Gleitkommaaufgabe

Grundlagen Gleitkommadarstellung

$z = M \cdot B^E$

M: Mantisse

B: Basis (meist B=2)

E: Exponent

Bsp: $6{,}23_{10} = 623{,}0_{10} \cdot 10^{-2} = 0{,}623_{10} \cdot 10^{1} = 0{,}0623 \cdot 10^{2}$

⇒ Darstellung nicht eindeutig

Abhilfe: Normalisierung der Gleitkommazahl

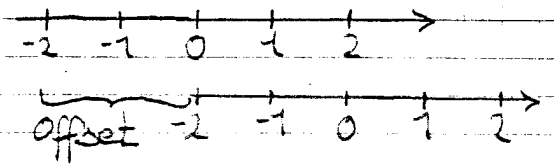
Bsp: - Mantisse $0 \leq M < 1$ (d.h. Komma vor Mantisse)

- 1. Nachkommastelle von 0 verschieden

(Achtung: bei IEEE gibt es hidden bit, d.h. es steht noch eine 1 vor Mantisse, die weggelassen wird (S.39f))

Meist wird statt des Exponenten die sog. Charakteristik C benutzt, damit im Exponenten nur pos. Zahlen stehen

Anschaulich:



$C = E + \left(\frac{1}{2} B^{c} - 1\right)$, wobei $c \hat{=}$ Anzahl der Bitstellen der Charakteristik

Gleitkommaarithmetik (S. 40 ff.)

Beachte: - bei Addition/Subtraktion Exponenten angleichen

- bei Multiplikation/Division Exponenten addieren/subtrahieren

- zum Schluss nochmals normalisieren

Probleme mit Gleitkommazahlen

- Zahlen wie $\frac{1}{3}$ nicht genau darstellbar
- liegen Zahlen weit auseinander gehen beim Angleichen der Exponenten Stellen verloren

Zahlendarstellung

jede Zahl lässt sich darstellen als:

$$Z = z_{n-1} B^{n-1} + \dots + z_1 B^1 + z_0 B^0 + z_{-1} B^{-1} + \dots + z_{-k} B^{-k}$$

$$160{,}464 = 1 \cdot 10^2 + 6 \cdot 10^1 + 0 \cdot 10^0 + 4 \cdot 10^{-1} + 6 \cdot 10^{-2} + 4 \cdot 10^{-3}$$

Horner-Schema:

$$160{,}464 = (1 \cdot 10 + 6) \cdot 10 + 0 + \left(4 + \left(6 \cdot \dots + 4 \cdot 10^{-1}\right) \cdot 10^{-1}\right) 10^{-1}$$

Umwandlung des ganzzahligen Anteils

1.)

29	: 16	= 1	
-16			
13	: 8	= 1	
-8			
5	: 4	= 1	
-4			
1	: 2	= 0	
1	: 1	= 1	

$(29)_{10} = (11101)_2$

2.) Horner-Schema

29 div 2 = 14		29 mod 2 = 1
14 div 2 = 7		14 mod 2 = 0
7 div 2 = 3		7 mod 2 = 1
3 div 2 = 1		3 mod 2 = 1
1 div 2 = 0		1 mod 2 = 1

$(29)_{10} = (11101)$

Erklärung:

$$29 = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 2^0$$

$$\left(\left(\left(1 \cdot 2^1 + 1\right) 2^1 + 1\right) 2^1 + 0\right) 2^1 + 1 \cdot 2^0$$

gebrochener Anteil

$0{,}4 \cdot 2$ ← Basis	$= 0{,}8$	$\rightarrow 0$
$0{,}8 \cdot 2$	$= 1{,}6$	$\rightarrow 1$
$0{,}6 \cdot 2$	$= 1{,}2$	$\rightarrow 1$
$0{,}2 \cdot 2$	$= 0{,}4$	$\rightarrow 0$
$0{,}4 \cdot 2$	$= 0{,}8$	$\rightarrow 0$

Abbruch

$(0{,}4)_{10} = (0{,}01100\overline{1})_2$

s. 32

Umwandlung in Hexadezimal- oder Oktalzahlen leicht möglich, da 8 bzw. 16 Zweierpotenzen sind
→ B.

