

Lösungen zum Übungsblatt 5 zur Vorlesung „Technische Informatik I“ SS 2001
Strey / Guenkova-Luy / Prager

Aufgabe 1

a) Umrechnung IEEE 754 Gleitkommadarstellung (Single Precision) → Zahl.

$$0 \underbrace{|10000101|}_{e=133} \underbrace{|000000100000000000000000}_{f=2^{-7}}$$

$$x = (-1)^0 \cdot (1 + f) \cdot 2^{133-127} = (1 + 2^{-7}) \cdot 2^6 = 2^6 + 1/2$$

$$x = 64.5$$

$$1 \underbrace{|10000100|}_{e=132} \underbrace{|100100000000000000000000}_{f=2^{-1}+2^{-4}}$$

$$y = (-1)^1 \cdot (1 + 2^{-1} + 2^{-4}) \cdot 2^{132-127} = -(1 + 2^{-1} + 2^{-4}) \cdot 2^5 = -(2^5 + 2^4 + 2^1)$$

$$y = -50$$

b) Umrechnung Zahl → IEEE 754 Gleitkommadarstellung (Single Precision).

$$1 = (-1)^0 \cdot (1 + 0) \cdot 2^0 = (-1)^0 \cdot (1 + 0) \cdot 2^{127-127}$$

$$\Rightarrow s = 0, f = 0, e = 127$$

$$1 = \boxed{0|01111111|000000000000000000000000}$$

$$-5 = (-1)^1 \cdot (4 + 1) \cdot 2^0 = (-1)^1 \cdot (1 + 2^{-2}) \cdot 2^2$$

$$= (-1)^1 \cdot (1 + 2^{-2}) \cdot 2^{129-127}$$

$$\Rightarrow s = 1, f = 0.01, e = 129$$

$$-5 = \boxed{1|10000001|010000000000000000000000}$$

Der Zahl $x = 6.25 \cdot 10^{-3}$ sieht man nicht mehr wie in den beiden vorangegangenen Beispielen unmittelbar die Mantisse der Gleitkommadarstellung an. Da $x < 1$ bestimmen wir die Binärdarstellung von x durch wiederholte Multiplikation:

$$\begin{array}{rcl} 6.25 \cdot 10^{-3} & \cdot 2 & = 0 + 0.0125 \\ 0.0125 & \cdot 2 & = 0 + 0.025 \\ 0.025 & \cdot 2 & = 0 + 0.05 \\ 0.05 & \cdot 2 & = 0 + 0.1 \\ 0.1 & \cdot 2 & = 0 + 0.2 \\ 0.2 & \cdot 2 & = 0 + 0.4 \\ 0.4 & \cdot 2 & = 0 + 0.8 \\ 0.8 & \cdot 2 & = 1 + 0.6 \\ 0.6 & \cdot 2 & = 1 + 0.2 \end{array}$$

Die letzte Zeile liefert einen Rest 0.2, der schon 4 Zeilen darüber auftrat \Rightarrow die letzten vier Ziffern wiederholen sich periodisch.

$$\begin{aligned}
 6.25 \cdot 10^{-3} &= 0.0000\overline{0011} = 1.\overline{1001} \cdot 2^{-8} \\
 &= (-1)^0 \cdot 1.\overline{1001} \cdot 2^{119-127} \\
 &\approx (-1)^0 \cdot 1.100110011001100110011001\underline{1} \cdot 2^{119-127} \quad (\text{gerundet}) \\
 6.25 \cdot 10^{-3} &\approx \boxed{0|01110111|10011001100110011001101}
 \end{aligned}$$

Man beachte, dass die letzte Stelle hierbei aufgerundet wurde.

c) Zur Darstellung der Zahl 0 wird eine Ausnahmeregel benötigt, da

$$\forall f > 0, \forall p: \underbrace{(1+f)}_{>1} \cdot \underbrace{2^p}_{>0} > 0$$

d) $1 \cdot 10^{-41}$ kann als denormalisierte Zahl in einem Single Precision Float noch dargestellt werden. Die (betragsmäßig) kleinste denormalisierte Zahl ist:

$$(0|00000000|000000000000000000000001) = (0 + 2^{-23}) \cdot 2^{-126} = 2^{-149} \approx 1.4 \cdot 10^{-45}$$

Große Zahlen sind grundsätzlich normiert, daher gibt es zur genauen Darstellung von $1 \cdot 10^{41}$ keine Möglichkeit - von der sehr groben Approximation $+\infty$ einmal abgesehen.

e) Addition/Subtraktion $c = a + b$

$$\begin{aligned}
 a &= 0|1000\ 0101|0000\ 0010 \dots 0 \\
 1.f_a &= 1.0000\ 0010 \dots 0 \\
 b &= 1|1000\ 0100|1001\ 0 \dots 0 \\
 1.f_b &= 1.1000\ 0100 \dots 0
 \end{aligned}$$

