Exercise Week 04

GianAndrea Müller
mailto:muellegi@student.ethz

March 21, 2018

Exercise Week 04

GianAndrea Müller
mailto:muellegi@student.ethz

March 21, 2018

Time Schedule

- 3' Nachbesprechung
- 10' Dezimalzahlen im Binärsystem mit Übung
- 10' Fliesskommasystem $\mathcal{F}^*(2, 24, -126, 127)$
- 10' Tips zu Fliesskommazahlen
- 10' Funktionen
- 15' Pause
- 10' Funktionsdefinition- und deklaration
- 10' Pre- and post-conditions
- 10' Übung zu Funktionen

Time Schedule

- 3' Nachbesprechung
- 10' Dezimalzahlen im Binärsystem mit Übung
- 10' Fliesskommasystem $\mathcal{F}^*(2, 24, -126, 127)$
- 10' Tips zu Fliesskommazahlen
- 10' Funktionen
- 15' Pause
- 10' Funktionsdefinition- und deklaration
- 10' Pre- and post-conditions
- 10' Übung zu Funktionen

Learning Objectives
Verständnis des Fliesskommasystems
Nutzung von Funktionen

Learning Objectives

- Verständnis des Fliesskommasystems
- Nutzung von Funktionen

Nachbesprechung
- Kommentieren heisst nicht: Dasselbe in grün.
- Snippets: Verständnis zeigen!

- Ein Kommentar übermittelt Verständnis der Semantik und nicht der Syntax. Wieso nicht Was.
- Man sollte versuchen, den Zweck des Snippets zu verstehen. Wichtig für die Prüfung!

Nachbesprechung

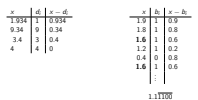
- Kommentieren heisst nicht: Dasselbe in grün.
- Snippets: Verständnis zeigen!

Dezimalzahlen im Binärsystem

Dezimalzahlen im Binärsystem

binär:	1	1	1	1	.	1	1	1
dezimal:	8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Dezimalzahlen im Binärsystem



Dezimalzahlen im Binärsystem

- Erklärung im Dezimalsystem
  1. Stelle abziehen.
  2. Verschiebung mit mal 10.
- Erklärung im Binärsystem
  1. Stelle abziehen.
  2. Verschiebung mit mal 2.
- Zusatzinfo: Es git also Zahlen, die in bestimmten Zahlensystemen keine **finite Darstellung** haben!

x	d_i	$x - d_i$
1.934	1	0.934
9.34	9	0.34
3.4	3	0.4
4	4	0

x	b_i	$x - b_i$
1.9	1	0.9
1.8	1	0.8
1.6	1	0.6
1.2	1	0.2
0.4	0	0.8
1.6	1	0.6
	$\vdots$	

$$1.1\overline{1100}$$

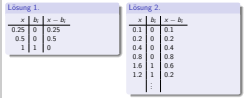
Exercise 4_1 ∼ 2'
Berechne die binäre Darstellung folgender Dezimalzahlen:
0.25
11.1

Exercise 4_1 ∼ 2'

Berechne die binäre Darstellung folgender Dezimalzahlen:

1. 0.25
2. 11.1

Solution 4_1



Solution 4_1

Lösung 1.

x	b_i	$x - b_i$
0.25	0	0.25
0.5	0	0.5
1	1	0

Lösung 2.

x	b_i	$x - b_i$
0.1	0	0.1
0.2	0	0.2
0.4	0	0.4
0.8	0	0.8
1.6	1	0.6
1.2	1	0.2
	⋮	

Unser kleines 10bit Fliesskommasystem

Beschreibung von Fliesskommasystemen

$$\mathcal{F}(\underbrace{\beta}_{\text{Basis} \geq 2}, \overbrace{p}^{\text{Anzahl Stellen} \geq 1}, \underbrace{e_{min}, e_{max}}_{\text{Kleinster und Grösster Exponent}})$$

- Erste Idee: $2.73 \cdot 10^{12}$
- Genau ein Stelle, also 1 bit vor dem Komma
- 5 bits für Nachkommastellen
- 4 bits für den Exponenten
- Anwenden des Bezeichnungsschemas

Unser kleines 10bit Fliesskommasystem

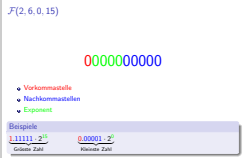
0000000000

- Vorkommastelle
- Nachkommastellen
- Exponent

Unser kleines 10bit Fliesskommasystem

0000000000

- Vorkommastelle
- Nachkommastellen
- Exponent



- Wir möchten auch negative Exponenten.
- Interpretation als unsigned int mit Verschiebung ins negative (IEEE 754 Standard), schnellere Rechnungen möglich)
-

$\mathcal{F}(2, 6, 0, 15)$

0000000000

- Vorkommastelle
- Nachkommastellen
- Exponent

Beispiele

$\underbrace{1.11111 \cdot 2^{15}}_{\text{Grösste Zahl}}$ $\underbrace{0.00001 \cdot 2^{0}}_{\text{Kleinste Zahl}}$

- Wir brauchen negative Zahlen!
- Wir brauchen das erste Bit nicht zwingend wenn wir annehmen, dass es immer 1 ist.