Zahlendarstellung

- mit Vorzeichen (linkes Bit wird als −, bzw + interpretiert)
  - 0 hat zwei Darstellungen +0, −0
  - Digitalrechner muss addieren/subtrahieren können
- Einerkomplement (führt Subtraktion auf Addition zurück)
  - Umwandlung einer Dualzahl durch Bitweises komplementieren

$K_1(10110) = (01001)$

- Zweierkomplement: $K_2(z) = K_1(z) + 1$

$K_2(10110) = (01010)$

+ jede Darstellung einer Zahl eindeutig

~~$K_2(0000) = 1000$~~

Subtraktion durch Addition des Zweierkomplements

$111 \hat{=} 7$ $\quad 7 = (111)_2$ $\quad \overset{7}{+(10-6)} \hat{=} K_{10}(6)$

$+ \overset{010}{110} \overset{1}{=} -6$ $\quad 6 = (110)_2$ $\quad 111 = 1$

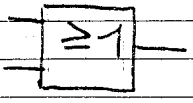
$(1001 \overset{1}{=} 1$

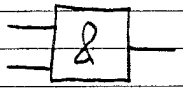
$7 - 6 = 7 + K_2(6)$ $\quad K_2(6) = 010$

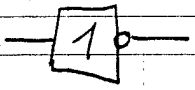
↳ 1: positives Ergebnis

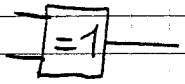
0: negatives Ergebnis ⇒ bilde wieder $K_2(z)$

Logische Schaltungen

$a + b$ 

$a \cdot b$ 

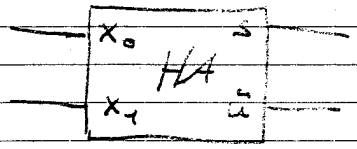
$\bar{a}$ 

$a \oplus b$ 

Halbaddierer

$s = x_1 \oplus x_0 \quad ; \quad \ddot{u} = x_1 x_0$

x_1	x_0	s	$\ddot{u}$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Volladdierer

$s = \bar{x}_1 \bar{x}_0 \ddot{u}' + \bar{x}_1 x_0 \bar{\ddot{u}}' + x_1 \bar{x}_0 \bar{\ddot{u}}' + x_1 x_0 \ddot{u}'$

(in Wertetabelle noch $\ddot{u}'$ dazu)

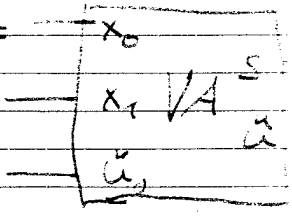
$\ddot{u} = \bar{x}_1 x_0 \ddot{u}' + x_1 \bar{x}_0 \ddot{u}' + x_1 x_0 \bar{\ddot{u}}' + x_1 x_0 \ddot{u}'$

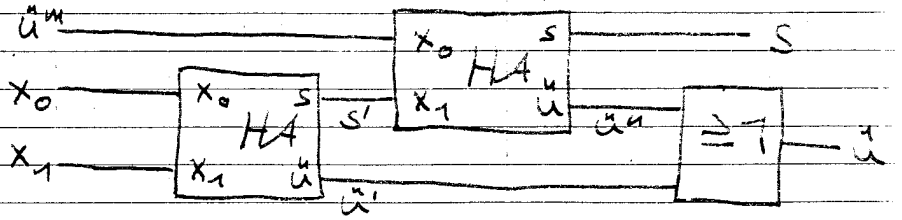
$\rightarrow s = (x_1 \leftrightarrow x_0)\ddot{u}' + (x_1 \oplus x_0)\bar{\ddot{u}}'$

