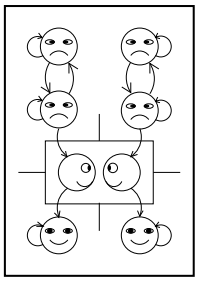
Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung
Institut für Technische Informatik und Ingenieurinformatik
Fachgebiet Integrierte Kommunikationssysteme



Arbeitsblätter

zur Lehrveranstaltung

Technische Informatik

(Basic Engineering School)

(Ausgabe September 2020)

Dr.-Ing. Heinz-Dietrich Wuttke

Dr.-Ing. Prof. h. c. Karsten Henke

Inhaltsübersicht

+++ Datei: deckbl.ti-bes +++ Datum: 8. September 2020 +++

BOOLEsche Mengenalgebra (BMA)

Algebra — Trägermenge und Relationen bzw. Operationen mit bestimmten Eigenschaften (Axiome) sowie neutrale Elemente bezüglich dieser Operationen definieren eine Algebra.

$BMA = [\mathcal{P}(M), \cup, \cap, \bar{}\,, \emptyset, M]$ mit:

- $\mathcal{P}(M)$ als Trägermenge
- $\cup, \cap, \bar{}$ als Operationen
- $\emptyset$ als neutrales Element der Vereinigung ($\cup$)
- M als neutrales Element der Schnittbildung ($\cap$)

Axiome der BOOLEschen Mengenalgebra, wobei $A, B, C \in \mathcal{P}(M)$

Kommutativität
$A \cup B = B \cup A$
$A \cap B = B \cap A$

Assoziativität
$A \cup (B \cup C) = (A \cup B) \cup C = A \cup B \cup C$
$A \cap (B \cap C) = (A \cap B) \cap C = A \cap B \cap C$

Distributivität
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Idempotenz
$A \cup A = A$
$A \cap A = A$

Adjunktivität
$A \cap (A \cup B) = A$
$A \cup (A \cap B) = A$

Komplementarität
$A \cup \bar{A} = M$
$A \cap \bar{A} = \emptyset$

Wichtige Regeln der BOOLEschen Mengenalgebra

*deMORGAN*sche Regel
$\overline{A \cup B} = \bar{A} \cap \bar{B}$
$\overline{A \cap B} = \bar{A} \cup \bar{B}$

Vereinigungsregel
$A \cup \emptyset = A$
$A \cup M = M$

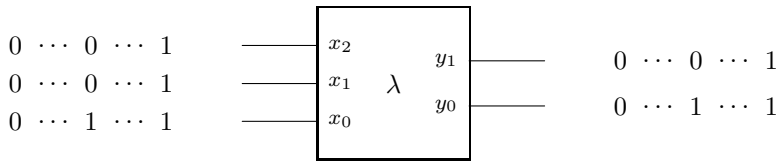
Schnittregel
$A \cap \emptyset = \emptyset$
$A \cap M = A$

Komplementregel
$\overline{\overline{A}} = A$
$\overline{\emptyset} = M$
$\overline{M} = \emptyset$

Anwendung der BMA auf die Funktionsbeschreibung digitaler Schaltungen (kombinatorisch)

Digitale Schaltung

$X = \{X_0, \cdots, X_i, \cdots, X_{2^n-1}\}$ $\qquad$ $Y = \{Y_0, \cdots, Y_t, \cdots, Y_{2^m-1}\}$



mit:

- X – Menge der Eingangsbelegungen X_i des Eingangsvektors $x = [x_{n-1}, \ldots, x_r, \ldots, x_0]$
- Y – Menge der Ausgangsbelegungen Y_t des Ausgangsvektors $y = [y_{m-1}, \ldots, y_k, \ldots, y_0]$
- λ – Abbildungsvorschrift (Funktion der Schaltung):

$\lambda: X \Rightarrow Y$	bzw.	$\lambda(X) = Y$	(determiniert)
$\lambda: X \Rightarrow \mathcal{P}(Y)\backslash\emptyset$	bzw.	$\lambda(X) = \mathcal{P}(Y)\backslash\emptyset$	(nichtdeterminiert)

