**Spicker M114 Winkler Olivier 26.03.18**

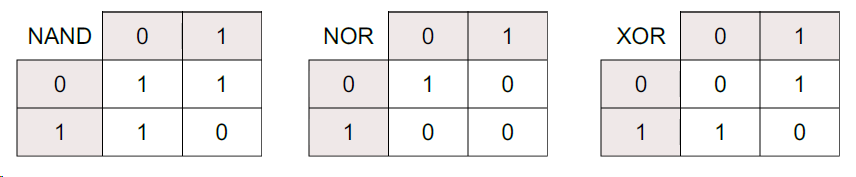
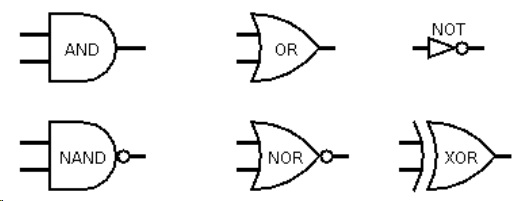
10er System = Dezimal (2’000)

2er System = Binär (20 🡪 0001 0100)

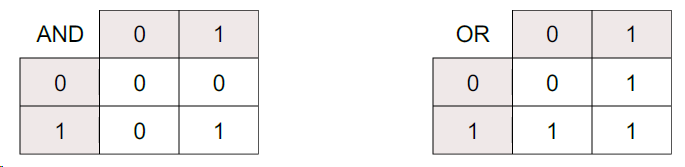
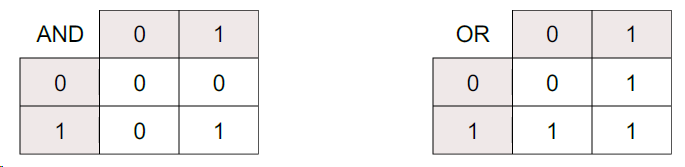
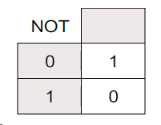
16er System = Hexa (10 zu 1010 = A) 0,1,2,3,4,5,6,7,8,9, A = 10, B = 11, C = 12, D = 13, E = 14, F = 15 (Immer in 4 Stellen aufteilen = 0000 0000 0000)

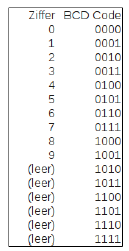
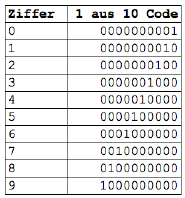
Beispiel: 1110 1011 = 14 11 🡪 EB

8er System = Octal (Immer in 3 Stellen aufteilen = 000 000 000)

Beispiel: 11101011 = 011 101 011 = 3 5 3 = 353

Logische Rechenoperationen



Pseudo-Tetrade = 6 redudante Zeichen im BCD Code, Ziffern werden einzeln ausgerechnet. Falls bei Addition Resultat in Pseudotetradebereich gelangt, 0110 dazurechnen.

Hamming Abstand zählt wie viele Bits sich

Von der einen zur nächsten Zahl ändern.

1010 0100

1110 0101 = Hamming Abstand = 2

Fehlererkennung (Paritätsbit)

Paritätsbit zählt wie viele 1 vorkommen.

Even parity = Gerade Anzahl 1 = 0 anhängen.

Odd Parity = Ungerade Anzahl 1 =1

**Hammingcode**

1001000 wird eingefüllt ohne 2er Potenzen. Danach werden bei allen Zahlen, die den Wert 1 haben (Nr. 11 und 7) diese Binär zusammengerechnet. Das Ergebnis wird in die 2er Potenzen eingefüllt. **Es werden keine Überträge gezählt.**

Ergebnis der beiden Zahlen = 1100

Bei der De-Codierung werden alle Zahlen mit dem Wert 1 (In Tabelle Nummern!) zusammengezählt. Falls das Endergebnis 0 ergibt, wurde die Nachricht richtig übertragen. Falls das Endergebnis nicht 0 ist, wurde die Nachricht mit Fehlern übertragen. Das Endergebnis wird Binär ausgerechnet und dann das Fehlerhafte Bit durch ein 0 & 1 ersetzt. Danach ist die Nachricht Fehlerfrei.

Zuerst = 1011 1001 000 Dann zu = 1001 1001 000

UTF-8 Codierung:

Unicode UTF-8 Codierung

0000 0011 0xxx xxxx (bei x die Unicode-Zeichen Hex in Binär ausgerechnet einfügen)

0000 0111 110x xxxx 10xx xxxx (Gleich wie oben)

0000 1111 1110 xxxx 10xx xxxx 10 xxxx (Gleich wie oben)

UTF-16 Codierung:

Unicode UTF-16 Codierung

0000 0000 – 0000 FFFF xxxx xxxx xxxx xxxx 🡪 Unicode Ebene 0

0001 0000 – 0010 FFFF 1110 xxxx 10xx xxxx 🡪 Unicode Ebene 1

UTF-32 Codierung:

Unicode UTF-32 Codierung

0000 0011 0000 0000 0000 0000 0000 0000 0000 0011 (Hex) (Unicode wird für die Codierung übernommen, muss aber von Hex in Binär umgerechnet werden und immer auf 32 Bits aufgefüllt werden mit 0.)

Float = 32 Bits = 4 Bytes Double = 64 Bits = 8 Bytes

Bit 32 = Vorzeichen Bit 64 = Vorzeichen (1 = Negativer Zahlenbereich 0 = Positiver Zahlenbereich)

Bit 31 – 23 = Exponent Bit 63 – 52 = Exponent bei Double 11 Gross

Bit 22 – 14 = Mantisse Bit = Mantisse bei Double 52 Gross

**3,703125**

1. Zahl vor Komma anschauen und in Binär umrechnen (3 = 0011)

0011,101101

0,703125 \* 2 = 1,40625 (1)

0,40625 \* 2 = 0,8125 (0)

0,8125 \* 2 = 1,625 (1)

0,625 \*2 = 1,25 (1)

0,25 \* 2 = 0,5 (0)

0,5 \*2 = 1,0 = 0 Gleich kein Rest also ENDE (1) = Binäre Zahl = 11,1011,01

2. Von Binär in Wissenschaftliche Form. (Ziel vor Komma eine 1)

1,1011,01 Eine Stelle nach Rechts schieben damit 1 vor Komma ist.

1,1011,01 \* 2 Hoch Exponent (1) Komma nach Rechts schieben = positiver Exponent. Komma nach Links = negativer Exponent.

3. IEEE752

127 + 2 Exponent = 128 und das in Binär 1000 0000.

**Excessdarstellung** (Endresultat = 0 1000 0000 1011 0100 0000 0000 0000 000)

Vorzeichen (Ist Kommazahl vom Anfang pos oder neg (**3,703125**))

0

Exponent (Von Schritt 3 ausgerechnet in Binär)

1000 0000

Mantisse (Alle Zeichen nach Komma. Restliche Stellen mit 0 auf 32 Bits auffüllen)

1011 0100 0000 0000 0000 000

**Zurückrechnen**

Zeichen von Vor und Exponent

11 = kleiner als -1

10 = kleiner als 0 aber grösser als -1

01 = grösser als 1

00 = grösser als 0 aber kleiner als 1

x (Dezimalzahl) = (-1)hoch 0 \* m \*