Lösung Aufgabenblatt 3-2

Darstellung von Zahlen

Aufgabe 1

Berechnen Sie folgende Aufgaben:

a) $101010_2 + 1001011_2$

b) $1011_2 \cdot 110_2$

c) $1001_2 - 110_2$

d) $101010 : 10$

Lösung

a)

$$1110101_2$$

b)

$$1000010_2$$

c)

$$11_2$$

d)

$$10101_2$$

Aufgabe 2

Konvertieren Sie die angebenen Zahlen ins

a) 255_{10} ins Binärsystem

b) 101010_2 in Dezimalsystem

c) 122_{10} ins Hexadezimalsystem

d) $a10gb_{16}$ ins Dezimalsystem

Lösung

a)

$$(11111111)_2$$

b)

$$(42)_{10}$$

c)

$$(7A)_{16}$$

d)

$$(2576)_{10}$$

Aufabe 3

Handelt es sich bei folgender, umgangssprachlich, prozeduralen Beschreibung um einen Algorithmus?

Überprüfen Sie dazu die Vorschrift auf die Eigenschaften

- Präszision
- Effektivität
- statische Finitheit
- dynamische Finitheit
- und Terminierung.

Begründen Sie kurz ihre Antwort.

Gegeben seien zwei positive Zahlen a, b. Setze k = 0. Solange k kleiner ist als b, führe folgende Schritte durch: Addiere b zu a hinzu (b bleibt unverändert). Falls a nun gerade ist, erhöhe k um 1. Ist k größer oder gleich b, gib a aus.

Lösung

Präzision ist erfüllt: Jeder Schritt hat eine klare, eindeutige Interpretation.

Effektivität ist erfüllt: Kein Schritt fordert etwas Unmögliches wie z.B. eine ganze Zahl zu finden die größer und kleiner 0 gleichzeitig ist.

Statische Finitheit ist erfüllt: Die prozedurale Beschreibung ist ein endlicher Text.

Terminierung ist nicht generell erfüllt: Ist `a` ungerade und `b` gerade, dann bleibt `a` stets ungerade, wodurch `k` nie erhöht wird und die Bedingung `k < b` gilt für immer. Das Wiederholen der Schritte bricht in diesen Fällen also nie ab.

Dynamische Finitheit ist nicht erfüllt: Für alle Eingaben `a` und `b` wird lediglich Speicher für die Werte von `a`, `b` und `k` benötigt. Da es sich bei den Werten aber um ganze Zahlen

handelt, die beliebig groß werden können, kann die dynamische Finitheit dennoch verletzt werden. Bei den Eingaben, die nicht terminieren, wird a mit jedem Schleifendurchlauf größer, dynamische Finitheit ist also nicht gegeben.

Die prozedurale Beschreibung ist also kein Algorithmus im Sinne der Vorlesung, da Terminierung und dynamische Finitheit nicht erfüllt sind.

Aufgabe 4

In einer Werbung wird eine SSD mit 1 Tbyte Speicherplatz beworben. Nachdem die SSD in den Computer eingebaut wurde, zeigt dieser, dass nur nur 931 Gbyte Speicher vorhanden sind. Beim Versuch des Umtauschs behauptet der Verkäufer, dies sei normal. Begründen Sie, warum der Verkäufer Recht hat.

Lösung

Grund ist die unterschiedliche Darstellung der Kapazitäten. Linuxbasierte Betriessysteme (Linux, MacOS) verwenden das Dezimalbyte-System, z.B.

1gByte = 1.000.000.000 Byte (Dezimalbytes)

Windows verwendet das Binärbyte-System, d.h. 1 kByte = 1024 Byte, 1 mByte = 1024 kByte 1GB = 1.000.000.000 Byte / 1.024 = 976.562.5 kByte /1.024 = 953,7 MByte

Aufgabe 5

1. Wahrheitstafel für $\neg(A \vee B)$:

A	B	$A \vee B$	$\neg(A \vee B)$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Ergebnis: Die Negation von $(A \vee B)$ ist nur dann wahr (1), wenn beide A und B falsch (0) sind.

---

2. Überprüfung von $(A \wedge B) \Rightarrow A$:

A	B	$A \wedge B$	$(A \wedge B) \Rightarrow A$	Erklärung
0	0	0	1	0 ? 0 = 1 (wahr)
0	1	0	1	0 ? 0 = 1 (wahr)
1	0	0	1	0 ? 1 = 1 (wahr)
1	1	1	1	1 ? 1 = 1 (wahr)

*Ergebnis: Die Aussage $(A \wedge B) \Rightarrow A$ ist eine **Tautologie** - sie ist immer wahr.*

---

3. Bitweise Multiplikation (AND-Verknüpfung):

Bit A	Bit B	$A \wedge B$	Erklärung
0	0	0	$0 \times 0 = 0$
0	1	0	$0 \times 1 = 0$
1	0	0	$1 \times 0 = 0$
1	1	1	$1 \times 1 = 1$

Beispiel: $1011 \wedge 1101 = 1001$

```
  1011
  1101
------
  1001
```

---

4. Bitweise Division - Komplexer Algorithmus:

Die bitweise Division kann nicht durch eine einfache Wahrheitstafel dargestellt werden. Sie erfordert einen mehrstufigen Algorithmus:

Operation	Verwendete Verknüpfungen	Zweck
Vergleich	XOR, AND	Prüfen ob Dividend ≥ Divisor
Subtraktion	XOR, AND, NOT	Rest berechnen
Verschiebung	Bit-Shift	Position anpassen

Vereinfachtes Beispiel für 1-Bit-Division: | Dividend | Divisor | Quotient | Rest | |–––-|–––|–––-|––| | 0 | 1 | 0 | 0 | | 1 | 1 | 1 | 0 |

---

5. Bitweise Subtraktion:

Wahrheitstafel für 1-Bit-Subtraktion ohne Borrow: | A | B | $A - B$ | Borrow | |–|–|–––|––| | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 0 |

Wahrheitstafel für 1-Bit-Subtraktion mit Borrow_in: | A | B | Borrow_in | Differenz | Borrow_out | |–|–|––––|––––|––––| | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | | 1 | 0 | 1 | | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 0 | | 1 | 1 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 | 1 |

Formeln: - Differenz = A ⊕ B ⊕ Borrow_in - Borrow_out = (¬A ∧ B) ∨ (¬A ∧ Borrow_in) ∨ (B ∧ Borrow_in)

Beispiel: $1001 - 0110 = 0011$

```
  1001
- 0110
------
  0011
```