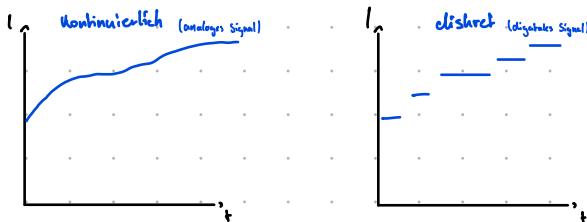



1. Einführung



Signal: Verlauf einer physikalischen Größe über der Zeit

digitales Signal: nimmt nur endlich viele verschiedene Werte an

analoges Signal: nimmt überabzählbar unendlich viele Werte an

Die Welt läuft kontinuierlich ab

binäres Signal: spezielles digitales Signal, das nur zwei verschiedene Werte annimmt

↳ Conventional Computing (Smartphone, Tablet, PC, ...)

↔ Unconventional Computing
(Analogue Computer, Quantum Computer)

Physikalische Realisierung eines binären Systems:

bsp: Strom fließt ↔ Strom fließt nicht

abstrakt

0 ↔ 1

reale Welt

Modellierung

Abstraktion

"digitale" Welt

Interpretation
Bewertung

2. Zahlensysteme

Desimalsystem

- Das Desimalsystem ist ein Stellenwertsystem mit der Basis 10 und den Ziffern 0, 1, ..., 9.

bsp: $4 \cdot 10^3 + 3 \cdot 10^2 + 0 \cdot 10^1 + 3 \cdot 10^0 = 4 \cdot 1000 + 3 \cdot 100 + 0 \cdot 10 + 3 \cdot 1$

Stellenwertigkeit
Potenz zur Basis 10

Hexadezimalsystem

um lange Bitfolgen schnell kürzer darstellen zu können

Stellenwertsystem mit der Basis 16 und den 16 Ziffern 0, 1, 2, ..., 9, A, B, ..., F

bsp: $4 \cdot 16^4 + 1 \cdot 16^3 + 14 \cdot 16^2 + 4 \cdot 16^1 + 4 \cdot 16^0 = 4 \cdot 256 + 1 \cdot 16 + 14 \cdot 4 = 1054$

Binärsystem

- Das Binärsystem ist ein Stellenwertsystem mit der Basis 2 und den Ziffern 0, 1.

bsp: $1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 101010_2 = 168_{10}$

Stellenwertigkeit
Potenz zur Basis 2

Umwandlung einer Desimalzahl in eine Hexadezimalzahl

mit dem Divisionsrestverfahren

z.B. $1054_{10} : 16 = 65 \text{ R } 14$

$65 : 16 = 4 \text{ R } 1$

$4 : 16 = 0 \text{ R } 4$

$1054_{10} = 41E_{16}$

Umrechnung einer Desimalzahl in ein Dualzahl mit dem Divisionsrestverfahren:

$$204 : 2 = 102 \text{ Rest } 0$$

$$102 : 2 = 51 \text{ Rest } 0$$

$$50 : 2 = 25 \text{ Rest } 0$$

$$25 : 2 = 12 \text{ Rest } 1$$

$$12 : 2 = 6 \text{ Rest } 0$$

$$6 : 2 = 3 \text{ Rest } 0$$

$$3 : 2 = 1 \text{ Rest } 1$$

$$1 : 2 = 0 \text{ Rest } 1$$

$$204_{10} = 11001001_2$$

↓ von unten nach oben lesen von links nach rechts aufschreiben

Alternativ: Desimal → Binär → 4er Blöcke → pro 4er Block 4 Zeichen
↔ Hex

Anmerkung: $32 \text{ Bit} = 4 \text{ Byte} = \text{IPv4-Adresse}$

z.B.: 1100.0000.1010.0000, 0001.1010.0010, 1001,

192. 168. 26. 44

gepunktete Dezimalnotation einer IPv4-Adresse

Dotted Decimal Notation DDN

2er-Potenzen:							
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
1	2	4	8	16	32	64	128
2048	1024	512	256	128	64	32	16
8	4	2	1				
$2^{12} = 4096$	$2^{13} = 1\ 048\ 576$						
$2^{14} = 65536$	$2^{15} = 16\ 777\ 216$						

16er-Potenzen:					
16^0	16^1	16^2	16^3	16^4	16^5
1	16	256	4096	65536	1
$16^5 = 1\ 048\ 576$					
$16^6 = 16\ 777\ 216 = (2^4)^6 = 2^{16}$					

Alle möglichen 4-Bit-Folgen:

0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Es gibt 2^n verschiedene Bitfolgen der Länge n.