Der Exponent der Zahl a ist um 1 größer als der von b . Daher müssen wir $1.f_b$ um eine Stelle nach rechts schieben. Die Zahl b ist zudem negativ, so daß wir vor der Addition der Mantissen noch das 2er-Komplement bilden müssen: Hierbei beachten wir, dass prinzipiell bei der Addition der Mantissen eine Zahl entstehen kann, die eine Stelle mehr als die beiden Summanden hat (also 2 Stellen vor dem Komma. Das Vorzeichen codieren wir in der 3. Stelle vor dem Komma:

$$\begin{array}{rll}
 0.1f_b &= & 000.1100\ 1000 \dots 0 & (\text{Stellen erweitert}) \\
 && 111.0011\ 0111 \dots 1 & (\text{1-er Komplement}) \\
 && 111.0011\ 1000 \dots 0 & (+1 \cdot 2^{-23} \Rightarrow \text{2-er-Komplement}) \\
 + && 001.0000\ 0010 \dots 0 & \\
 \hline
 && 000.001\ \underbrace{1\ 1010 \dots 0}_{f_c} & (\text{Übertrag an der höchsten Stelle ignoriert})
 \end{array}$$

Das Ergebnis hat die erste 1 von links gesehen an der 3. Stelle nach dem Komma, daher schieben wir es um 3 Stellen nach links und subtrahieren 3 vom Exponenten des Ergebnisses (Normalisierung). Wir erhalten:

$$a + b = \boxed{0|10000010|1101000000000000000000}$$

f) Ausnahme-Codierungen:

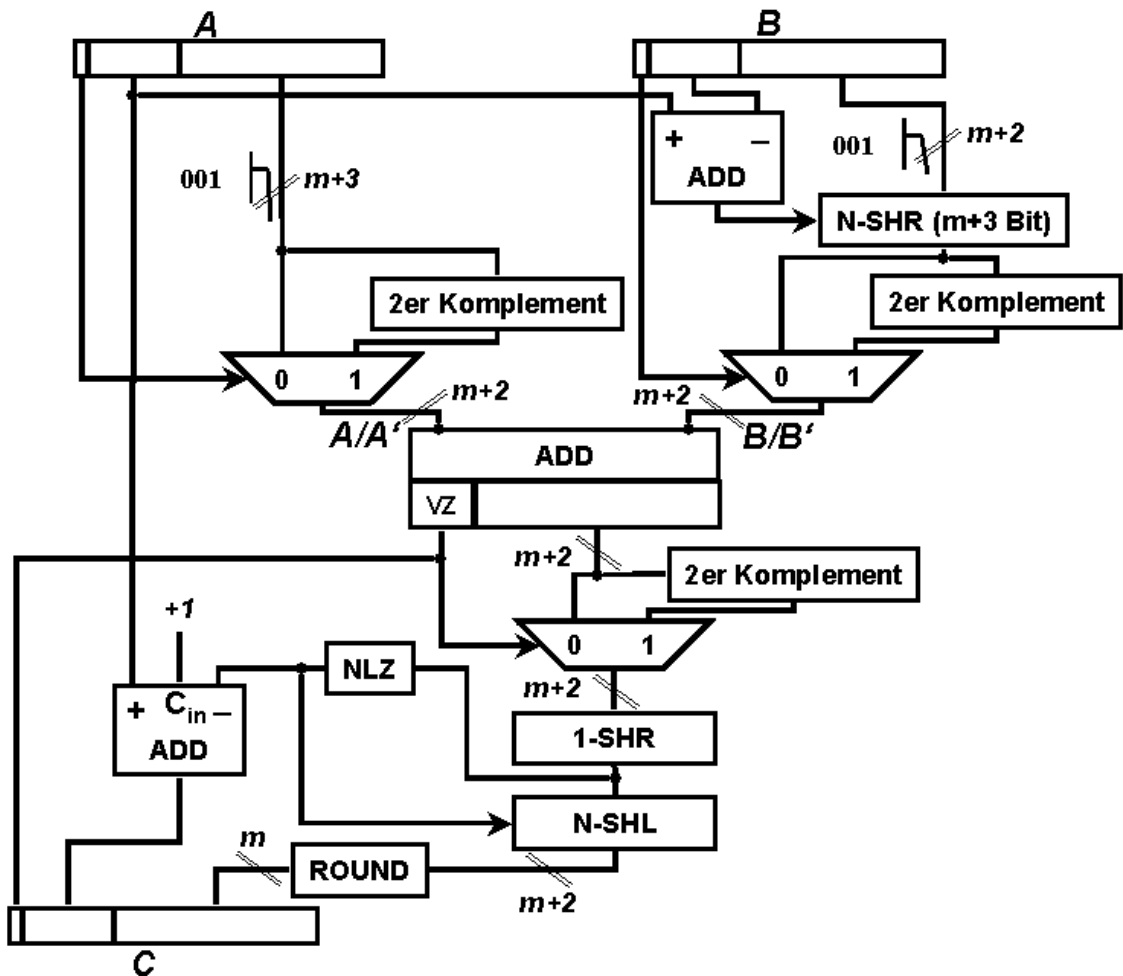
$$\sqrt{-13} = NaN$$

$$\log_{10}(0) = -\infty$$

$$\log_{10}(-1) = NaN$$

Aufgabe 2

a) Gleitkomma-Addierwerk



b) Division von Gleitkommazahlen.