Definitionen

Ein − /Ausgangsvektor $\quad x = [x_{n-1}, \ldots, x_r, \ldots, x_0]$ bzw. $y = [y_{m-1}, \ldots, y_k, \ldots, y_0]$
aus binären Variablen x_r bzw. y_k mit n bzw. m als Stelligkeit von x bzw. y

Eingangsbelegung $\quad X_i(x) \Rightarrow \{0,1\}^n$ $\quad$ geordnete Menge von n bits; $n = |x|$
$X_i = [X_i(x_{n-1}), \ldots, X_i(x_r), \ldots, X_i(x_0)]$

bit der Eingangsbelegung $\quad X_i(x_r) \in \{0,1\}$ $\quad$ Wert der Eingangsvariablen x_r in der Belegung X_i

Ausgangsbelegung $\quad Y_t(y) \Rightarrow \{0,1\}^m$ $\quad$ (determiniert)
$Y_t(y) \Rightarrow \{0,1,*,H(g)\}^m$ $\quad$ (nichtdeterminiert siehe Beispiel Seite 3)
$Y_t = [Y_t(y_{m-1}), \ \ldots\ , Y_t(y_k), \ \ldots, Y_t(y_0)]$
$= [\lambda_{m-1}(X_i), \ldots, \lambda_k(X_i), \ldots, \lambda_0(X_i)] = \lambda(X_i)$

bit der Ausgangsbelegung $\quad Y_t(y_k) \in \{0,1\}$ $\quad$ (determiniert)
$Y_t(y_k) \in \{0,1,*,H(g)\}$ $\quad$ (nichtdeterminiert)

Belegungsmenge $\quad X = \{X_i \mid 0 \le i \le 2^n - 1\}$ $\qquad$ $Y = \{Y_t \mid 0 \le t \le 2^m - 1\}$

Belegungsindizes $\quad i = \sum_{r=0}^{n-1} X_i(x_r) \cdot 2^r$ $\qquad$ $t = \sum_{k=0}^{m-1} Y_t(y_k) \cdot 2^k$

Indexmenge $\quad$ Menge der Indizes der Eingangsbelegungen
$M = \{i \mid 0 \le i \le 2^n - 1\}$; $\qquad |M| = |X|$

Wertetabelle

$\lambda : X \Rightarrow Y$ Belegung Y des Ausgangsvektors y als Funktion $\lambda(X)$ der Belegung X des Eingangsvektors x.

BI	x_{n-1}	$\cdots$	x_r	$\cdots$	x_0	y_{m-1}	$\cdots$	y_k	$\cdots$	y_0	BI
0	0	$\cdots$	0	$\cdots$	0						
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
i	$X_i(x_{n-1})$	$\cdots$	$X_i(x_r)$	$\cdots$	$X_i(x_0)$	$\lambda_{m-1}(X_i)$	$\cdots$	$\lambda_k(X_i)$	$\cdots$	$\lambda_0(X_i)$	t
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
2^n-1	1	$\cdots$	1	$\cdots$	1						

