TI II

Sommersemester 2010
PD Dr. Katinka Wolter

2. Aufgabenblatt mit Lösungen

Problem 1: (6*1 = 6)

Geben Sie für jede der folgenden Zahlen deren Ziffernschreibweisen im Dezimal-, Dual-, Oktal- und Hexadezimal-System an.

a) $(2748)_{10}$

b) $(1010011011)_2$

c) $(52056)_8$

d) $(D1)_{16}$

Rechnen Sie die folgende Zahl ins 10er-System um:

e) $(2161)_7$

Rechnen Sie die folgende Zahl ins 7er-System um:

f) $(2161)_{10}$

Solution 1

1. $(2748)_{10}$

   (a) Dual-System mit Horner-Schema

$$2748/2 = 1374 \text{ R } 0$$
$$1374/2 = 687 \text{ R } 0$$
$$687/2 = 343 \text{ R } 1$$
$$343/2 = 171 \text{ R } 1$$
$$171/2 = 85 \text{ R } 1$$
$$85/2 = 42 \text{ R } 1$$
$$42/2 = 21 \text{ R } 0$$
$$21/2 = 10 \text{ R } 1$$
$$10/2 = 5 \text{ R } 0$$
$$5/2 = 2 \text{ R } 1$$
$$2/2 = 1 \text{ R } 0$$
$$1/2 = 0 \text{ R } 1$$

   Damit lautet das Ergebnis: $(101010111100)_2$

(b) Oktal-System ebenfalls mit Horner-Schema

$$2748/8 = 343\text{R}4$$

$$343/8 = 42\text{R}7$$

$$42/8 = 5\text{R}2$$

$$5/8 = 0\text{R}5$$

Damit lautet das Ergebnis: $(5274)_8$

(c) Hexadezimal-System mit Horner-Schema

$$2748/16 = 171\text{R}12$$

$$171/16 = 10\text{R}11$$

$$10/16 = 0\text{R}10$$

Damit lautet das Ergebnis: $(\text{ABC})_{16}$

2. $(1010011011)_2$

(a) Oktal-System

$$\underbrace{1}_{1}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{011}_{3}$$

(b) Hexadezimal-System

$$\underbrace{10}_{2}\ \underbrace{1001}_{9}\ \underbrace{1011}_{\text{B}}$$

(c) Dezimal-System

$$\underbrace{2*16^2}_{512} + \underbrace{9*16^1}_{144} + \underbrace{11*16^0}_{11} = 667$$

3. $(52056)_8$

(a) Dual-System

$$\underbrace{5}_{101}\ \underbrace{2}_{010}\ \underbrace{0}_{000}\ \underbrace{5}_{101}\ \underbrace{6}_{110}$$

(b) Hexadezimal-System

$$\underbrace{101}_{5}\ \underbrace{0100}_{4}\ \underbrace{0010}_{2}\ \underbrace{1110}_{\text{E}}$$

(c) Dezimal-System

$$\underbrace{5*16^3}_{20480} + \underbrace{4*16^2}_{1024} + \underbrace{2*16^1}_{32} + \underbrace{14*16^0}_{14} = 21550$$

4. $(\text{D}1)_16$

(a) Dual-System: 11010001_2

(b) Oktal-System: 321_8

(c) Dezimal-System: 209_{10}

5. $(2161)_7$ ins Dezimal-System

$$2*7^3 + 1*7^2 + 6*7^1 + 1*7^0 = 778_{10}$$

6. $(2161)_{10}$ ins 7-System. Wie bei der Umwandelung von Dezimal ins {Oktal, Hexadezimal, Dual}-System. Es wird hier nur durch 7 statt $\{8, 16, 2\}$ geteilt. Ergebnis: 6205_7

$$2161/7 = 308\text{R}5$$

$$308/7 = 2\text{R}0$$

$$44/7 = 1\text{R}2$$

$$6/7 = 0\text{R}6$$

Problem 2: (14*1/2 + 1 = 8)