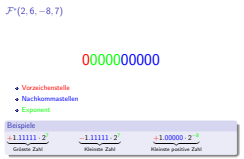
$\mathcal{F}(2, 6, -8, 7)$

0000000000

- Vorkommastelle
- Nachkommastellen
- Exponent

Beispiele

$\underbrace{1.11111 \cdot 2^7}_{\text{Grösste Zahl}}$ $\underbrace{0.00001 \cdot 2^{-8}}_{\text{Kleinste Zahl}}$



- Wir können 0 nicht mehr darstellen
- Ein Exponent für spezielle Zahlen: -8 = 0000
- Und die Nachkommastellen als Codierung für diese Zahlen

$\mathcal{F}^*(2,6,-8,7)$

0000000000

- Vorzeichenstelle
- Nachkommastellen
- Exponent

Beispiele

$\underbrace{+1.11111 \cdot 2^7}_{\text{Grösste Zahl}}$ $\underbrace{-1.11111 \cdot 2^7}_{\text{Kleinste Zahl}}$ $\underbrace{+1.00000 \cdot 2^{-8}}_{\text{Kleinste positive Zahl}}$

- Wir können 0 nicht mehr darstellen
- Ein Exponent für spezielle Zahlen: -8 = 0000
- Und die Nachkommastellen als Codierung für diese Zahlen
- Jetzt verstehen wir auch 32bit float und 64bit double

$\mathcal{F}^*(2,6,-7,7)$

$$0000000000$$

- Vorzeichenstelle
- Nachkommastellen
- Exponent

Beispiele

$\underbrace{0000000000}_{\text{Null}}$	$\underbrace{0000000001}_{+\infty}$	$\underbrace{0000000010}_{-\infty}$	$\underbrace{0000000011}_{\text{NaN}}$

Floating point systems following IEEE 754
- float (IEEE 745): $\mathcal{F}^*(2, 24, -126, 127)$
- double (IEEE 745): $\mathcal{F}^*(2, 53, -1022, 1023)$

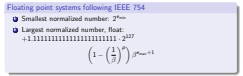
1. What is the largest possible normalized single and double precision floating point number?
2. What is the smallest possible normalized single and double precision floatig point number?

Exercise 4_2 ∼ 5'

Floating point systems following IEEE 754

- float (IEEE 745): $\mathcal{F}^*(2, 24, -126, 127)$
- double (IEEE 745): $\mathcal{F}^*(2, 53, -1022, 1023)$

1. What is the largest possible normalized single and double precision floating point number?
2. What is the smallest possible normalized single and double precision floatig point number?

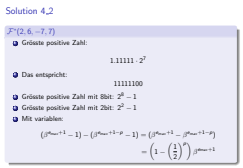


Solution 4_2

Floating point systems following IEEE 754

1. Smallest normalized number: $2^{e_{min}}$
2. Largest normalized number, float:
   $+1.11111111111111111111111 \cdot 2^{127}$

$$\left(1-\left(\frac{1}{\beta}\right)^{p}\right)\beta^{e_{max}+1}$$



Solution 4_2

$\mathcal{F}^*(2, 6, -7, 7)$

1. Grösste positive Zahl:

$$1.11111 \cdot 2^7$$

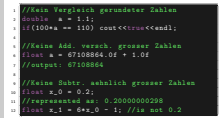
2. Das entspricht:

$$11111100$$

3. Grösste positive Zahl mit 8bit: $2^8 - 1$
4. Grösste positive Zahl mit 2bit: $2^2 - 1$
5. Mit variablen:

$$(\beta^{e_{max}+1} - 1) - (\beta^{e_{max}+1-p} - 1) = (\beta^{e_{max}+1} - \beta^{e_{max}+1-p})$$
$$= \left(1 - \left(\frac{1}{2}\right)^p\right) \beta^{e_{max}+1}$$

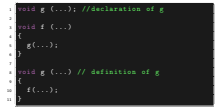
Tipps zu Fliesskommazahlen



Tipps zu Fliesskommazahlen

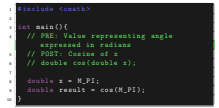
```
1  //Kein Vergleich gerundeter Zahlen
2  double  a = 1.1;
3  if(100*a == 110) cout<<true<<endl;
4
5  //Keine Add. versch. grosser Zahlen
6  float a = 67108864.0f + 1.0f
7  //output: 67108864
8
9  //Keine Subtr. aehnlich grosser Zahlen
10 float x_0 = 0.2;
11 //represented as: 0.20000000298
12 float x_1 = 6*x_0 - 1; //is not 0.2
```

Funktionsdefinition und -deklaration



Funktionsdefinition und -deklaration

```
1  void g (...); //declaration of g
2
3  void f (...)
4  {
5    g(...);
6  }
7
8  void g (...) // definition of g
9  {
10   f(...);
11 }
```



PRE- und POST-Bedingungen

```cpp
#include <cmath>

int main(){
  // PRE: Value representing angle
     expressed in radians
  // POST: Cosine of x
  // double cos(double x);

  double x = M_PI;
  double result = cos(M_PI);
}
```