Beispiel: Funktion mit 3 Eingangs- und 2 Ausgangsvariablen

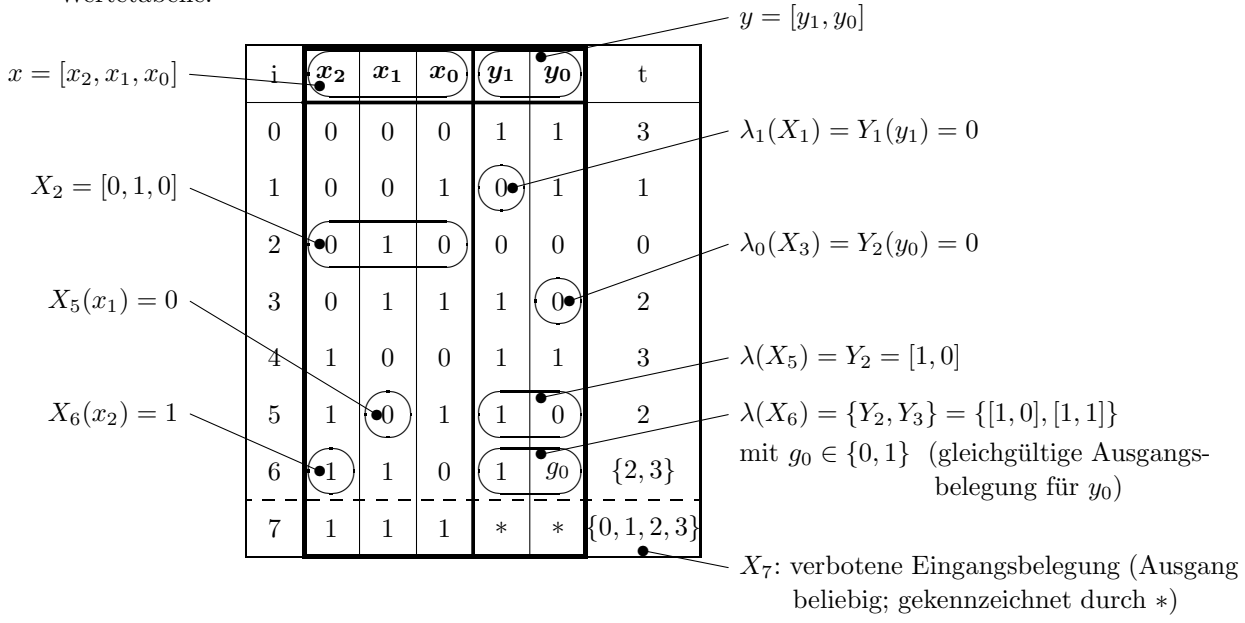
Ein-/Ausgangsvektor: $x = [x_2, x_1, x_0]$ mit $|x| = 3$ bzw. $y = [y_1, y_0]$ mit $|y| = 2$

Eingangsbelegung: $X_5(x) = [1, 0, 1]$ als geordnete Menge von $n = |x| = 3$ bits mit dem Belegungsindex $i = 5 = \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0$)

bit der Eingangsbelegung: $X_5(x_0) = 1$ das "0"-te bit der Eingangsbelegung X_5

Ausgangsbelegung: $\lambda(X_5) = [\lambda_1(X_5), \lambda_0(X_5)] = [Y_2(y_1), Y_2(y_0)] = [1, 0] = Y_2$

Wertetabelle:



i	x_2	x_1	x_0	y_1	y_0	t
0	0	0	0	1	1	3
1	0	0	1	0	1	1
2	0	1	0	0	0	0
3	0	1	1	1	0	2
4	1	0	0	1	1	3
5	1	0	1	1	0	2
6	1	1	0	1	g_0	$\{2, 3\}$
7	1	1	1	$*$	$*$	$\{0, 1, 2, 3\}$

Belegungsmengen – schaltalgebraische Ausdrücke

schaltalgebraischer Ausdruck

$h_i(x), h_j(x) \in H$ sind elementare oder zusammengesetzte schaltalgebraische Ausdrücke und sind folgendermaßen induktiv definiert:

1. Konstante 0 und 1 sind schaltalgebraische Ausdrücke;
2. binäre Variable x_r eines n-stelligen Vektors x sind schaltalgebraische Ausdrücke;
3. wenn $h_i(x)$ und $h_j(x)$ Ausdrücke sind, so auch:

   $\overline{h_i(x)}$, $(h_i(x) \to h_j(x))$,
   $(h_i(x) \wedge h_j(x))$, $(h_i(x) \sim h_j(x))$,
   $(h_i(x) \vee h_j(x))$, $(h_i(x) \not\sim h_j(x))$

4. andere Zeichenketten sind keine schaltalgebraischen Ausdrücke.

Wertbestimmung für schaltalgebraische Ausdrücke

Wertfunktion

Die Wertfunktion $\mathcal{W}$ ordnet einem *schaltalgebraischen Ausdruck* $h_j \in H$ bei einer *Belegung* $X_i \in X$ einen Wert aus $\{0, 1\}$ zu.