Im folgenden sei die Wortlänge gleich 8 (d. h.: es wird mit Bytes gearbeitet).

a) i) Wie ist die Darstellung von -50 im Zweier-Komplement?
ii) Wie ist die Darstellung von -62 im Einer-Komplement?
iii) Wie ist die Darstellung von $+44$ im Einer-Komplement?
iv) Berechnen Sie $76-44$ im Einer-Komplement.
v) Berechnen Sie $116-29$ im Zweier-Komplement.
vi) Welche Zahl wird durch 183 im Zweier-Komplement dargestellt?
vii) Welche Zahl wird durch 186 im Einer-Komplement dargestellt?
viii) Berechnen Sie $44-76$ im Zweier-Komplement.
ix) Berechnen Sie $-116+29$ im Einer-Komplement.
x) Berechnen Sie $-35-(-100)$ im Zweier-Komplement.
xi) Berechnen Sie $-18-(-100)$ im Einer-Komplement.
xii) Wie ist die Darstellung von -88 im Einer-Komplement?
xiii) Wie ist die Darstellung von -45 im Zweier-Komplement?
xiv) Berechnen Sie $-47-16$ im Zweier-Komplement.

b) Interpretieren Sie die Beträge[1] der Ergebnisse des vorigen Aufgabenteils der Reihe nach als ASCII-Codierungen mit ungerader Parität[2]. Korrigieren Sie eventuell falsche Paritätsbits.

Solution 2

a) **i)** -50 **im Zweier-Komplement:**

$$\begin{aligned} 50_{10} &\rightarrow 00110010_2 && \text{(Umwandlung in Binärdarstellung, auf 8 bit)} \\ &\rightarrow 11001101_2 && \text{(Negierung der Stellen, Einer-Komplement)} \\ &\rightarrow 11001110_2 && \text{(addiere 1 hinzu, Zweier-Komplement)} \\ & && \text{('-50' als Zweier-Komplement)} \end{aligned}$$

ii) -62 **im Einer-Komplement:**

$$\begin{aligned} 62_{10} &\rightarrow 00111110_2 && \text{(Umwandlung in Binärdarstellung, auf 8 bit)} \\ &\rightarrow 11000001_2 && \text{(Negierung der Stellen, Einer-Komplement)} \\ & && \text{('-62' als Einer-Komplement)} \end{aligned}$$

[1] Für den Betrag $|z|$ einer Zahl z gilt: $|z| = z$, falls $z \geqslant 0$; $|z| = -z$, falls $z < 0$.

[2] „ASCII"=Code ungerade Parität" bedeutet: In einem Byte wird das Paritätsbit P jeweils so gewählt (0 oder 1), dass sich eine ungerade Anzahl von Einsen im Byte ergibt, z. B.: $P1000111 \Rightarrow P = 1$, $P1001001 \Rightarrow P = 0$.

iii) 44 **im Einer-Komplement:**

$44_{10} \rightarrow 00101100_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
('+44' als Einer-Komplement)

iv) Berechnung von $+76-44$ **im Einer-Komplement:**

$76_{10} \rightarrow 01001100_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
('+76' als Einer-Komplement)
$44_{10} \rightarrow 00101100_2$ (siehe iii))
$\rightarrow 11010011_2$ (Negierung der Stellen, Einer-Komplement)
('-44' als Einer-Komplement)

```
              01001100
+             11010011
Übertrag     11
              --------
Zw.Ergebnis   00011111

Da ein Überlauf aufgetreten ist, wird noch 1 hinzuaddiert.

+                    1
              --------
Endergebnis   00100000

Das Ergebnis ist 32.
```