Äquivalenz — Antivalenz

$= (x_1 \oplus x_0) \oplus \ddot{u}'$

$\ddot{u} = (x_1 \oplus x_0)\ddot{u}' + x_1 x_0$

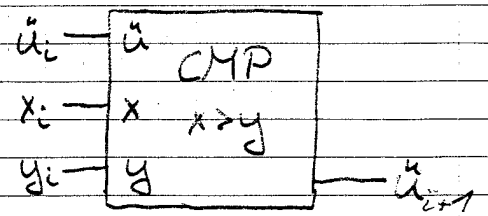




Vergleicher

$\ddot{u}_{i+1} = x_i \bar{y}_i + \ddot{u}_i (x_i + \bar{y}_i)$

$\ddot{u}_i$	x_i	y_i	$\ddot{u}_{i+1}$

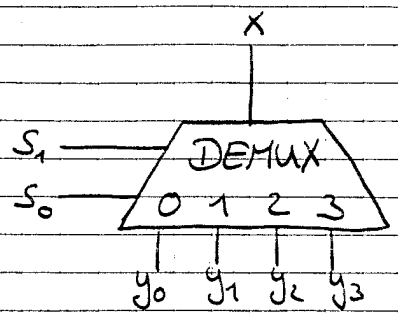


Laufzeit

$t_l = n_k \cdot n_s \cdot t$ ← Gatterlaufzeit

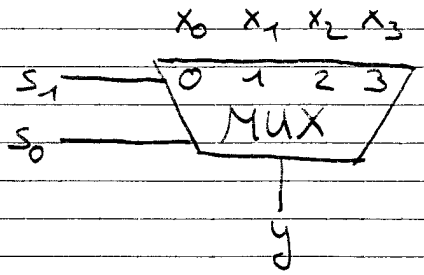
Demultiplexer

s_1	s_0	y_0	y_1	y_2	y_3
0	0	x	0	0	0
0	1	0	x	0	0
1	0	0	0	x	0
1	1	0	0	0	x

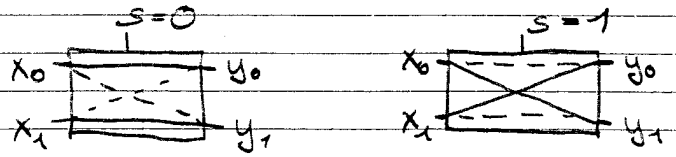


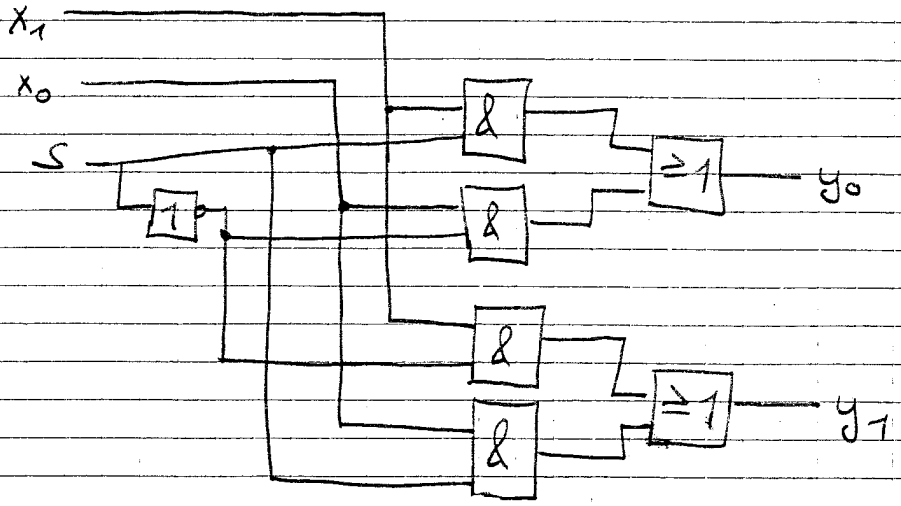