Einheiten

Bit: binary digit, kleinste informationstragende Einheit,

kann die Werte 0 oder 1 annehmen

1 Byte: 8 Bit, kleinste adressierbare Speicherinheit

K 1 Kilo = $1000 = 10^3$	$\Rightarrow 1024 = 2^10 = 1\text{Ki}$
M 1 Mega = $1000\text{ Kilo} = 1\ 000\ 000 = 10^6$	$\Rightarrow 2^10 = 1\text{Mi}$ Negbinary
G 1 Giga = $1000\text{ Mega} = 10^9$	$2^{10} = 1\text{Gi}$
T 1 Tera = $1000\text{ Giga} = 10^{12}$	$2^{10} = 1\text{Ti}$
P 1 Petas = $1000\text{ Tera} = 10^{15}$	$2^{10} = 1\text{Pi}$
E 1 Exa = $1000\text{ Petas} = 10^{18}$	$2^{10} = 1\text{Ei}$

n	#Bitfolgen	Zahlenbereich	Java-Datentyp
8	$2^8 = 256$	-128 bis 127	byte
16	$2^{16} = 65536$	-32768 bis 32767	short
22	2^{22}	- 2^{34} bis $2^{34}-1$	int
64	2^{64}	- 2^{63} bis $2^{63}-1$	long

Negative Zahlen

Addition \longrightarrow Addierwerk

Subtraktion \longrightarrow Subtraktor

Komplement

Subtraktionsmodul im Prozessor Kern nicht vorhanden

Der Betrag einer Zahl x geschrieben ist, ist die zu x zugehörige positive Zahl:

$$\text{Bsp: } |-4710| = 4710$$

$$1451 = 15$$

Der Betrag einer negativen Zahl erhält man durch

2er-Komplementbildung:

Bsp: 1. $1111\ 1110 = \underline{\underline{2}} - 2$ \rightarrow führende 1 \Rightarrow Betrag bilden

$$\begin{array}{r} 0000\ 0001 \\ + 1111\ 1110 \\ \hline 1111\ 1111 \end{array}$$

2. // A und 0 umdrehen \Rightarrow 1er-Komplement
3. // +1 \Rightarrow 2er-Komplement \Rightarrow Betrag

Binärzahl mit führender null 0 ist positiv

Binärzahl mit führender eins 1 ist negativ

$2^n = 16$ Bitfolge

Zahlbereich mit 4 Bits: -8 bis 7

Mit n Bit lassen sich 2^n Bitfolgen bilden

Zahlbereich mit n Bit: -2^{n-1} bis $2^{n-1}-1$

Anmerkung: Darstellung negativer

Zahl aus gegebener positiver:

$$\begin{array}{r} \text{darstellung: } -57 \\ 0011\ 1011 = 57 \\ 1100\ 0100 = 160\ 6 \\ + 0000\ 0001 \\ \hline 1100\ 0101 = 261 = -57 \end{array}$$

erst positive aufschreiben,
dann 2er-Komp.



4+5

$$\begin{array}{r} 0100\ 1 \\ + 0101\ 1 \\ \hline 1000\ 1 = ? \end{array}$$

Fehlüberlauf
Overflow

$\xrightarrow{x > 1000\ 1}$ $1000\ 1 = 16\ 1$
 $0101 = 2\ 1$
 $0101 = 2\ 0\ 1$

Darstellung gebrochener Zahlen

Bsp. S.11

$$205,375 \quad | \quad |$$

$$10^5 \quad 10^4 \quad 10^3 \quad 10^2 \quad 10^1 \quad 10^0 \quad 10^{-1} \quad 10^{-2}$$