v) Berechnung von $+116-29$ **im Zweier-Komplement:**

$116_{10} \rightarrow 01110100_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
('+116' als Zweier-Komplement)
$29_{10} \rightarrow 00011101_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 11100010_2$ (Negierung der Stellen, Einer-Komplement)
$\rightarrow 11100011_2$ (1 hinzuaddieren, Zweier-Komplement)
('-29' als Zweier-Komplement)

```
              01110100
+             11100011
Übertrag     111
              --------
Endergebnis   01010111

Das Ergebnis ist 87.
```

vi) Bitmuster von 183 **im Zweier-Komplement:**

$183_{10} \rightarrow 10110111_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 10110110_2$ (ziehe 1 ab)
$\rightarrow 01001001_2$ (negiere die Stellen)
$\rightarrow 73_{10}$ (entspricht also der -73)

vii) Bitmuster von 186 **im Einer-Komplement:**

$186_{10} \rightarrow 10111010_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 01000101_2$ (negiere die Stellen)
$\rightarrow 69_{10}$ (entspricht also der -69)

viii) Berechnung von $44-76$ **im Zweier-Komplement:**

$44_{10} \rightarrow 00101100_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
('44' als Zweier-Komplement)
$76_{10} \rightarrow 01001100_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 10110011_2$ (Negierung der Stellen, Einer-Komplement)
$\rightarrow 10110100_2$ (1 hinzuaddieren, Zweier-Komplement)
('-76' als Zweier-Komplement)

```
              00101100
+             10110100
Übertrag       1111
              --------
Endergebnis   11100000

Das Ergebnis ist -32.
```

ix) Berechnung von $-116+29$ **im Einer-Komplement:**

$116_{10} \rightarrow 01110100_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 10001011_2$ (Negierung der Stellen, Einer-Komplement)
('-116' als Einer-Komplement)
$29_{10} \rightarrow 00011101_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
('+29' als Einer-Komplement)

```
              10001011
+             00011101
Übertrag        11111
              --------
Endergebnis   10101000

Das Ergebnis ist -87.
```

x) Berechnung von $-35-(-100)$ im Zweier-Komplement:

35_{10}	$\rightarrow 00100011_2$	(Umwandlung in Binärdarstellung, auf 8 bit)
	$\rightarrow 11011100_2$	(Negierung der Stellen, Einer-Komplement)
	$\rightarrow 11011101_2$	(1 hinzuaddieren, Zweier-Komplement)
		('-35' als Zweier-Komplement)
100_{10}	$\rightarrow 01100100_2$	(Umwandlung in Binärdarstellung, auf 8 bit)
	$\rightarrow 10011011_2$	(Negierung der Stellen, Einer-Komplement)
	$\rightarrow 10011100_2$	(1 hinzuaddieren, Zweier-Komplement)
		('-100' als Zweier-Komplement)
	$\rightarrow 01100011_2$	(Negierung der Stellen, Einer-Komplement)
	$\rightarrow 01100100_2$	(1 hinzuaddieren, Zweier-Komplement)
		('-(-100)' als Zweier-Komplement)

```
              11011101
+             01100100
Übertrag     111111
              --------
Endergebnis   01000001

Das Ergebnis ist 65.
```

xi) Berechnung von $-18-(-100)$ im Einer-Komplement:

18_{10}	$\rightarrow 00010010_2$	(Umwandlung in Binärdarstellung, auf 8 bit)
	$\rightarrow 11101101_2$	(Negierung der Stellen, Einer-Komplement)
		('-18' als Einer-Komplement)
100_{10}	$\rightarrow 01100100_2$	(Umwandlung in Binärdarstellung, auf 8 bit)
	$\rightarrow 10011011_2$	(Negierung der Stellen, Einer-Komplement)
		('-100' als Einer-Komplement)
	$\rightarrow 01100100_2$	(Negierung der Stellen, Einer-Komplement)
		('-(-100)' als Einer-Komplement)

```
              11101101
+             01100100
Übertrag     111 11
              --------
Zw.ergebnis   01010001

Da ein Überlauf aufgetreten ist, wird noch 1 hinzuaddiert.

+                    1
              --------
Endergebnis   01010010

Das Ergebnis ist 82.
```