Multiplexer

s_1	s_0	y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3



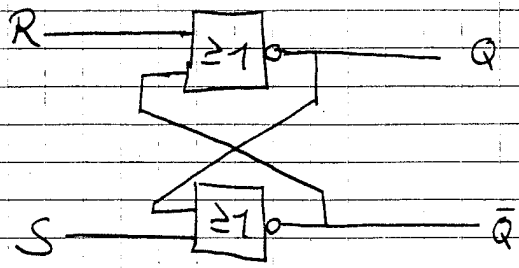
Kreuzschienenverteiler (crossbar switch)

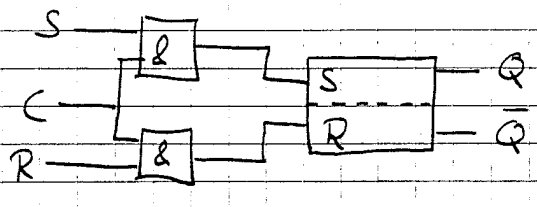


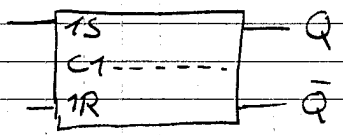


Speicherglieder

RS-Flipflop

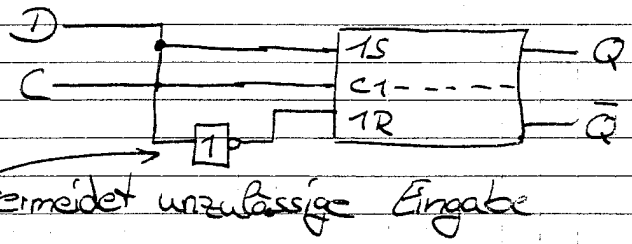




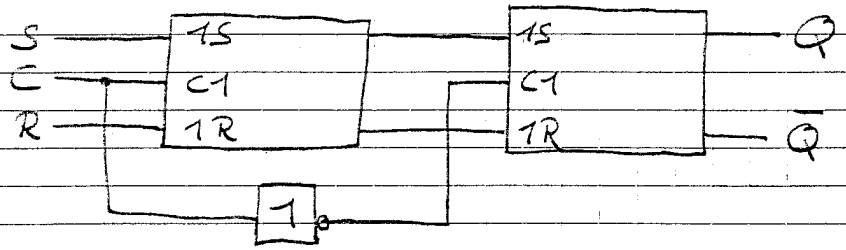


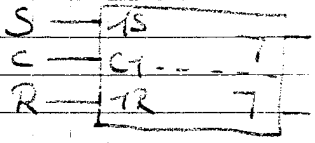
S	R	Q
0	0	speichern (wie vorher)
0	1	0
1	0	1
1	1	(0) unzulässig

D-Flipflop

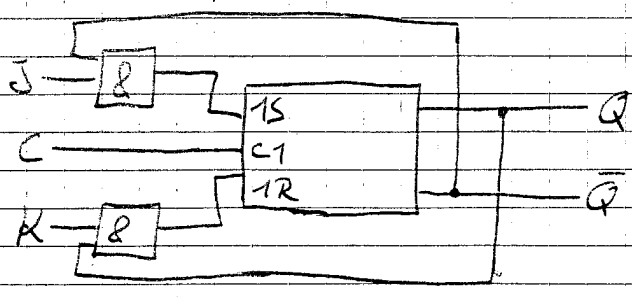


RS-Flipflop mit Zwei-Zustandssteuerung (rückflankengetriggert)