Stellenwerte

$$\underline{= 2 \cdot 10^4 + 0 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 + 5 \cdot 10^{-3}}$$

ganztägiger Anteil gebrochener Anteil

Umwandlung einer gebrochenen Dezimalzahl in eine Dualzahl: 205,375

- (1) ganztägiger Anteil mit Divisionssatzverfahren:

$$205 : 2 = 102 \quad R_1$$

$$102 : 2 = 51 \quad R_2$$

$$51 : 2 = 25 \quad R_3$$

$$25 : 2 = 12 \quad R_4$$

$$12 : 2 = 6 \quad R_5$$

$$6 : 2 = 3 \quad R_6$$

$$3 : 2 = 1 \quad R_7$$

$$1 : 2 = 0 \quad R_8$$

$$205:1100 \quad \rightarrow \quad \uparrow$$

- (2) gebrochener Anteil mit Multiplikationsverfahren:

$$0,375 \cdot 2 = 0,75 + 0 \quad \downarrow$$

$$0,75 \cdot 2 = 0,5 + 1 \quad \downarrow$$

$$0,5 \cdot 2 = 0 + 1 \quad \downarrow$$

$$0,375 = 0,011$$

$$\uparrow \uparrow \uparrow \uparrow \quad \text{Stellenwerte}$$

$$= 0 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3$$

$$= 0 \cdot 4 + 0 \cdot 2 + 0,25 + 0,125$$

$$= 0,375$$

- (III) zusammen:

$$205,375 = 1100 \ 1001,0011$$

- (IV) Normalisierung

$$\text{Binär: } 28,421875 = +11100,011000$$

$$\begin{array}{c} \downarrow \quad \text{Normalisierung} \\ +1,110000000 \cdot 2^4 \end{array}$$

decimal:

$$\begin{array}{r} -15,308 \\ \downarrow \\ -1,5308 \cdot 10^2 \end{array} \quad \begin{array}{r} 0,000110 \\ \downarrow \\ 8,34 \cdot 10^{-5} \end{array}$$

V Vorzeichen
M Mantisse; R Restmantisse
E Exponent

mit Single Precision Darstellung (in Java float)

ist der Bias $B = 127$

damit Charakteristik $C = E + B = 4 + 127 = 131$

$$= 1000 \ 0011_2$$

Binär:

$$\begin{array}{ccccccccccccc} 0 & 100 & 0000 & 1100 & 0000 & 0000 & 0000 & 0000 & 0000 \\ \downarrow & C & \quad \\ 0 & 1 & 100 & 1100 & 0000 & 0000 & 0000 & 0000 & 0000 \\ & R & \quad \end{array} \quad \rightarrow R wird mit 0 aufgefüllt, sodass 4 Bytes (bei float)$$

hexa:
 $41E36000$

R: alles nach Komma (noch vorsteher 1)

E: Exponent, um wie viel das Komma verschoben wurde

C: Exponent + Bias

Zur Rückberechnung: eine 1 vor Restmantisse, Komma um Exponent verschieben $E = C - B$

Bsp. S.12

Binär \rightarrow Decimal ($0.11111111 \dots$) , Vorzeichen

größte darstellbare Zahl mit SP:

$$\begin{array}{ccccccccccccc} 0 & 111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 \\ \downarrow & C & \quad & \quad & \quad & \quad & \quad & \quad \\ 0 & 1 & 111 & 1111 & 1111 & 1111 & 1111 & 1111 \\ & R & \quad & \quad & \quad & \quad & \quad & \quad \end{array}$$

$$E = C - B = 254 - 127 = 127$$

$$(-1)^C \cdot \underbrace{1,111\dots1}_\infty \cdot 2^{E-B}$$

$$= 2 \cdot 2^{127} = 2^{128} = 3,4 \cdot 10^{38} = 340.282.366.900.000.000.000.000.000.000.000.000.000$$

Zahlenbereich von Java-Datentyp float: $-3,4 \cdot 10^{38}$ bis $3,4 \cdot 10^{38}$

größte darstellbare Zahl mit DP:

$$\begin{array}{ccccccccccccc} 0 & 111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 \\ \downarrow & C & \quad \\ 0 & 1 & 111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 \\ & R & \quad \end{array}$$

$$E = C - B = 2046 - 1023 = 1023$$

$$(-1)^C \cdot 2^E$$

$$= 2^{1023} = 1,8 \cdot 10^{308}$$

Zahlenbereich von Java-Datentyp double: $-1,8 \cdot 10^{308}$ bis $1,8 \cdot 10^{308}$

SP und DP

SP: 32 Bit \rightarrow 4 Bytes

U	C	R	B: 127
1Bit	8Bit	23Bit	$E: -126 \leq E \leq 127$
cccc cccc cRRR RRRR RRRR RRRR RRRR RRRR			