$$a = (-1)^{s_a} \cdot 1.f_a \cdot 2^{e_a - bias}$$

$$b = (-1)^{s_b} \cdot 1.f_b \cdot 2^{e_b - bias}$$

- Dividiere die Mantissen $\gamma = 1.f_a / 1.f_b$
- Addiere die Exponenten $e_c = e_a - e_b + bias$
- Vorzeichen $s_c = s_a \oplus s_b$

- (d) Normalisierung. $\gamma \in (\frac{1}{2}, 2)$
 Falls $\gamma \leq 1$, dann schiebe y um eine Stelle nach links und erhöhe den Exponenten e_c um eins. Setze $1.f_c := \gamma$
- (e) Sonderfälle
- Überlauf, falls $e_c \geq s^p - 1 \Rightarrow \pm\infty$.
 - Unterlauf, falls $e_c \leq 0 \Rightarrow$ Denormalisierung, bzw. Abbildung auf ± 0 .
 - $(-1)^{s_c} \cdot \infty$, falls $b = 0, a \neq 0$
 NaN oder „indeterminate“, falls $a = b = 0$
 - Weitere Sonderfälle, wenn (mindestens) einer der Operanden schon eine Ausnahmehedingung darstellt.

Aufgabe 3

- a) Die kleinste natürliche Zahl $l_{float} = 1.f \cdot 2^n$, die im IEEE Single Precision Format **nicht** darstellbar ist hat folgende Eigenschaft:

$l_{float} - 1$ hat einen Exponenten n , der so groß ist, dass ein erhöhen der Mantisse um 1 auf der niedrigsten Stelle (2^{-23}) einen erhöhen des Ergebnisses um einen Wert > 1 bewirkt.

$$2^{-23} \cdot 2^n > 1 \Leftrightarrow 2^n > 2^{23} \Leftrightarrow n > 23$$

Wir schließen daraus, daß $l_{float} - 1$ einen Exponenten $n = 24$ hat. Die kleinste Zahl mit diesem Exponenten ist $1.0 \cdot 2^{24}$ und damit $l_{float} = 2^{24} + 1$ oder anders formuliert $l_{float} = (1 + 2^{-24}) \cdot 2^{24}$.

- b) Das Programm funktioniert nicht, weil das gewünschte Inkrementieren der Variablen x im Runden der Mantisse untergeht - x wird niemals inkrementiert.
- c) Verwendung eines genaueren Gleitkommatyps, z.B. `double`. $l_{double} = 2^{53} + 1$, d.h. es sind nun ganze Zahlen bis ca. $9 \cdot 10^{15}$ exakt darstellbar. Alternativ: Verwendung von Ganzzahltypen wie `int` oder `long`.
- d) Die relative Genauigkeit ist konstant. Da wir nun die 10-fache absolute Genauigkeit benötigen, können wir auch nur Zahlen $< \frac{1}{10} \cdot l_{float}$ nutzen.

Aufgabe 4

Allen Befehlen gemeinsam ist die Fetch-Phase:

```
[MAR] ← [PC]
[PC]   ← [PC] + 1
[MBR]  ← [M(MAR)]
[IR]   ← [MBR]
CU     ← [IR(Opcode)]
```

Daran schließen sich die folgenden Execute-Phasen an:

- a) SUB #17,D0

```
[MBR] ← [IR(Addr)]
ALU   ← [MBR]
ALU   ← [D0]
[D0]  ← ALU
```

- b) BRA 15

```
[PC] ← [IR(Addr)]
```

c) BMI 35

$[PC] \leftarrow [IR(Addr)],$ falls $[N] = 1$

Aufgabe 5

```
        MOVE x,D0
        BEQ else           ; jump to else, if x=0
        SUB #16,D0
        BPL else           ; jump to else, if x-16>=0
then:    MOVE a,D0
        MUL x,D0
        MOVE D0,a
        BRA done
else:    MOVE a,D0
        ADD #1,D0
        MOV D0,a
done:    STOP
```

Die erste Bedingung $x \neq 0$ lässt sich 1:1 in Maschinenbefehle umsetzen. Die Bedingung $x \leq 15$ ist bei ganzen Zahlen äquivalent mit $x \leq 16$ oder $x - 16 < 0$. Der letzte Ausdruck lässt sich wieder mit zwei Maschinenbefehlen umsetzen.