xii) -88 **im Einer-Komplement:**

$88_{10} \rightarrow 01011000_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 10100111_2$ (Negierung der Stellen, Einer-Komplement)
('-88' als Einer-Komplement)

xiii) -45 **im Zweier-Komplement:**

$45_{10} \rightarrow 00101101_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 11010010_2$ (Negierung der Stellen, Einer-Komplement)
$\rightarrow 11010011_2$ (addiere 1 hinzu, Zweier-Komplement)
('-45' als Zweier-Komplement)

xiv) Berechnung von $-47-16$ **im Zweier-Komplement:**

$47_{10} \rightarrow 00101111_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 11010000_2$ (Negierung der Stellen, Einer-Komplement)
$\rightarrow 11010001_2$ (addiere 1 hinzu, Zweier-Komplement)
('-47' als Zweier-Komplement)
$16_{10} \rightarrow 00010000_2$ (Umwandlung in Binärdarstellung, auf 8 bit)
$\rightarrow 11101111_2$ (Negierung der Stellen, Einer-Komplement)
$\rightarrow 11110000_2$ (1 hinzuaddieren, Zweier-Komplement)
('-16' als Zweier-Komplement)

```
              11010001
              11110000
Übertrag     1111
              --------
Endergebnis   11000001

Das Ergebnis ist -63.
```

b) Man kann diese Aufgabe auf zwei Arten interpretieren.

   (a) Man nimmt einfach das Bitmuster und sieht das höchstwertigste Bit (das Vorzeichen) einfach als Paritätsbit. Den Rest sieht man als Betrag.

   (b) Wenn die Zahl negativ ist, wird sie wieder negiert (Betrag gebildet) und passt dann das höchstwertigste Bit als Paritätsbit entsprechend an.

Im folgender Tabelle markiert $*$ die Bitmuster bei dem das Paritätsbit gekippt werden muss.

Aufgabe	dezimal	binär a)	binär b)
i)	-50_{ZK}	11001110_2	00110010_2
ii)	-62_{EK}	11000001_2*	00111110_2
iii)	44_{EK}	00101100_2	siehe a)
iv)	32_{EK}	00100000_2	siehe a)
v)	87_{ZK}	01010111_2	siehe a)
vi)	-73_{ZK}	10110111_2*	01001001_2
vii)	-69_{EK}	10111010_2	01000101_2
viii)	-32_{ZK}	11100000_2	00100000_2
ix)	-87_{EK}	10101000_2	01010111_2
x)	65_{ZK}	01000001_2*	01000001_2*
xi)	82_{EK}	01010001_2	01010010_2
xii)	-88_{EK}	10100111_2	01011000_2
xiii)	-45_{ZK}	11010011_2	00101101_2*
xiv)	-63_{ZK}	11000001_2	00111111_2*

Problem 3: (3*2 = 6)

Gegeben sei die Boolesche Funktion[3]

$$f(x_1,x_2,x_3) \;=\; (x_1 \nleftrightarrow x_2)\cdot(\overline{x}_1 + x_3)$$

a) Stellen Sie die Funktionstabelle auf, und bestimmen Sie die einschlägigen Indizes.

b) Bestimmen Sie die DNF und die KNF von f.

c) Welche der beiden Darstellungen ist „günstiger“? Begründen Sie Ihre Antwort.

[3]Zur Erinnerung: $x \nleftrightarrow y = (x+y)\cdot\overline{(x\cdot y)} = (x+y)\cdot(\overline{x}+\overline{y}) = (x\cdot\overline{y})+(\overline{x}\cdot y)$ („XOR“).