JK-Flipflop



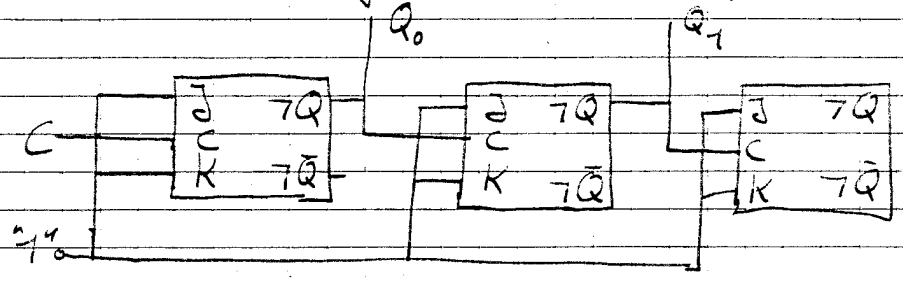
J	K	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	$\bar{Q}_n$

$Q_n \rightarrow Q_{n+1}$	J
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	*
$1 \rightarrow 1$	*

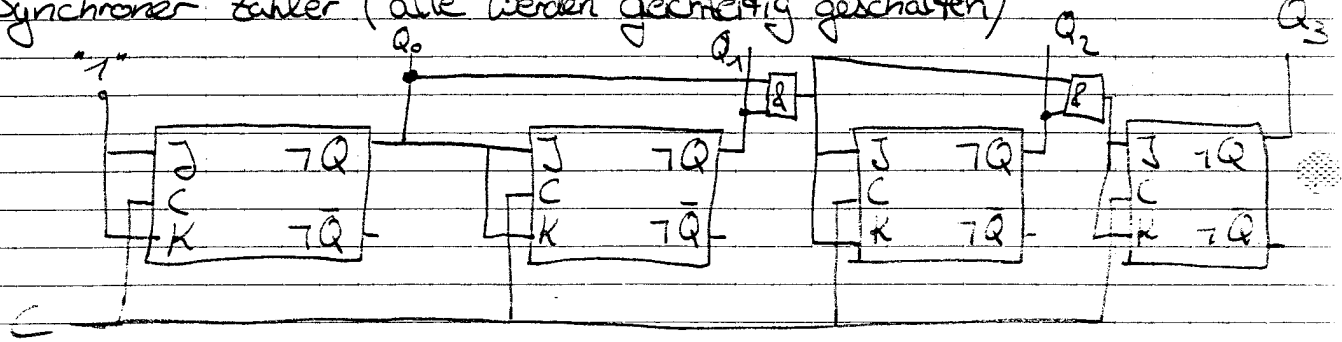
Zähler

Asynchrone Zähler (Schalten erfolgt in zeitlicher Abfolge)

Aufwärtszähler:

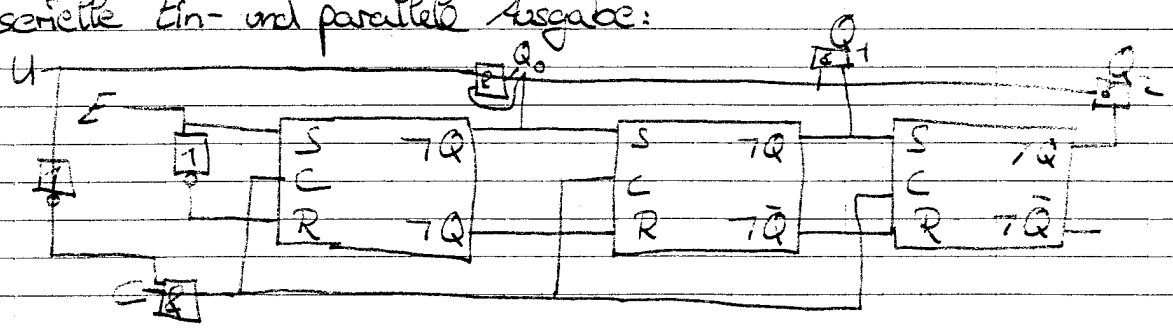


Synchroner Zähler (alle werden gleichzeitig geschaltet)