DP: 64 Bit \rightarrow 8 Bytes

U	C	R	B: 1023
1Bit	11Bit	51Bit	$E: -1022 \leq E \leq 1023$
cccc cccc cccc RRRR RRRR			

Sonderfälle

$$0,0: \text{Test: } \underbrace{\text{0000} \dots \text{0000}}_C \underbrace{\text{0000} \dots \text{0000}}_E \underbrace{\text{0000} \dots \text{0000}}_B = E-C = 0 - 127 = -127 \quad z = (-1)^B \cdot 1, \underbrace{000 \dots 0}_2 \cdot 2^{-127}$$

$$= \underbrace{0,00 \dots 01}_2 = 1 \cdot 2^{-127} \neq 0,0$$

$\Rightarrow 0 : \text{Spezialfall}$

$C=0 \quad R=0$

Wert von C (z)	Wert von R	Bedeutung	Anmerkung
$C=0$ $(E=-127 \text{ / } -1023)$	$R=0$	Null	+0 und -0 unterscheidbar
$C=0$	$R>0$	denormalisierte Zahl	$(-1)^B \cdot (0,R) \cdot 2^{(E-B)}$
$C=255 / 2047$ $E=128 / 1024$	$R=0$	Unendlich	+ ∞ und - ∞ unterscheidbar
$C=255 / 2047$ $E=128 / 1024$	$R>0$	Keine Zahl (NaN)	ungefähige Operationen

Zu denormalisierte Zahlen:

$$\text{0000} \dots \text{0000} \text{ 0100} \text{ 0000} \dots \quad \text{Denormalisierte Zahlen dienen nur}$$

→ Spezialfall & Tabelle

Darstellung sehr nahe bei Null liegender Zahlen

$$(-1)^B \cdot 0, \underbrace{100 \dots 0}_2 \cdot 2^{-127}$$

$$= 0,1 \cdot 2^{-126} = \underbrace{0,00 \dots 01}_2 = 2^{-127}$$

126 Nullen

ASCII American Standard Code for Information Interchange

Zeichensatz: Vorrat an Zeichen

Zeichencode: Vorrat an Zeichen mit einer Ordnung

Anzahl Zeichen: $128 = 2^7$ Zeichen $\rightarrow 7$ Bit pro Zeichen (3 Bits Bit Paritätstest) \hookrightarrow Erkennung von Fehlern

Zeichenarten: Steuerzeichen, Sonderzeichen, Zahlen, Buchstaben (kein ä, ö, ü, ß)

UTF-8

Codepunkt: von Unicode-Konsortium für jedes Zeichen vergebene Nummer
 → Zeichenwert eines Zeichens
 z.B. U+007C

UTF-8: 8/16/24/32 Bits pro Zeichen (1-4 Bytes)

1 Byte	0xxxxxxx	$2^7 = 128$ Zeichen
2 Byte	110xx 10xx xxxx	$2^{11} = 2048$ Zeichen
3 Byte	1110x 10xx 10xx xxxx	$2^{16} = 65536$ Zeichen
4 Byte	11110 10xx 10xx 10xx xxxx	$2^{21} = 2.097.152$ Zeichen

UTF: Unicode Transportation Format

→ konkrete Speicher- und Übertragungsrate

UTF-16: 16 oder 32 Bit pro Zeichen

UCS: Universal Coded Character Set

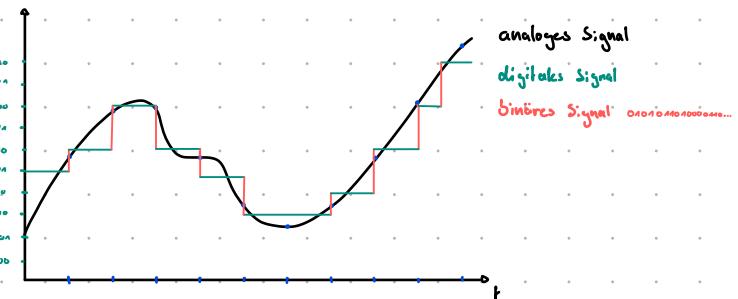
(ISO 10646)

