

Aufgabe 1:

- a) Bestimmen Sie die Darstellung der Zahl 113_{10} zur Basis 7.

Lösung: $113_{10} = 221_7$ (Sehen Sie auch Aufgabe 1.b)

- b) Wenden Sie die Berechnungsmethode „sukzessive Division mit Rest“ in aller Ausführlichkeit auf eine beliebige Zahl $z = z_2 \cdot 7^2 + z_1 \cdot 7^1 + z_0 \cdot 7^0$ an. Berechnen Sie die Darstellung zur Basis 7 in Abhängigkeit von z_2, z_1, z_0 .

Lösung:

$$(z_2 \cdot 7^2 + z_1 \cdot 7^1 + z_0 \cdot 7^0) : 7 = (z_2 \cdot 7^1 + z_1 \cdot 7^0) \rightarrow \text{Rest } z_0$$

$$(z_2 \cdot 7^1 + z_1 \cdot 7^0) : 7 = (z_2 \cdot 7^0) \rightarrow \text{Rest } z_1$$

$$(z_2 \cdot 7^0) : 7 = 0 \rightarrow \text{Rest } z_2$$



Die Zahl ist: (z_2, z_1, z_0)

- c) Bestimmen Sie Darstellungen der Zahl 113_{10} im Hexadezimalsystem (Basis $b=16$), im Binärsystem (Basis $b=2$), im Oktalsystem (Basis $b=8$) sowie im 12er-System. Wie kann die Umwandlung zwischen den Darstellungen der Basen 2, 8, 16 vereinfacht werden?

Lösung: $113_{10} = 71_{16}$, $113_{10} = 161_8$, $113_{10} = 1110001_2$, $113_{10} = 95_{12}$

$(0111|0001)_2 \rightarrow (7|1)_{16}$ – Zusammenfassung von 4er Blöcken der Binärzahl

$(001|110|001)_2 \rightarrow (1|6|1)_8$ – Zusammenfassung von 3er Blöcken der Binärzahl

- d) Bestimmen Sie die 6-stellige 2-er-Komplement Binärdarstellung der Zahl -17_{10} . Wie sieht die 8-stellige Darstellung dieser Zahl aus?

Lösung: $17_{10} = 010001_2$ (6-Stellig, positive)

101110_2 (1-er Komplement)

101111_2 (2-er Komplement)

11101111_2 (2-er Komplement mit 8 Stellen, Vorzeichenerweiterung)

- e) Bestimmen Sie die Binärdarstellung der Zahlen 0.375_{10} und 0.1_{10} . Wie viele Nachkommastellen werden zur exakten Darstellung jeweils benötigt?

Lösung: $0.375_{10} = 0.011_2$, $0.1_{10} \approx 0.0(0011)_2$ – Der Ausdruck in den Klammern wiederholt sich periodisch.

- f) Wie lautet die 2-er-Komplement Binärdarstellung der Zahl -4.375_{10} bei Verwendung von $k=3$ Vorkommastellen und $m=4$ Nachkommastellen?

Lösung: $4.375_{10} = 0100,0110$ (Mit Vorzeichenbit)

1-er Komplement: $-4.375_{10} = 1011.1001_2$

2-er Komplement: $-4.375_{10} = 1011.1010_2$

Aufgabe 2:

Forscher des SETI-Projekts haben aus den Tiefen des Universums eine Botschaft intelligenten Lebens empfangen. Die Wesen haben offensichtlich 3 Hände mit jeweils 5, 2 und 5 Fingern pro Hand. Zur Darstellung nichtnegativer ganzer Zahlen $\{0, 1, \dots\}$ verwenden Sie die positionale Notation $(z_2, z_1, z_0)_\psi$, welche sich hervorragend zum Zählen mit den Fingern eignet. Die Forscher kamen zu dem Schluss, dass gilt:

$$z = z_0 + 6 \cdot z_1 + b_2 \cdot z_2 \text{ mit } z_0, z_2 \in \{0..5\}, z_1 \in \{0, 1, 2\},$$

Allein die Zahl b_2 vermochten Sie nicht zu bestimmen.

a) Wie wird die Zahl b_2 von den Wesen sinnvollerweise gewählt worden sein, um mit ihren Fingern einem möglichst großen, lückenlosen Wertebereich zählen zu können?

Lösung: $b_2 = z_{0\max} + 6 \cdot z_{1\max} + 1$, $b_2 = 18$

b) Welchen Zahlenbereich kann man mit dieser Codierung abdecken?