Solution 3

Musterlösung Übungszettel 2

Aufgabe 3

3a.)

x_1	x_2	x_3	$(x_1 \dot{\vee} x_2)$	$(\overline{x_1} + x_3)$	$(x_1 \dot{\vee} x_2) \cdot (\overline{x_1} + x_3)$
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	1	0

$\dot{\vee} \equiv XOR$

3b.)

Die DNF ergibt sich aus der obigen Tabelle. Es werden alle Minterme mit einem „ODER“ verbunden:

$DNF: (\overline{x_1} \cdot x_2 \cdot \overline{x_3}) + (\overline{x_1} \cdot x_2 \cdot x_3) + (x_1 \cdot \overline{x_2} \cdot x_3)$

Für die KNF werden alle Maxterme mit einem „UND“ verknüpft:

$KNF: (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + \overline{x_3}) \cdot (\overline{x_1} + x_2 + x_3) \cdot (\overline{x_1} + \overline{x_2} + x_3) \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3})$

3c.)

Bei dieser Aufgabe ist die DNF „günstiger“, weil für ein Ergebnis weniger Terme zu berechnen sind.

Problem 4: Gleitkomma-Darstellung (2+2+2+2+2+2=12)

Gehen Sie bei der binären Gleitkommadarstellung von 2-Byte großen Zahlen aus. Der Charakteristik stehen 4 Bit zur Verfügung, der Mantisse 11 und ein Vorzeichenbit. Geben Sie folgende Zahlen in binärer Gleitkomma- und in der normalen dezimalen Festkommadarstellung an, bzw. berechnen Sie den nächsten Nachbarn, falls die Zahl selbst nicht darstellbar ist:

a) die kleinste in dieser binären Gleitkommadarstellung darstellbare Zahl.

b) die größte in dieser binären Gleitkommadarstellung darstellbare Zahl.

c) $11/10$

d) $17+2^{-3}$

e) 7^7

f) $0,F_{16}$

Solution 4

(i) Variante 1 aus dem Skript: 0,M bzw. 0,1Mantisse (implizites 0,1)

$$\mathtt{Bias(Exzess)} = 2^{n-1} \Rightarrow \mathtt{Charakteristik} = \mathtt{Exp} + 2^{n-1} \Leftrightarrow \mathtt{Exp} = \mathtt{Charakteristik} - 2^{n-1}$$

(ii) Variante 2: 1,M (implizite 1,)

$$\mathtt{Bias} = 2^{n-1} - 1 \Rightarrow \mathtt{Charakteristik} = \mathtt{Exp} + 2^{n-1} - 1 \Leftrightarrow \mathtt{Exp} = \mathtt{Charakteristik} - 2^{n-1} + 1$$

Vergleich: Mantisse gleich (i) + (ii), Charakteristik in Variante (i) um 2 höher als in (ii), weil Bias um eins höher (wird ja zum Exp. addiert um Ch. zu erhalten, und Komma um eins nach rechts gerückt). Aufgabe: Gebe die folgenden Zahlen in binärer Gleitkommadarstellung (VZ: 1B, Ch: 4B, M: 11B) und dezimaler Festkommadarstellung an.

(a) kleinste in binärer Gleitkommadarstellung darstellbare Zahl:

(i)
betragsmäßig: 0/1 0000 00000000000 = $\pm 0,100000000000_2 * 2^{0-2^3} = \frac{1}{2^9} \approx \pm 0,001953125$
absolut: 1 1111 11111111111 = $-,111111111111_2 * 2^{2^4-1-2^3} = -1111111,11111_2 = -(2^7 - 1 + 1/2 + 1/4 + 1/8 + 1/16 + 1/32) = -127,96875$

(ii)
betragsmäßig: 0/1 0000 00000000000 = $\pm 1,00000000000_2 * 2^{0-2^3+1} = \frac{1}{2^7} \approx \pm 0,0078125$
absolut: 1 1111 11111111111 = $-1,11111111111_2 * 2^{2^4-1-2^3+1} = -111111111,111_2 = -(2^9 - 1 + 1/2 + 1/4 + 1/8) = -511,875$