UTF-32: 32 Bit pro Zeichen

UCS-4 \equiv UTF-32

UCS-2 \equiv UTF-16

Analog/Digital-Wandlung



1. Abtastung (Sampling)

Zerlegen der Zeitachse in Intervalle,

Messen der Signalwerte an den Intervallgrenzen

Samplingrate = Anzahl Abtastungen pro Zeiteinheit

$$\text{z.B. } 44.100 \cdot \frac{1}{2} = 44.1 \text{ kHz}$$

2. Quantierung / Quantisierung

Zerlegen der Wertekurve in endlich viele Intervalle

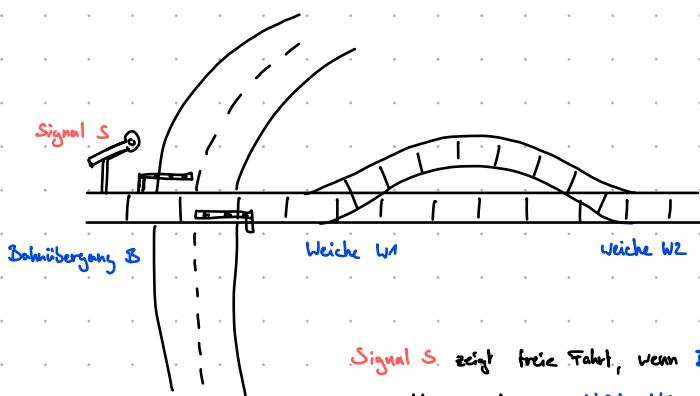
Zuordnung der gemessenen Signalwerte zu den Wertekintervallen

Signalauflösung = Anzahl Bits die notwendig sind, um alle endliche Anzahl Wertekintervalle unterscheiden zu können.

$$\text{Hier: 4 Bit d.h. } 2^4 = 16 \text{ Wertekintervalle}$$

$$\text{oder z.B.: 16 Bit, d.h. } 2^{16} = 65536 \text{ Intervalle}$$

3. Datenverarbeitung



Signal S zeigt freie Fahrt, wenn Bahnübergang B geschlossen ist und Weiche W_1 und Weiche W_2 geradeaus zeigen oder \neg und \neg beide aufs Überholgleis gestellt sind

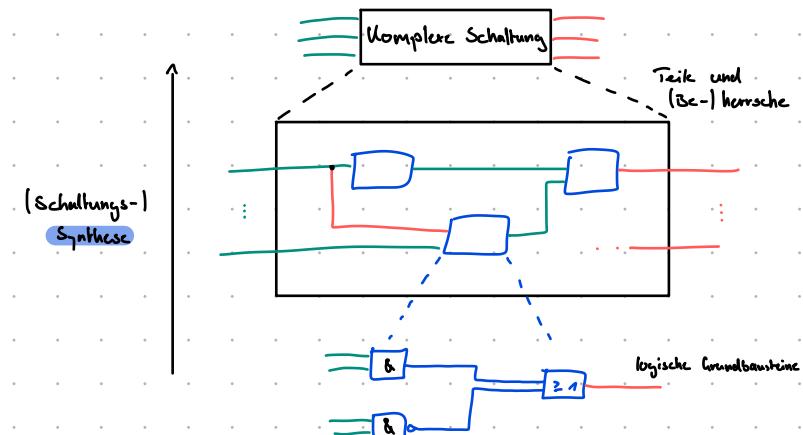
gegebenes Problem

Ausgangsvariable S freie Fahrt
geschlossen 0
 1

Eingangsvariable B offen
geschlossen 0
 1

W_1 } geradeaus
 W_2 } Überholgleis 0
 1

Modellierung



Digitaltechnik

beschäftigt sich mit der Verarbeitung und Speicherung von endlich vielen abstrakten (= kontinuierlichen) Signalwerten

(Digitale Signale) in digitalen Schaltungen

Zwei Signalwerte (als 0 und 1 bezeichnet)

Bauelemente: Logikgatter, Mikroprozessoren, Datenspeicher



Logikgatter:

Name Zeichen Schreibweise Wahrheitstabelle

1. NOT



$$y = \bar{x}$$

x	y
0	1
1	0

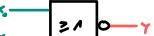
2. OR



$$y = (x_1 \vee x_2)$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

3. NOR



$$y = (\bar{x}_1 \vee \bar{x}_2)$$

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

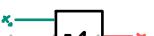
4. AND



$$y = (x_1 \wedge x_2)$$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

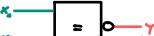
5. XNOR



$$y = ((\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2))$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

6. XOR



$$y = ((\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2))$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Aquivalenz

