Lernziele 1. ZP Modul 114

Inhalt

Einführung

Codierung

Dadurch bestimmen wir, mit welchen Symbolen oder Zeichen Daten bzw. Werte dargestellt werden. Damit können diese von verschiedenen Sendern und Empfängern verwendet werden.

Beispiele aus dem Alltag: Schriftsprache, Messeeinheiten, Schulnoten.

Komprimierung

Mit Kompressionsverfahren können wir die Daten/Zeichenmenge reduzieren.

Beispiele aus dem Alltag: Ich kann "Menü 1" bestellen und muss nicht alle Gänge einzeln erwähnen. Kantone kann ich mit zwei Buchstaben angeben. Alle Chat-Abkürzungen: LOL, BRB, GLG...

Verschlüsselung

Mit Verschlüsselungsverfahren werden Daten/Übertragungen privat gehalten, d.h. nur Berechtigte können die Inhalte entschlüsseln. Verschlüsselung ist eine Codierung, wo die Vorgehensweise bewusst nicht allen bekanntgegeben wird.

Bespiele aus dem Alltag: "Jugendsprache" / "Verbrechersprache", Geheimdienstnachrichten, Finanzielle Transaktionen

Zahlensysteme

Positionsbasierte und nicht Positionsbasierte

Die Position links/rechts gibt an, das eine Zahl ein vielfaches mehr «zählt», je nach Stelle. Mit 847 zählt die 2. Stelle von links (8) 100 Mal (10x10) mehr als die Stelle ganz links (7). Die 1. Stelle von links (4) zählt 10 mal mehr.

Nicht positionsbasierte Zahlensysteme: Strichliste oder römischen Zahlen (Erweitert: warum verwenden diese so wenig: Versuche mal zwei römischen Zahlen zu addieren....)

binäres Zahlensystem

Positionsbasiertes System mit 2 Zahlen: 0 und 1. Jede Stelle nach links ist daher 2x mehr Wert als die vorherige (1, 2, 4, 8, 16...)

Die Informatik verwendet das binäre Zahlensystem, da fast alle Rechner mit Elektrizität funktionieren. Es ist viel einfacher zu messen, ob Strom fliesst oder nicht (0/10) als zum Beispiel 10 verschiedene Stromstufen messen zu müssen (10er System).

oktales Zahlensystem

Das oktale Zahlensystem, auch als Basis-8-System bekannt, verwendet 8 Symbole (0, 1, 2, 3, 4, 5, 6, 7) zur Darstellung von Zahlen. Im Gegensatz zum dezimalen System, das 10 Symbole (0-9) verwendet, hat das oktale System eine Basis von 8.

Um eine Zahl im oktalen System darzustellen, schreibst du sie mit den oben genannten Oktalziffern.

hexadezimales Zahlensystem

Positionsbasiertes System mit 16 Zahlen (0-F). Jede Stelle nach links ist daher 16x mehr Wert als die Vorherige (1, 16, 256, 4096...)

Mit eine zweistellige Hexadezimalzahl können 256 Werte dargestellt, was genau ein Byte entspricht. Man «verschwendet» also kein Platz wie wenn man 0-255 mit dem Dezimalsystem darstellt.

Zahlensysteme verglichen

Dezimal	Binär	Oktal	Hexadezimal	Eigenes
0	0	0	0	#
1	1	1	1	?
2	10	2	2	!
3	11	3	3	?#
4	100	4	4	??
5	101	5	5	?!
6	110	6	6	!#
7	111	7	7	!?
8	1000	10	8	!!
9	1001	11	9	?##
10	1010	12	A	?#?
11	1011	13	B	?#!
12	1100	14	C	??#
13	1101	15	D	???
14	1110	16	E	??!
15	1111	17	F	?!#
16	10000	20	10	?!?
17	10001	21	11	?!!
18	10010	22	12	!##
19	10011	23	13	!#?

Verwendung in der IT

Binäre Kodierung

Binäre Kodierung von negativen Zahlen (Einerkomplement/Zweierkomplement/Offset Binary)

Einerkomplement: Wie funktioniert die Negierung einer Zahl?

<u>Aufgaben</u> negative Zahlen im 1er Komplement

Rechnen Sie die Zahlen links (1er Komplement) in Dezimalzahlen um!

00000000_2 ≙ 0_{10}

11111111_2 ≙ 0_{10}

00000001_2 ≙ 254_{10}

11111110_2 ≙ 1_{10}

Zweierkomplement: Wie funktioniert die Negierung einer Zahl?

