2^m - 2^{-n}\right]$
  - Auflösung: 2^{-n}
- Implementierung für Übungsaufgabe *vorgegeben*

Implementierung als Integer

$\leadsto$ passendes Q-Format ist **anwendungsspezifisch**

Q-Notation – Beziehung zu Fließkommazahlen

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 $\rightsquigarrow$ cast
2. Multiplikation mit 2^{-n}

Operationen – Addition/Subtraktion

- Addition und Subtraktion wie bei Ganzzahlen

Addition

```c
int32_t      a = ...;
int32_t      b = ...;
int32_t result = a + b;
```

$$
\begin{array}{rrrrrr}
 & & 2 & , & 8 & 0 \\
+ & 1 & 3_{1} & , & 3 & 7 \\
\hline
= & 1 & 6 & , & 1 & 7
\end{array}
$$

Subtraktion

```c
int32_t      a = ...;
int32_t      b = ...;
int32_t result = a - b;
```

$$
\begin{array}{rrrrrr}
 & 1 & 6 & , & 1 & 7 \\
- & & 2_{1} & , & 8 & 0 \\
\hline
= & 1 & 3 & , & 3 & 7
\end{array}
$$

Operationen – Multiplikation/Division

- Braucht Zwischenergebnis von doppelter Bitbreite

Multiplikation

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

$$
\begin{aligned}
& a \cdot b \\
\overset{Q.n}{=} & (a \cdot 10^n) \cdot (b \cdot 10^n) \\
= & (a \cdot b) \cdot 10^{2n} \\
\neq & (a \cdot b) \cdot 10^n
\end{aligned}
$$

Division

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

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

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

Literatur I

Erick L. Oberstar.
Fixed-point representation & fractional math.
Technical report, Oberstar Consulting, August 2007.