$\mathcal{W} : H \times X \Rightarrow \{0, 1\}$ z.B. $\mathcal{W}(h_j(x), X_i) = 0$

X − Partitionierung

Jeder schaltalgebraische Ausdruck $h_j(x)$ teilt die Belegungsmenge X des Vektors x disjunkt in zwei Teilmengen X^1 und X^0, wobei gilt:

- $X = X^1 \cup X^0; \quad X^1 \cap X^0 = \emptyset; \quad X^1 = \overline{X^0}$
- $\forall X_i (X_i \in X^1 \leftrightarrow \mathcal{W}(h_j(x), X_i) = 1)$
- $\forall X_i (X_i \in X^0 \leftrightarrow \mathcal{W}(h_j(x), X_i) = 0)$

Wertbestimmung

$\forall X_i (\mathcal{W}(h_j(x), X_i) = 1 \leftrightarrow p_j(X_i))$

sprich: *Für alle Belegungen X_i gilt: Der Wert eines schaltalgebraischen Ausdrucks $h_j(x)$ bei der Belegung X_i ist 1, genau dann, wenn die von X_i abhängige Aussage $p_j(X_i)$ wahr ist.*

	$h_j(x)$	$p_j(X_i)$
(a)	0	f
(b)	1	w
(c)	x_k	$X_i(x_k) = 1$
(d)	$\overline{x_k}$	$X_i(x_k) = 0$
(e)	$\overline{h_k(x)}$	$\overline{\mathcal{W}(h_k(x), X_i) = 1}$
(f)	$h_k(x) \wedge h_l(x)$	$(\mathcal{W}(h_k(x), X_i) = 1) \wedge (\mathcal{W}(h_l(x), X_i) = 1)$
(g)	$h_k(x) \vee h_l(x)$	$(\mathcal{W}(h_k(x), X_i) = 1) \vee (\mathcal{W}(h_l(x), X_i) = 1)$

z.B. Zeile (d): $\forall X_i \big(\mathcal{W}(\overline{x_k}, X_i) = 1 \leftrightarrow X_i(x_k) = 0\big)$

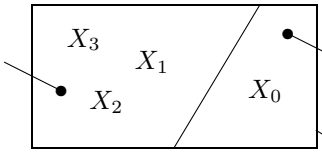
verbal: *Der Wert des Ausdrucks $\overline{x_k}$ bei der Belegung X_i ist gleich 1, genau dann, wenn bit k der Belegung X_i gleich 0 ist.*

Beispiel

$h_i = x_0 \vee x_1 \quad x = [x_1, x_0] \quad X = \{X_3, X_2, X_1, X_0\}$

X-Partitionierung

$h_i = x_0 \vee x_1 \;\hat{=}\; X^1$



$\overline{h_i} = \overline{x_0 \vee x_1} = \overline{x_0} \wedge \overline{x_1} \;\hat{=}\; \overline{X^1} = X^0$

$X = X^1 \cup X^0$

Wertbestimmung

$\mathcal{W}\big((x_0 \vee x_1), X_0\big) = \mathcal{W}(x_0, X_0) \vee \mathcal{W}(x_1, X_0) = X_0(x_0) \vee X_0(x_1) = 0 \vee 0 = 0$

$\mathcal{W}\big((x_0 \vee x_1), X_2\big) = \mathcal{W}(x_0, X_2) \vee \mathcal{W}(x_1, X_2) = X_2(x_0) \vee X_2(x_1) = 0 \vee 1 = 1$

Verallgemeinerte Werteverlaufsgleichheit ($\underset{*}{=}$)

Werteverlaufsgleichheit bezüglich 0 ($\underset{o}{=}$) ("streng")

$h_i(x) \underset{o}{=} h_j(x) \leftrightarrow \forall X_k(\mathcal{W}(h_i(x), X_k) = \mathcal{W}(h_j(x), X_k))$