<u>Aufgabe</u> zu negative Zahlen im 2er Komplement

Lösen Sie folgende Aufgaben mit 8 Bit Dualzahlen im 2-Komplement:

<u>Aufgabe</u>: Nennen Sie die grösste und kleinste Zahl (Dezimal) welche mit dem 2-Komplement mit 8 Bits dargestellt werden kann:

$+32_{10}$ $+15_{10}$ **0010 0000 (+32)** **0000 1111 (+15)** ***0000 0000 (Übertrag)*** **0010 1111 = 47_{10}**	$+32_{10}$ -15_{10} **0010 0000 (+32)** **1111 0001 (-15 im Zweierkomplement)** ***1100 0000 (Übertrag)*** **0001 0001= 17_{10}** Führender Übertrag wird ersatzlos abgeschnitten
-32_{10} $+15_{10}$ **1110 0000 (-32 im Zweierkomplement)** **0000 1111 (+15)** ***0000 0000 (Übertrag)*** **1110 1111 = -17_{10}** **Da führende 1 ist es eine negative Zahl => Zweierkomplement bilden** **0001 0000 (invertiert)** **0000 0001 (Plus 1)** **0001 0001 = 17_{10} aber Minus!**	-32_{10} -15_{10} **1110 0000 (-32 im Zweierkomplement)** **1111 0001 (-15 im Zweierkomplement)** ***1100 0000 (Übertrag)*** **1101 0001 = -47_{10}** **Da führende 1 ist es eine negative Zahl => Zweierkomplement bilden** **0010 1110 (Invertiert)** **0000 0001 (Plus 1)** **0010 1111 = 47_{10} aber Minus!**

Kleinste Zahl: 127_{10} $0111\ 1111_2$

Grösste Zahl: 128_{10} $1000\ 0000_2$

Was passiert, wenn Sie die Zahl 0_{10} mittels 8 Bit ins 2-Komplement umrechnen?

Es ergibt wieder 0 → Es gibt nur eine Darstellung für 0 im 2-Komplement

Positiv: 1. Stelle = 1; Negativ: 1. Stelle = 0

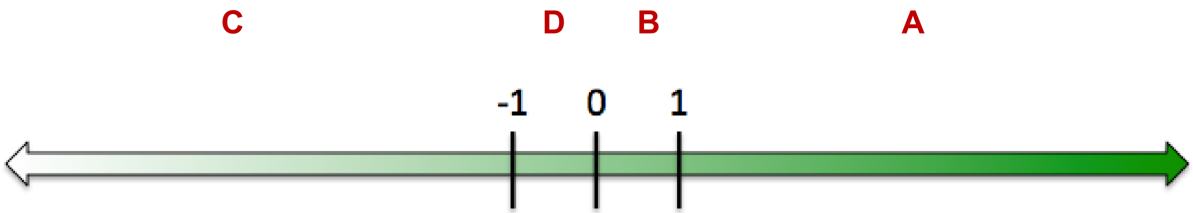
Die entspricht dem «Sprung» 127→ 128 (vorderstes Bit) in der normalen binären Darstellung

Binäre Kodierung von Fliesskommazahlen

Fliesskommazahlen (engl. floating point numbers)

Ordnen Sie die Vorzeichenvarianten der entsprechenden Stelle auf dem Zahlenstrahl zu!

Vorzeichen Mantisse	+	+	−	−
Vorzeichen Exponent	+	−	+	−
Zuordnen:	A	B	C	D



Stellen Sie mit Hilfe der Exceldatei die folgenden Zahlen dar. Notieren Sie dazu die Bits des Exponents (gelbe Zellen) und der Mantisse (blaue Zellen).

Das "hidden Bit" wird dabei nicht notiert. Alle Bits in einer Reihe werden weiter zu einer Hexadezimalen Zahl umgerechnet, um diese einfacher darzustellen.

Beispiel:

376917.25 ≙ 010010001 01110000000101010101000
≙ 48B80AA8

5.75 ≙ **0100 0000 1011 1000 0000 0000 0000 0000**
≙ **40B80000**

5.75 ≙ **1100 0000 1011 1000 0000 0000 0000 0000**
≙ **C0B80000**

0.0089 ≙ **0011 1100 0001 0001 1101 0001 0100 1110**
≙ **3C11D14E**

Welche Fliesskommazahl wird aus der folgenden Hexadezimalzahl gewonnen?

493C8C74 ≙ **1.47303628921508**