(b) (i)
größte darstellbare Zahl: $127,96875$

(ii)
größte darstellbare Zahl: $511,875$

(c) (i)
$\frac{11}{10} = 1,1_{10} = 1,0\overline{0011}_2 = 0,10\overline{0011}_2 * 2^1$
VZ = 0
Exp = 1 $\Rightarrow Charakteristik = 1+2^3 = 9_{10} = 1001_2$
M = 00011001100/1
Gleitkommazahl: 0 1001 00011001100/1

(ii)
$\frac{11}{10} = 1,1_{10} = 1,0\overline{0011}_2 \approx 1,00011001100_2$
VZ = 0
Exp = 0 $\Rightarrow Charakteristik = 0+2^3-1 = 7_{10} = 0111_2$
M = 0001100110/1
Gleitkommazahl: 0 0111 00011001100/1

(d) (i)
$17+2^{-3} = 17,125_{10} = 10001,001_2 \overset{norm.}{=} 0,10001001_2 * 2^5$
VZ = 0
Exp = 5 $\Rightarrow Charakteristik5+2^3 = 13_{10} = 1101_2$
Mantisse = 00010010000
Gleitkommazahl: 0 1101 00010010000

(ii)
$17+2^{-3} = 17,125_{10} = 10001,001_2 \overset{norm.}{=} 1,0001001_2 * 2^4$
VZ = 0
Exp = 4 $\Rightarrow Charakteristik = 4+2^3-1 = 11_{10} = 1011_2$
M = 00010010000
Gleitkommazahl: 0 1011 00010010000

(e) (i)
siehe Maximum bei (i) b)

(ii)
$7^7 = 823.543_{10} = 11001001000011110111_2 \overset{norm.}{=} 1.10010010000 * 2^{19}$
VZ = 0
Exp = 19 $\Rightarrow Charakteristik = 19+2^3-1 = 24 = 11000_2$ (nächste darstellbare Charakteristik: $1111_2 = 15_{10}$)
Gleitkommazahl = siehe Maximum bei (ii) b)

(f) (i)
$0,F_{16} = \frac{15}{16}_{10} = 0.9375_{10} = 0,1111_2(norm.)$
VZ = 0
Exp = 0 $\Rightarrow Charakteristik = 0+2^3 = 8_{10} = 1000_2$
Mantisse = 11
Gleitkommazahl = 0 1000 11100000000

(ii)
$0,F_{16} = \frac{15}{16}_{10} = 0.9375_{10} = 0,1111_2 \overset{norm.}{=} 1,111+2^{-1}$
VZ = 0
Exp = -1 $\Rightarrow Charakteristik = -1+2^3-1 = 6_{10} = 0110_2$
Mantisse = 111
Gleitkommazahl = 0 0110 11100000000

Problem 5: Gleitkommazahlen nach IEEE 754 (2+4+2=8)

Das IEEE-754-Format für Gleitkommazahlen definiert (unter anderem) zwei Darstellungen: Bei 32-Bit Gleitkommazahlen werden 1 Bit für das Vorzeichen, 8 Bits für den Exponenten und 23 Bits für die Mantisse verwendet; bei 64-Bit Gleitkommazahlen sind es 1 Bit für das Vorzeichen, 11 Bits für den Exponenten und 52 Bits für die Mantisse (jeweils in dieser Reihenfolge). Der Exponent wird dabei in der Excess-127 bzw. Excess-1023 Darstellung gespeichert, der Bias ist also 127 bzw. 1023. Betrachten wir die 32-Bit Gleitkommazahlen genauer: Ist die Bitfolge der Zahldarstellung durch $\mathsf{VEM} \in \mathsf{B}^{32}$ mit $\mathsf{V} \in \mathsf{B}, \mathsf{E} \in \mathsf{B}^8, \mathsf{M} \in \mathsf{B}^{23}$ gegeben, so ist der Zahlenwert durch