Werteverlaufsgleichheit bezüglich h^* ($\underset{*}{=}$) ("verallgemeinert")

$h_i(x) \underset{*}{=} h_j(x) \leftrightarrow \forall X_k(\mathcal{W}(h_i(x), X_k) = \mathcal{W}(h_j(x), X_k) \vee \mathcal{W}(h^*(x), X_k) = 1)$

$h_i(x) \underset{*}{=} h_j(x)$ d.h. $h_i \vee h^\bullet \underset{o}{=} h_j \vee h^\bullet$ mit $h^\bullet = h_i \not\sim h_j$ und $h^\bullet \rightarrow h^* \underset{o}{=} 1$

Verallgemeinerte Umformungsregeln

mit $h^\bullet \rightarrow h^* \underset{o}{=} 1$ gilt: $h_i \underset{*}{=} h_i \vee h^\bullet$

$h_i \underset{*}{=} h_i \wedge \overline{h^\bullet}$

Verallgemeinerte X-Partitionierung

$X = X^1 \cup X^0 \cup X^*$ $M = M^1 \cup M^0 \cup M^*; \quad M^1 \cap M^0 = \emptyset$

$X_i \in X^1 \leftrightarrow \mathcal{W}(h_j, X_i) \underset{o}{=} 1$

$X_i \in X^0 \leftrightarrow \mathcal{W}(h_j, X_i) \underset{o}{=} 0$

$X_i \in X^* \leftrightarrow \mathcal{W}(h_j, X_i) =$ nicht determiniert

$X_i \in X^\bullet \leftrightarrow \mathcal{W}(h_j, X_i) \underset{*}{=} 1$ mit $X^\bullet \subseteq X^*$

Beispiel verallgemeinerte Werteverlaufsgleichheit zweier Ausdrücke

BI	x_2	x_1	x_0	h_i	h_j	h^*	$h^\bullet$
0	0	0	0	1	1	0	0
1	0	0	1	1	1	1	0
2	0	1	0	1	0	1	1
3	0	1	1	0	1	1	1
4	1	0	0	0	0	1	0
5	1	0	1	0	0	0	0
6	1	1	0	0	0	1	0
7	1	1	1	1	1	1	0

$h_i(x) = \overline{x_2}(\overline{x_1} \vee \overline{x_0}) \vee x_2 x_1 x_0$

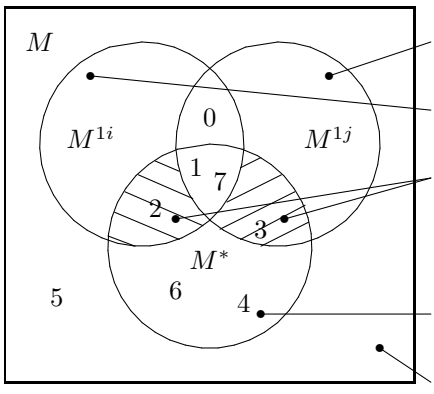
$h_j(x) = \overline{x_2}(\overline{x_1} \vee x_0) \vee x_1 x_0$

$h^*(x) = x_1 \vee \overline{x_2} x_0 \vee x_2 \overline{x_0}$

$h_i \not\underset{o}{=} h_j$ aber

$h_i \underset{*}{=} h_j$ denn

$h_i \vee h^* \underset{o}{=} h_j \vee h^*$



$h_j = \overline{x_2}(\overline{x_1} \vee x_0) \vee x_1 x_0 \;\hat{=}\; X^{j1} = \{X_0, X_1, X_3, X_7\}$

$h_i = \overline{x_2}(\overline{x_1} \vee \overline{x_0}) \vee x_2 x_1 x_0 \;\hat{=}\; X^{i1} = \{X_0, X_1, X_2, X_7\}$

$h^\bullet = \overline{x_2} x_1 \;\hat{=}\; X^\bullet = X^{i\bullet} \cup X^{j\bullet} = \{X_2, X_3\}$