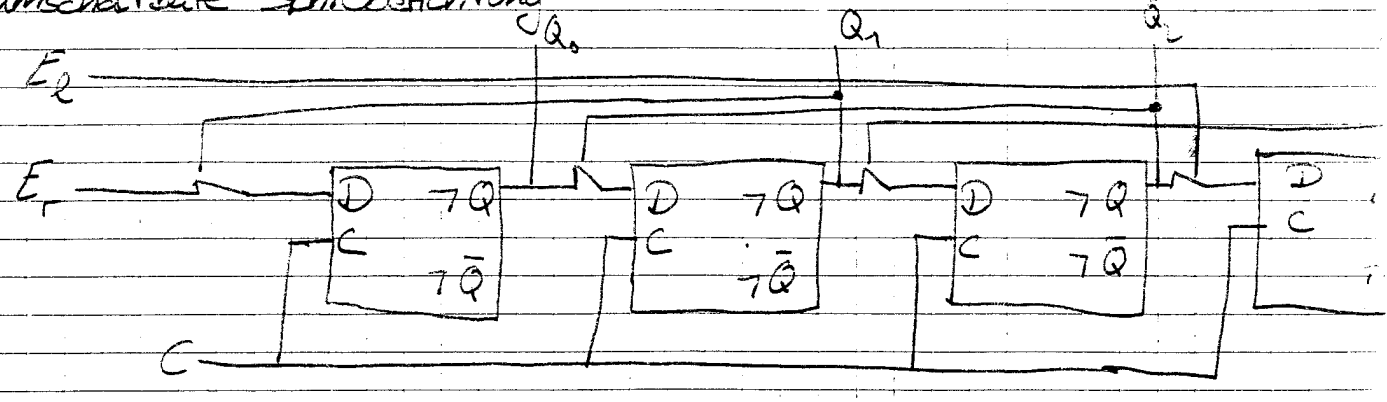


Schieberegister

serielle Ein- und parallele Ausgabe:



umschaltbare Schieberichtung



Programmierbare Logik

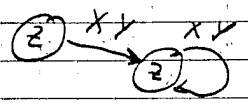
PROM (Programmable read-only-Memory) rechts programmieren

Automaten

$A = (X, Y, z, f, g)$ Quintupel des Automaten, beschreibt diesen vollständig

Darstellungsweisen

- Automatengraph



- Automatentabelle

Übergangstabelle

n	n+1	
	X_0	X_1
z_0	z_0	z_1
z_1	z_1	z_0

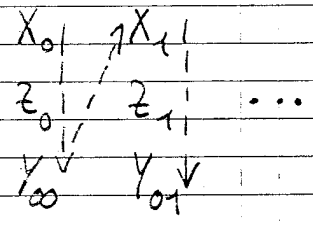
Ausgangstabelle

n	n+1	
	X_0	X_1
z_0	Y_{00}	Y_{01}
z_1	Y_{11}	Y_{10}

Automatentabelle

n	n+1	
	X_0	X_1
z_0	z_0/Y_{00}	z_1/Y_{01}
z_1	z_1/Y_{11}	z_0/Y_{10}

- Automatenband



Mealy-Automat allgemeinste Automat (Ausgabe und Folgezustand abhängig von Eingabe und Zustand), d.h. Zustand hat versch. Y_n

Moore-Automat g hängt nur von Zustand z und nicht von X ab, d.h. jedem Zustand ist nur 1 Y zugewiesen

Zuordner ordnet Eingabe eine Ausgabe zu

Autonomer Automat gibt nur ein Eingabeelement (z.B. Zähler)

Halb Automat nur eine oder keine Ausgabe

Akzeptor kein Quintupel, wird zwischen akzeptierten und zurückgewiesenen Eingaben unterschied

Umwandlung Moore -> Mealy

- Anzahl der Zustände bleibt gleich
- Jede Kante mit dem Ausgabeelement bezeichnen, auf den sie zeigt
- Ausgabeelemente aus Knoten entfernen