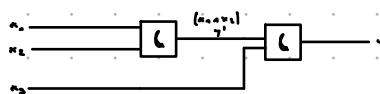
$$\text{XOR} \equiv \text{XNOR} \oplus 1$$

$$y = ((\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2))$$

Assoziationsgesetze siehe S.24

Konjunktion

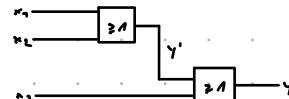
$$(x_1 \wedge x_2) \wedge x_3 = (x_1 \wedge (x_2 \wedge x_3))$$



daraus:

Disjunktion

$$(x_1 \vee x_2) \vee x_3 = (x_1 \vee (x_2 \vee x_3))$$



daraus:

Distributivgesetze siehe S.25

$$1. ((x_1 \wedge x_2) \vee (x_1 \wedge x_3)) = (x_1 \wedge (x_2 \vee x_3))$$

$$2. ((x_1 \vee x_2) \wedge (x_2 \vee x_3)) = (x_2 \wedge (x_1 \vee x_3))$$

Schaltungserstellung (-Vorgehen)

1. Modellieren \rightarrow S.24

2. Wahrheitstabelle

3. DNF \longrightarrow Disjunktive Normalform: Disjunktion (OR-Verknüpfung) von Mintermen

4. Schaltung

Konjunktion (AND-Verknüpfung) aller negierten oder nicht-negierten Eingangsvariablen

die Variablen einer Spalte AND-verknüpft wo das Ergebnis "1" ist

Satz der doppelten Negation

$$\gamma = \bar{x} = x$$

nur gleichlange Striche Wörter

DeMorgan'sche Gesetze

$$1. (\bar{x}_1 \wedge \bar{x}_2) = (\bar{x}_1 \vee \bar{x}_2) \quad 2. (\bar{x}_1 \wedge x_2) = (\bar{x}_1 \vee \bar{x}_2)$$

Paritätsbit

→ Erkennung von Fehlern

1010 MOD

↓ durch bitweise KDR-Verknüpfung der 7 Bits

$$\begin{array}{r} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{array} \xrightarrow{\text{KDR-Verknüpfung}} \begin{array}{r} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{array}$$

$$\begin{array}{l} 0 = 0 = 0 \\ 1 = 0 = 1 \\ 0 = 1 = 0 \\ 0 = 0 = 0 \\ 1 = 1 = 1 \\ 0 = 0 = 0 \\ 1 = 0 = 1 \end{array}$$

Sender: 0001 0100

Übertragung S

Empfänger 0001 0100 ⇒ bitweise KDR-Verknüpfung
0, höchstwahrscheinlich fehlerfrei Übertragung
1, Fehler bei der Übertragung

Stichwort
RATID

Addierer

$$\begin{array}{r} 0101 \times 1 \\ + 0001 \times 2 \\ \hline 0110 \quad y \\ y_4 \quad y_3 \quad y_2 \end{array}$$

Halbaddierer HA

2 Eingänge x_1, x_2

2 Ausgänge y_0, co (Übertrag, carry out)

Volladdierer VA

3 Eingänge x_1, x_2, ci (carry in)

2 Ausgänge y, co

WT:	x_1	x_2	y_0	co
0	0	0	0	0
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

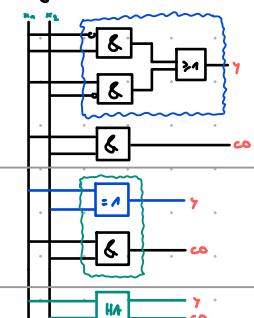
WT:	x_1	x_2	ci	y	co
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

DNF:

$$y = ((\bar{x}_1 \wedge \bar{x}_2) \vee (x_1 \wedge \bar{x}_2))$$

$$co = (x_1 \wedge x_2)$$

Schaltung:



DNF:

$$y = ((\bar{x}_1 \wedge \bar{x}_2 \wedge ci) \vee (\bar{x}_1 \wedge x_2 \wedge \bar{ci}) \vee (x_1 \wedge \bar{x}_2 \wedge \bar{ci}) \vee (x_1 \wedge x_2 \wedge ci))$$