$h^* = x_1 \vee \overline{x_2} x_0 \vee x_2 \overline{x_0} \;\hat{=}\; X^* = \{X_1, X_2, X_3, X_4, X_6, X_7\}$

$\overline{(h_i \vee h_j \vee h^*)} = x_2 \overline{x_1} x_0 \;\hat{=}\; X^0 = \{X_5\}$

BOOLEsche Ausdrucksalgebra (BAA)

$BAA = [K, \vee, \wedge, \bar{}, 0, 1]$ mit:

- K als Menge von Repräsentanten h_i je einer unendlichen Menge H^i werteverlaufsgleicher Ausdrücke a_i, wobei gilt:
  - $a_i \underset{o}{=} h_i \leftrightarrow h_i, a_i \in H^i \quad |H^i| = \infty \; , |X| = 2^n \; , 0 \le i \le |K| - 1$
  - $|K| = |P(X)| = 2^{2^n}$ mit n = Anzahl der x-Variablen

  z.B. für $n = 2$: $|X| = 2^2 = 4$; $|K| = 2^{2^2} = 16$ (siehe auch Seite 11)
  $K = \{h_0, h_1, \ldots, h_{15}\}$
  $H^9 = \{x_1 x_0 \vee \overline{x_1}\;\overline{x_0}, (\overline{x_1} \vee x_0)(x_1 \vee \overline{x_0}), x_1 \sim x_0, \ldots\}$
  $h_9 = x_1 x_0 \vee \overline{x_1}\;\overline{x_0}$ als Repräsentant aus H^9 in DNF
- $\vee, \wedge, \bar{}$ als Operationen
- 0 als neutrales Element der Disjunktion ($\vee$)
- 1 als neutrales Element der Konjunktion ($\wedge$)

Axiome und Regeln der BOOLEschen Ausdrucksalgebra

Kommutativität

$$h_i \vee h_j \underset{o}{=} h_j \vee h_i$$
$$h_i \wedge h_j \underset{o}{=} h_j \wedge h_i$$

Assoziativität

$$h_i \vee (h_j \vee h_k) \underset{o}{=} (h_i \vee h_j) \vee h_k \underset{o}{=} h_i \vee h_j \vee h_k$$
$$h_i \wedge (h_j \wedge h_k) \underset{o}{=} (h_i \wedge h_j) \wedge h_k \underset{o}{=} h_i \wedge h_j \wedge h_k$$

Distributivität

$$h_i \vee (h_j \wedge h_k) \underset{o}{=} (h_i \vee h_j) \wedge (h_i \vee h_k)$$
$$h_i \wedge (h_j \vee h_k) \underset{o}{=} (h_i \wedge h_j) \vee (h_i \wedge h_k)$$

Idempotenz

$$h_i \vee h_i \underset{o}{=} h_i$$
$$h_i \wedge h_i \underset{o}{=} h_i$$

Adjunktivität

$$h_i \wedge (h_i \vee h_j) \underset{o}{=} h_i$$
$$h_i \vee (h_i \wedge h_j) \underset{o}{=} h_i$$

Negation

$$h_i \vee \overline{h_i} \underset{o}{=} 1$$
$$h_i \wedge \overline{h_i} \underset{o}{=} 0$$
$$\overline{\overline{h_i}} \underset{o}{=} h_i$$
$$\overline{0} \underset{o}{=} 1$$
$$\overline{1} \underset{o}{=} 0$$

*Disjunktions*regel	$h_i \vee 0 \underset{o}{=} h_i$
	$h_i \vee 1 \underset{o}{=} 1$
*Konjunktions*regel	$h_i \wedge 0 \underset{o}{=} 0$
	$h_i \wedge 1 \underset{o}{=} h_i$
*deMORGAN*sche Regel	$\overline{h_i \vee h_j} \underset{o}{=} \overline{h_i} \wedge \overline{h_j}$
	$\overline{h_i \wedge h_j} \underset{o}{=} \overline{h_i} \vee \overline{h_j}$
Implikationsregel	$h_i \rightarrow h_j \underset{o}{=} \overline{h_i} \vee h_j$
Äquivalenzregel	$h_i \sim h_j \underset{o}{=} h_i h_j \vee \overline{h_i}\ \overline{h_j}$
Antivalenzregel	$h_i \not\sim h_j \underset{o}{=} \overline{h_i \sim h_j} \underset{o}{=} h_i \overline{h_j} \vee \overline{h_i} h_j$