Lösung: $\max = 525_{\psi} = 107_{10}$ (Insgesamt 107 Zahlen (0..107) – Maximum)

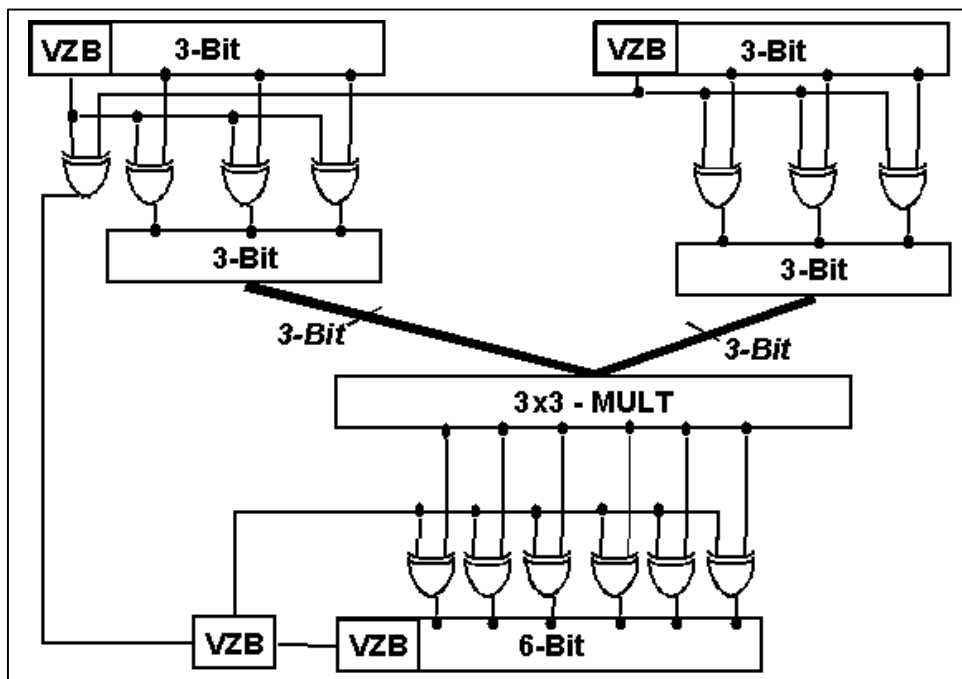
c) Berechnen Sie $(z_2, z_1, z_0)_{\psi} = 315_{\psi} + 113_{\psi}$. Eignet sich diese Zahlendarstellung zur Berechnung von Summen?

Lösung: $502_{\psi} = 92_{10}$, Für Nichteinheimischen – schwierig!!! Aber prinzipiell zur stellenweisen Addition geeignet.

Aufgabe 3:

Gegeben sei ein 3x3-Bit Multiplizierer für vorzeichenlose Zahlen. Entwerfen Sie eine Schaltung, die unter Zuhilfenahme dieses Multiplizierers sowie zusätzlichen Gattern vorzeichenbehaftete 4-Bit Zahlen multiplizieren kann. Sowohl Eingabe als auch Ausgabe sollen im 1-er Komplement dargestellt sein. Welche Wortbreite sollten Sie für die Ausgabe mindestens vorsehen, um Fehler durch Überläufe zu vermeiden?

Lösung:



Aufgabe 4:

Eine IEEE 754 32-Bit Gleitkommazahl hat folgendes Format: $x = (-1)^s \cdot (1.f) \cdot 2^{e-127}$ wobei die einzelnen Bits ($b_{31}..b_0$) in einem Rechner in folgender Reihenfolge abgelegt werden: ($s, e_7, .. e_0, f_{-1}, .. f_{-23}$). Details zu diesem Format finden Sie auf den Vorlesungsfolien.

a) Welche Zahlen werden durch die Bitmuster $0|10000101|000000100000000000000000$ und $1|10000100|100100000000000000000000$ dargestellt? (Die senkrechten Striche dienen hierbei lediglich der Übersicht.)

Lösung:

$x = 0|10000101|000000100000000000000000$

$s = 0$

$$e=133 \text{ (10000101}_2=133_{10})$$

$$f=2^{-7} \text{ (000000100000000000000000}_2=(2^{-7})_{10})$$

$$x = (-1)^0 \cdot (1 + f) \cdot 2^{133-127} = (-1)^0 \cdot (1 + 2^{-7}) \cdot 2^6 = 1 \cdot (2^6 + 2^{-1}) = 64 + \frac{1}{2} = 64.5$$

$$y=1|10000100|100100000000000000000000$$

$$s=1$$

$$e=132 \text{ (10000100}_2=132_{10})$$

$$f=2^{-1}+2^{-4}$$

$$y = (-1)^1 \cdot (1 + 2^{-1} + 2^{-4}) \cdot 2^{132-127} = -(1 + 2^{-1} + 2^{-4}) \cdot 2^5 = -(2^5 + 2^4 + 2^1) = -50$$

- b) Bestimmen Sie die Gleitkommadarstellung der Zahlen 1, -5 und $6.25 \cdot 10^{-3}$.