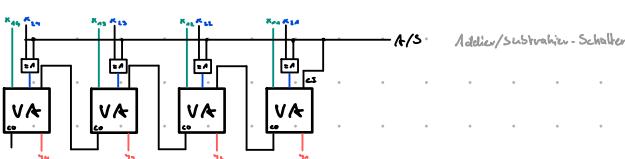
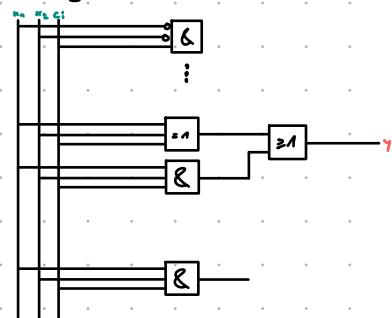
$$co = ((\bar{x}_1 \wedge x_2 \wedge ci) \vee (x_1 \wedge \bar{x}_2 \wedge ci) \vee (x_1 \wedge x_2 \wedge ci))$$

HA aus VA durch CI=0:

$$y = \begin{cases} 0 & \text{v. da } x_1 \wedge 0 = 0 \\ (x_1 \wedge x_2) \vee & \text{da } x_1 \wedge 1 = x_1 \\ (x_1 \wedge \bar{x}_2) \vee & \\ 0 & \end{cases} = ((\bar{x}_1 \wedge \bar{x}_2) \vee (x_1 \wedge \bar{x}_2))$$

$$co = \begin{cases} 0 & v \\ 0 & v \\ (x_1 \wedge x_2) \vee & \\ 0 & \end{cases} = (x_1 \wedge x_2)$$

Schaltung:



4-Bit Paralleladdierer und -subtrahierer:
Wertebereich: -8 bis 7

Over- bzw Underflow-Erkennung

K_{M4}	x_{M4}	y_4	O/u
0	0	0	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1 overflow
1	1	0	0
1	1	1	1 underflow

$$O/u = ((\bar{x}_{M4} \wedge \bar{x}_{M4} \wedge y_4) \vee (x_{M4} \wedge x_{M4} \wedge \bar{y}_4))$$

VNR S34+35

ECB S42, 46-57

8085

- Befehlsatz IP, SP was funktioniert ; M₂: LXI ; LDAx; MOV M,D
- Add+Sub Zwei Zahlen
- Inrementierung wie A+1 → MOV A,00 oder SUB A
- Sprungbefehl: bedingt, unbedingt
- Unterprogramm: call, ret
- Addition aufeinanderfolgender
- For Loop (multiplizieren)
- Callatz-Folge (vergrößert/größer)

VNR (Regeln)

1. 5 Funktionseinheiten: Steuerwerk, Rechenwerk, Speicher, Eingabewerk, Ausgabewerk

2. Struktur von Prozessen unabhängig

zur Lösung des P muss Bearbeitungsanweisung (das Programm [Folge von Maschinenelementen]) von außen eingegeben werden und im Speicher abgelegt werden

3. Programme und Daten im selben Speicher

4. Speicher in Zellen aufgeteilt. [1 Zelle = 1 Byte / 8 Bit] (fortlaufend durchnummeriert) \Rightarrow Processor Register: schnelle Speicher für Instructions, Daten oder Adressen

Über Adresse/Nummer einer Zelle kann ihr Inhalt abgelesen werden.

A (Akkuakkulator)

B C

D E

H L

F (Flag)

5. Ansprechen des nächsten Befehls vom Steuerwerk aus
↳ durch erhöhen/verändern der Befehlsadresse

6. aufeinanderfolgende Befehle werden in aufeinanderfolgenden Befehlsstellen abgelegt
! Abweichung durch Jump-Befehl

7. Befehle: arithmetische: Addieren, Multiplizieren, ...

logische: NOT, AND, ...

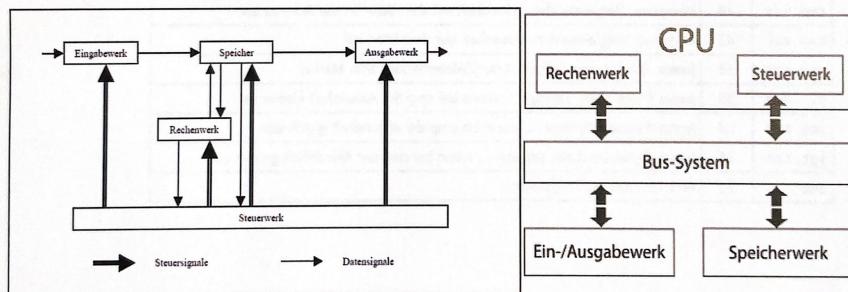
transport: von Speicher zu Rechenwerk, ...