$$(\mathsf{VEM})_{\mathrm{IEEE}} = (-1)^{\mathsf{V}} \cdot 2^{(\mathsf{E})_2 - 127} \cdot (1,\mathsf{M})_2$$

gegeben. Das führende 1-Bit vor dem Komma in der normalisierten Darstellung ist also implizit und wird nicht gespeichert ('hidden bit'). Beispiel:

$$\begin{aligned}(1\ 01111111\ 10000000000000000000000)_{\mathrm{IEEE}} &= (-1)^1 \cdot 2^{127-127} \cdot (1,1)_2 \\ &= -1 \cdot 1 \cdot 1{,}5 \quad = \quad -1{,}5\end{aligned}$$

Dabei sind die Bitfolgen $\mathsf{E} = 0\ldots0$ und $\mathsf{E} = 1\ldots1$ jeweils für Spezialfälle reserviert ('not a number', unendlich, denormalisierte Darstellung, Null).

a) Stellen Sie die folgenden Zahlen im IEEE-32Bit-Format dar: $z_1 = 1{,}125$, $z_2 = 1{,}5$, $z_3 = 16777216{,}0$.

b) Berechnen Sie $(z_1 + z_2) + z_3$ und $z_1 + (z_2 + z_3)$ im IEEE-32Bit-Format, wobei nach jeder Operation gerundet wird. Runden Sie (i) zur nächsten darstellbaren Zahl; (ii) auf die betragsmäßig nächst kleinere darstellbare Zahl (Abschneiden nicht darstellbarer Stellen). Bei welcher der vier Rechnungen erhalten Sie das genaueste Ergebnis?

c) Wie viele Normalisierungsschritte sind bei der Berechnung von $16777218 - 16777216$ im IEEE-Format notwendig?

Solution 5

Problem 6: Primzahltest MMIX (8)

Schreiben Sie ein Programm, das den Benutzer dazu auffordert eine Zahl von 1 bis 10.000 einzugeben. Das Programm soll dann berechnen, ob es sich bei dieser Zahl um eine Primzahl handelt oder nicht und das Ergebnis dem Benutzer mitteilen.

Solution 6

Es gibt unterschiedliche Methoden zur Bestimmung, ob eine Zahl C Primzahl ist. Ein in einfacher Test, ob C durch eine Zahl kleiner als C teilbar ist, wäre eine Lösung. Besser ist jedoch einen Algorithmus wie "Das Sieb des Eratosthenes"zu implementieren:
Hier wird iterativ eine Liste von Primzahlen aufgebaut (initialisiert mit 2) wobei jede nachfolgende Zahl darauf hin geprüft wird, ob sie durch eine Zahl im "Sieb", der Liste, teilbar ist. Wenn nicht, ist wird sie der Liste am Ende als Primzahl hinzugefügt. Die hier implementierte Variante geht den umgekehrten Weg. In einem Feld der größe der maximalen Zahl (maxcandidate) + 1 werden Wahrheitswerte gespeichert, die Aussagen, ob die Zahl eine Primzahl ist oder nicht:

