Datenpräsentation

1. Bits

- Computer kennt zwei Zustände_ Strom an - Strom aus
- kleinste Inforamtionseinheit ist ein biary digit = Bit
- Mögliche Werte: 0 oder 1
- Logische Interpretation: falsch = 0, wahr = 1

Grundooperationen auf Bits

1. Logisches AND: $b_1 \wedge b_2$

b_1	b_2	$b_1 \wedge b_2$
0	0	0
1	0	0
0	1	0
1	1	1

2. Logisches OR: $b_1 \vee b_2$

b_1	b_2	$b_1 \vee b_2$
0	0	0
1	0	1
0	1	1
1	1	1

3. Logisches NOT (Negation, Komplement): $\neg b$

b	$\neg b$
0	1
1	0

Satz: Mit diesen drei Grundoperationen können alle möglichen Operationen auf Bits definiert werden.

Beispiel 1

b_1	b_2	$f(b_1, b_2)$
0	0	0
1	0	0
0	1	0
1	1	1

Auflösung: $f(b_1, b_2) = b_1 \land b_2$

Beispiel 2

b_1	b_2	$f(b_1, b_2)$
0	0	1
1	0	1
0	1	0
1	1	1

Auflösung: $f(b_1, b_2) = \neg b_1 \lor b_2$

2. Bytes

- Rechnen mit Bits ineffizient
- Computer fas Bits zu einem Byte (=8 Bits) zusmmen. Man spricht auch von Bitvektor.
- Es gibt aber auch 16, 32, 64 Bit Worte. Daher spricht man von Wortbreite
- vgl. 32-Bit- und 64-Bit-Architekturen
- Alle Daten werden im Computer als Bitvektoren dargestellt
- Die Interpretation des Bitvektors häng vom Datentyp ab. (int vs. float vs string)

Grundoperationen auf Worten

- Analog der Bitoperationen gibt es $w_1 \land w_2, w_1 \lor w_2, \neg w$
- Nur definiert auf Worte gleicher Breite
- Anwendung der Bitoperation auf entsprechenden Operationen

Beipsiele:

$1011 \wedge 1010 = (1 \wedge 1)(0 \wedge 0)(1 \wedge 1)(1 \wedge 0))$

$\neg 1010 = (\neg 1)(\neg 0)(\neg 1)(\neg 0) = 0101$

3. Zahlen - Binärsystem

- Mensch: Dezimalsystem = Zehnersystem = Basis 10.
- Ziffern: von 0 bis 9
- Jede Stelle einer Zahl entspricht einer 10er Potenz.
- beginnend von rechts mit 10^0

Beispiel:

$3455_{10} = 3 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 5 \cdot 10^0$

$= 3 \cdot 1000 + 4 \cdot 100 + 5 \cdot 10 + 5 \cdot 5$

$= 3455$

- Computer: Binärsystem = Basis 2
- Ziffern: 0, 1
- Eine Ziffer ist ein Bit.
- Jede Stelle entspricht einer 2er Potenz beginnen mit 2^0 von rechts

Beipsiel

$110011_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$= 32 + 16 + 0 + 0 + 2 + 1$

$= 51_{10}$

4. Zahlen - Hexadezimalsystem

- Stellewertsystem mit Basis 16 = 4 Bit pro Stelle
- Ziffern: 0, 1, ... 9, a, b, c, d, e, f
- Jede Stelle entspricht einer 16er Potenz beginnen mit 16^0 von rechts

Beipsiel

$beef_16 = 11 \cdot 16^3 + 14 \cdot 16^2 + 14 \cdot 16^1 + 15 \cdot 16^0$

$= 11 \cdot 4096 + 14 \cdot 256 + 14 \cdot 16 + 15$

$= 48879_{10}$

5. Algorithmus zum Umformen der Zahldarstellung

Defintion:

Ein Algorithums ist eine Vorschriftz zur Lösung einer Klasse von Problemen. Er besteht aus einer endlichen folge von Schritten, mit der aus bekannten Eingangsdaten neue Ausgangsdaten verechnet werden können.

Beispiele:

- Bedienungsanleitungen
- Aufbauanleitung bei Möbeln zum Beispiel
- Verahltensvorschriften bie Unfällen, Alarmen, usw.
- Rezepte
- Rechenvorschriften.
- usw.

Algorithmus zur Umrechnung von Dzeimalzahl in Binärzahl: Gegeben Dezimalzahl n und die Basis b =2 q ist der Quotient aus n//b und r der Divisonsrest

1. Bestimme q = n//b und r = n%b
2. Schreibe r links an die Ausgabe
3. Falls $q \neq 0$ gehe zu 1.
4. Sonst fertig.

Beispiel: B=2 und n = 42

- $42 : 2 = 21$ Rest **0**
- $21 : 2 = 10$ Rest **1**
- $10 : 2 = 5$ Rest **0**
- $5 : 2 = 2$ Rest **1**
- $2 : 2 = 1$ Rest **0**
- $1 : 2 = 0$ Rest **1**
- Fertig, weil $q = 0$
- Ergebnis 101010_2 von unten nach oben gelesen.

6. Grundrechenarten von Zahlen in Binärdarstellung

6.1 Addition

- Wortbreite 1:

+	0	1
0	0	1
1	1	0 mit Übertrag 1

- Schriftliche Addition:

$$\begin{array}{r} 111001 \\ +111011 \\ 110110 \\ \hline 1110100 \end{array}$$

Weitere Grundrechenarten auf dem Arbeitsblatt

7. Datentypen - Syntax und Semantik

1. Semantik eines Datentyps (Bedeutung)

Menge von Werten und den Operationen auf diesen Werten.

2. Syntax (Schreibweise, Darstellung)

Darstellung eines Wertes (=Literal) und die Operationssymbole zu den Operationen.

3. Pragmatik

Syntax und Semantik sollen den üblichen mathematischen Konventionen und Defintionen entsprechen.

Beispiele:

1. Integer

- Die Zahl sechszehn kann durch das Literal 16 dargestellt werden
- Die Zahl sechszehn kann auch durch das Literal 0x10 (hexadezimal) dargestellt werden.
- Die Zahl sechszehn kann auch durch das Literas 0b10000 (binär) dargestellt werden.

2. Float

- Die Zahl nullkommazwei kann durch das Literal 0.2 dargestellt werden.
- Die Zahl nullkommazwei kann durch das Literal 2.0e-1 dargestellt werden.

3. String

- Die Zeichenkette *Hund* kann als Literal "Hund" dargestellt werden.
- Die Zeichenkette *Hund* kann als Literal 'Hund' dargestellt werden.
- Die Zeichenkette *Hund* kann als Literal '''Hund''' dargestellt werden.

In Python Jeder Wert besteht aus zwei Teilen:

a) Typ

b) interen Repräsentaiton des Werts

Die interne Representation des Wertes ist immer eine Folge von Bits. (Bitvektor) Der Bitvektor wird anhand des Typs interpretiert:

0x10 im Datentyp **int** würde als **16** interpretiert werden.

0x10 im Datentyp **float** würde als **2.24E44** interpretiert werden.

0x40490fd0 im Datentyp **float** würde als **3.14159** interpretiert werden.

0x40490fd0 im Datentyp **int** würde als **1078530000** interpretiert werden.

0x68656c6c6f00 im Datentyp **string** würde als *"hello** interpretiert werden.