Lösung:

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

$$s=0; f=0; e=127$$

$$1=0|01111111|000000000000000000000000$$

$$-5 = (-1)^1 \cdot (1 + 4) \cdot 2^0 = (-1)^1 \cdot (1 + 2^{-2}) \cdot 2^2 = (-1)^1 \cdot (1 + 2^{-2}) \cdot 2^{129-127}$$

$$s=1; f=0.01_2; e=129$$

$$-5=1|10000001|010000000000000000000000$$

Die Zahl $6.25 \cdot 10^{-3}$ muss zuerst umgewandelt werden, um die Mantisse für die Gleitkommaform darstellen zu können. Da diese Zahl <1 muss man zuerst die Binärdarstellung der Zahl $6.25 \cdot 10^{-3}$ haben (Benutzen Sie dazu wiederholte Multiplikation.).

$$6.25 \cdot 10^{-3} \cdot 2 = 0 + 0.0125$$

$$0.0125 \cdot 2 = 0 + 0.025$$

$$0.025 \cdot 2 = 0 + 0.05$$

$$0.05 \cdot 2 = 0 + 0.1$$

$$0.1 \cdot 2 = 0 + 0.2$$

$$0.2 \cdot 2 = 0 + 0.4$$

$$0.4 \cdot 2 = 0 + 0.8$$

$$0.8 \cdot 2 = 1 + 0.6$$

$$0.6 \cdot 2 = 1 + 0.2$$

Diese Zahl ist periodisch (Siehe auch Aufgabe 1.e).

$$6.25 \cdot 10^{-3}_{10} = 0.0000(0011)_2 = 1 \cdot (1001) \cdot 2^{-7}$$

$$6.25 \cdot 10^{-3} = (-1)^0 \cdot 1.10011001100110011001101 \cdot 2^{120-127}$$

$$6.25 \cdot 10^{-3} = 0|01111000|10011001100110011001101$$

Die letzte Stelle der Zahl $6.25 \cdot 10^{-3}$ ist aufgerundet.

- c) Warum wird zur Darstellung der Zahl 0 eine Ausnahmeregel benötigt?

Lösung: Nach der IEEE 754 Standard die Formel für die Berechnung der Mantissen $(1+f) \cdot 2^p$ kann nur Zahlen >0 darstellen ($1+f>0$, $2^p>0$), deswegen braucht man Sonderdarstellung für Null.

- d) Warum ist es in Programmiersprachen möglich, die Zahl $1 \cdot 10^{-41}$ erfolgreich in einer 32-Bit Gleitkommazahl abzulegen, wohingegen das Zuweisen von $1 \cdot 10^{41}$ vom Übersetzer zurückgewiesen wird?

Aufgabe 5:

Die Programmiersprache Java kennt 2 (primitive) Typen von Gleitkommazahlen gemäß dem IEEE 754-Standard: float ("Single Precision") und double ("Double Precision"). Ein verzweifelter Programmierer legt Ihnen folgendes Fragment eines Java-Programms vor, welches die Werte einer Funktion `some_function()` in einem Zahlenbereich aufsummieren soll:

```
float integrate() {
    float sum = 0;
    for (float x = 20000000; x < 20000010; x = x + 1) {
        sum = sum + some_function(x);
    }
    return sum;
}
```

Der Programmierer beklagt sich, dass sich das Programm grundsätzlich bei einem Aufruf der Methode `integrate` „aufhängt“, d.h. in eine Endlosschleife gerät. Können Sie ihm weiterhelfen?

- a) Bestimmen Sie die kleinste natürliche Zahl l_{float} , die im IEEE Single Precision Format **nicht** darstellbar ist.

Lösung:

Die kleinste natürliche Zahl $l_{float} = (1.f) \cdot 2^n$, die in 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}), ein Erhöhen des Ergebnisses um einen Wert > 1 bewirkt.

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

Daher $l_{float} - 1$ hat einen Exponenten $n=24$ und die kleinste mit diesem n darstellbare Zahl ist $(1.0) \cdot 2^{24}$ oder $l_{float} = (1+2^{-24}) \cdot 2^{24}$

- b) Warum funktioniert das Programmfragment nicht wie erwartet?

Lösung: Die letzten Stellen der Mantisse der Variablen X gehen verloren bei der Rundung der Mantisse nach der Summation, daher wird X im Programm nie inkrementiert.

- c) Wie können Sie obiges Programm modifizieren, damit es die gewünschte Funktionalität bietet? Innerhalb welcher Grenzen für die Laufvariable x funktioniert das Programm nun korrekt?

Lösung: 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 `int` oder `long`.

In Java:

`double` = Double Precision Format (64-bit IEEE 754)

`int` = 32-Bit 2-er Komplement = $2^{32} = \text{ca. } 4 \cdot 10^9$

`long` = 64-Bit 2-er Komplement = $2^{64} = \text{ca. } 2 \cdot 10^{19}$