Umwandlung Mealy -> Moore

- Zustandsmenge ändert sich (wird größer)
- Kreuzprodukt aus Eingangselement und altem Zustand wird Zustand des Moore-Automaten zugeordnet

	x_0	x_1
z_0	$z_1 \mid y_2$	$z_1 \mid y_1$
z_1	$z_0 \mid y_3$	$z_1 \mid y_4$

	x_0	x_1	
z_{00} ($z_0 \times x_0$)	z_{10}	z_{11}	y_2
z_{01} ($z_0 \times x_1$)	z_{10}	z_{11}	y_1
z_{10} ($z_1 \times x_0$)	z_{00}	z_{01}	y_3
z_{11} ($z_1 \times x_1$)	z_{10}	z_{11}	y_4

- Betrachte zunächst den Folgezustand z. x_0, z_0 dieser ist $z_{1,1}$
- Suche dann aus dem Moore Automaten die Kombinationen $z_1 \times x_0$ $z_1 \times x_1$ heraus, die sind die gesuchten Zustände

Zustandsreduktion

z_0, z_1 äquivalent → Automat reagiert bei unterschiedlichen Eingabe mit gleicher Ausgabe

k-äquivalent → reagiert nach k Eingaben gleich

- Bilde K-Äquivalenzklassen (alle die gleiche Ausgabe haben)
  K_0 ← Endindex, K_1 ← Index der Äquivalenz
- hinter jeden ~~Ausgabe~~ (n+1)-Zustand die k des Zustandes schreiben

Statusregister (S. 124)

C = Carry (Übertragsbit) $\quad C = c_n$

N = Negative (Ergebnis ist negativ) $\quad N = s_{n-1}$

Z = Zero (Ergebnis nur Nullen) $\quad Z = \prod_{i=0}^{n-1} \bar{s}_i$

V = oVerflow (Überlauf) $\quad V = c_n \bar{c}_{n-1} + \bar{c}_n c_{n-1}$

Relation	Statusregisterbelegung
$x = y$	$Z = 1$
$x \neq y$	$Z = 0$
$x \geq y$	$N = V$
$x < y$	$N \neq V$
$x > y$	$N = V$ und $Z = 0$
$x \leq y$	entweder $N \neq V$ oder $Z = 1$

$C = c_n$: Übertragsbit gesetzt, wenn eine 1 "rausgeschoben" wird ($c_n = 1$)

```
  10
+ 10
(1)00   ≙ c
(1)00   ≙ s
```

$N = s_{n-1}$: Negativebit gesetzt, wenn most significant bit 1 ist

$Z = \prod_{i=0}^{n-1} \bar{s}_i$: Zerobit gesetzt, wenn Ergebnis nur aus Nullen besteht

$V = c \oplus c_{n-1}$: Overflowbit gesetzt, wenn eines der zwei vorderen c-Bits eine 1 ist, nicht bei beiden

X-Register	LB:	R26	0x1A
	HB:	R27	0x1B
Y-Register	LB:	R28	0x1C
	HB:	R29	0x1D
Z-Register	LB:	R30	0x1E
	HB:	R31	0x1F

Stack: LIFO-Prinzip (sichert Rücksprungadresse, wird zum Unterfunktionsaufruf gebraucht)

Stackpointer Initialisierung:

```
.include "m8def.inc"
LDI   r16, LOW(RAMEND)   ; lädt Lowbyte der höchsten Speicheradresse
OUT   SPL, r16           ; schreibt es in Stackpointer
LDI   r16, HIGH(RAMEND)  ; lädt Highbyte der höchsten Speicheradresse
OUT   SPH, r16           ; schreibt es in Stackpointer
```

Sofort-Operanden-Befehle: K → wird als Zahl aufgefasst

Direkte Adressierung: k (Doppelwortbefehle) → Adresse im Speicher

Relative Adressierung: k → Bits werden auf aktuelle Adresse addiert

Indexregister: aktuelle Wert des Z-Registers wird als Adresse interpretiert

Indexregister mit Offset: wird noch ein Offset (q) hinzu addiert