bedingte, unbedingte Sprünge

8. Alle Daten, Befehle, Adressen binär codiert

→ Schaltwerte im Steuerwerk sorgen für richtige Interpretation (Decodierung)

Anordnungsmöglichkeiten



Assemblerbefehl $\xrightarrow{\text{Assembler}}$ Maschinenelement (Instruction)

Seiner Leser:

z.B. LD A X 3
load
Register A
Akkuakkulator
Registerpaar
3 und C

Folge von 0n und 1n
0000 1010

Bedeutung

Inhalt von A
 $cA = \langle\langle B, C \rangle\rangle$
Adressen
16 Bit
Inhalt der Speicheradresse
8 Bit

Befehlssatz Menge an Maschinenelementen die ein Prozessor ausführen kann

Registersatz Bezeichnung der Menge aller Register die ein Prozessor hat

1 Byte

	Bedeutung	Maschinenelement
MOV B,A	$cB = cA$	47
HLT	Programmstopp	76
ADD 3	$cA = cA + c3$	80
ADD M	$cA = cA + cH,L$	86
CMA	1er Komplement Akku	2F

$cA = \text{Akkuakkulator}$

$cI = \text{Immazet / Direktwert (nicht Adresse)} \quad LXi H, 1001 \rightarrow cH,L = 1001$

$cX = \text{Registerpaar } B,C,D,E,H,L // SP$

$M = cH,L$ Inhalt der Adresse die an H,L steht $\quad MOV M,E \quad ccH,L = cE$
→ z.B. $c18001 = cE$

2 Byte

$I = 2001$ $\Rightarrow 2 \text{ Bytes}$	$MUL I, 05$	$cB = 05 \text{ (hence)}$	0605
	ADI 07	$cA = cA + 07$	C607

3 Byte

LDA 1900	$cA = c1900$	$H \text{ und } L \text{ werden geladen}$
Direktwert	$cD,E = 1900$	$L \text{ vor } H$

Direktwert
L1
L2 und L3
Registerpaar
Load

$cD,E = 1900$
LW
 $cD = 19$
 $cE = 00$

Incrementierungsbefehle

INR A Increment Register A hius 3c
 A **[19]** → **[1A]**
 $cA = cA + 1$

INX H Increment Registerpaar H,L 23
 H **[19 1F]** L → H **[1A 00]** L
 $cH,L = cH,L + 1$

INR L
 H **[19 FF]** L → H **[1A 00]** L

INX H
 H **[19 FF]** L → H **[1A 00]** L

Unterprogramm

LXI SP, 1C00 n600
 Stackpointer SP setzt oben auf den Stack
 Initialisierung **[1C 00]**

180E CALL Mul → **[AB FE]**
1811 SUBC

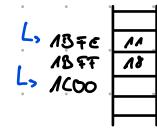
Hauptprogramm

Mul: Mloop: 1900 DCR C
 3M EndM
 ADD D
 JMP Mloop
 RET zu 1811
CA

Unterprogramm

- setzt den IP auf die Startadresse von "Mul"
- legt die Rücksprungadresse auf den Stack
- erhöht den Wert des SP

- Sprung zurück zu der Adresse die oben auf dem Stack liegt
- aktualisiert den Wert des SP



1A00, damit weit genug weg ist.

Befehlsatz

Menge an Maschinenelementen die ein Prozessor ausführen kann

→ Maschinenelement
Instruction Pointer

Ein Register in dem die Adresse des nächsten auszuführenden Maschinenelements gespeichert ist

Registersatz

Bezeichnung der Menge aller Register die ein Prozessor hat.

Stack Pointer

Ein Register in dem die Rückkehradresse nach einem Unterprogrammaufruf gespeichert ist.

Mit RET wird die obere Adresse angelesen und dahin zurückgekehrt.

Prozess

Programm zur Laufzeit

Thread

Ausführbare Einheit innerhalb eines Prozesses

Dsp: Addition S47

Multiplication S49

SP, CALL/RET S50

Add aufeinanderfolgende S51
for-loop

verschachtelte for-loop S53

Simultaneous Multithreading

Fähigkeit eines Prozessors mehrere

Threads gleichzeitig auszuführen

S.57 - 98