1. SIEB[0,1] = FALSE

2. SIEB[2..maxcandidate] = TRUE (alle Zahlen ab der 2 könnten Primzahl sein!)
3. solange i <= maxcandidate, finde das nächste i so dass Feld[i] = TRUE. Das ist eine Primzahl!
4. Setze alle Felder, die Vielfache von i sind FALSE.
5. weiter mit 3

```
              LOC     Data_Segment
              GREG    @
output        BYTE    "Eingabe 1 - 10.000:",#A,0
isprime       BYTE    "Ist Primzahl!",#A,0
noprime       BYTE    "Ist keine Primzahl!",#A,0

bufLng        IS      6 % 5 Ziffern und 0
buffer        BYTE    0
bufArg        OCTA    buffer,bufLng
maxprime      IS      10000
sieveArg      OCTA    sieve,maxprime+1
sieve         BYTE    0,0
              LOC     sieve+maxprime+1

              LOC     #100
              GREG    @

              PREFIX  :buildprimes:

args          IS      $10
maxprime      IS      $11
raddr         IS      $12
sieve         IS      $13
last          IS      $14
isprime       IS      $15
true          IS      $16
false         IS      $17
cand          IS      $18    % der Primzahlkandidat
multiple      IS      $19
c             IS      $20

begin         LDO     sieve,args
              LDO     last,args,8
              SUB     last,last,1
              SET     true,1
              SET     false,0
              // Zunächst kann jede Zahl >= 2
              // Primzahl sein, also initialisieren
              // wir das Array mit 1 (True).
              // Die ersten beiden Felder (0 und 1)
              // wurden beim anlegen des Arrays auf
              // 0 (False) gesetzt.

              SET     cand,2
allNumPrime   CMP     c,cand,last
```

```
                BP      c,remNonPrimes
                STB     true,sieve,cand
                ADD     cand,cand,1
                JMP     allNumPrime

                // Von der ersten Primzahl an (2)
                // setzen wir alle Nicht-Primzahlen
                // auf False
remNonPrimes    SET     cand,1
findNextPrime   ADD     cand,cand,1
                CMP     c,cand,last
                BP      c,end

                // Wir gehen zum nächsten Feld, in
                // dem eine 1 (True) steht...
                // (Das ist eine Primzahl!)
skipNonPrimes   CMP     c,cand,last
                BP      c,remMultOfPrime
                LDB     isprime,sieve,cand
                BNZ     isprime,remMultOfPrime
                ADD     cand,cand,1
                JMP     skipNonPrimes

                // ... und entfernen alle Vielfachen
                // des Feldindex, da das keine Prim-
                // zahlen sein können.
remMultOfPrime  ADD     multiple,cand,cand
findNextMult    CMP     c,multiple,last
                BP      c,findNextPrime
                STB     false,sieve,multiple
                ADD     multiple,multiple,cand
                JMP     findNextMult
end             GO      raddr,raddr,0

                PREFIX  :atoi:

raddr           IS      $50
result          IS      $51
string          IS      $52
strlen          IS      $53
digit           IS      $54
deci            IS      $55
c               IS      $56

begin           SET     result,0
                SET     deci,1
next            SUB     strlen,strlen,1
                LDB     digit,string,strlen
                CMP     c,digit,'0'
                BN      c,next
                CMP     c,digit,'9'
```

```
                BP      c,next
                SUB     digit,digit,'0'
                MUL     digit,digit,deci
                ADD     result,result,digit
                BZ      strlen,end
                MUL     deci,deci,10
                JMP     next
end             GO      raddr,raddr,0


                PREFIX  :

                // Hier wird das "Sieb des Eratosthenes" aufgebaut
Main            LDA     :buildprimes:args,sieveArg
                GO      :buildprimes:raddr,:buildprimes:begin

                // Hier findet die reine Ein-und Ausgabe statt
                LDA     $255,output
                TRAP    0,Fputs,StdOut

                LDA     $255,bufArg
                TRAP    0,Fgets,StdIn

                LDA     :atoi:string,buffer
                SET     :atoi:strlen,$255
                GO      :atoi:raddr,:atoi:begin

                // Der Primzahl-Test ist jetzt ein einfacher
                // Array-Zugriff...
                LDA     $0,sieve
                LDB     $1,$0,:atoi:result
                BZ      $1,no
                // ...und ein Vergleich für die Ausgabe!
                LDA     $255,isprime
                TRAP    0,Fputs,StdOut
                TRAP    0,Halt,0
no              LDA     $255,noprime
                TRAP    0,Fputs,StdOut
                TRAP    0,Halt,0
```