Wichtige Kürzungsregeln

1. $h_i h_j \vee \overline{h_i} h_j \underset{o}{=} (h_i \vee h_j)(\overline{h_i} \vee h_j) \underset{o}{=} h_j$
2. $h_i \vee h_i h_j \underset{o}{=} h_i(h_i \vee h_j) \underset{o}{=} h_i$
3. $h_i \vee \overline{h_i} h_j \underset{o}{=} h_i \vee h_j$
4. $h_i(\overline{h_i} \vee h_j) \underset{o}{=} h_i h_j$
5. $h_i h_j \vee h_i \overline{h_k} \vee h_j h_k \underset{o}{=} h_i \overline{h_k} \vee h_j h_k$
6. $(h_i \vee h_j)(h_i \vee \overline{h_k})(h_j \vee h_k) \underset{o}{=} (h_i \vee \overline{h_k})(h_j \vee h_k)$

Elementarkonjunktion und -disjunktion

Elementarkonjunktion $\quad k_i(x) ::= \bigwedge_{r=0}^{n-1} (X_i(x_r) \sim x_r)$

Beispiel: Ermittlung von $k_3(x) = \overline{x_2} x_1 x_0$

BI	x_2	x_1	x_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
⋮	⋮	⋮	⋮

$$
\begin{aligned}
k_3(x) &::= (X_3(x_2) \sim x_2) \wedge (X_3(x_1) \sim x_1) \wedge (X_3(x_0) \sim x_0) \\
&::= (0 \sim x_2)(1 \sim x_1)(1 \sim x_0) \\
&::= (0x_2 \vee 1\overline{x_2})(1x_1 \vee 0\overline{x_1})(1x_0 \vee 0\overline{x_0}) \\
&::= (0 \vee \overline{x_2})(x_1 \vee 0)(x_0 \vee 0) \\
&::= \overline{x_2} x_1 x_0
\end{aligned}
$$

Elementardisjunktion $\quad d_i(x) ::= \bigvee_{r=0}^{n-1} (X_i(x_r) \not\sim x_r)$

Ermittlung **expliziter BOOLEscher Gleichungen in Normalform**
für je eine Ausgangsvariable $y_k \in y$ entsprechend folgender Definitionen:

KDNF
Kanonisch
Disjunktive
Normalform

$$y_k = \bigvee_{i=0}^{2^n-1} k_i(x) \wedge \lambda_k(X_i)$$

$$h^*(x) \underset{\text{o}}{=} \bigvee_{i \in M_k} k_i(x) \qquad \text{mit } \forall i (i \in M_k \leftrightarrow \lambda_k(X_i) = *)$$

KKNF
Kanonisch
Konjunktive
Normalform

$$y_k = \bigwedge_{i=0}^{2^n-1} (d_i(x) \vee \lambda_k(X_i))$$

$$\overline{h^*(x)} \underset{\text{o}}{=} \bigwedge_{i \in M_k} d_i(x) \qquad \text{mit } \forall i (i \in M_k \leftrightarrow \lambda_k(X_i) = *)$$

KNANF
Kanonische
NAND-
Normalform

$$KDNF \xleftrightarrow[und\ deMorgan]{doppelteNegation} KNANF$$

$$y_k = \overline{\bigwedge_{i=0}^{2^n-1} \overline{k_i(x) \wedge \lambda_k(X_i)}}$$

KNONF
Kanonische
NOR-
Normalform

$$KKNF \xleftrightarrow[und\ deMorgan]{doppelteNegation} KNONF$$

$$y_k = \overline{\bigvee_{i=0}^{2^n-1} \overline{d_i(x) \vee \lambda_k(X_i)}}$$

Ausgangspunkt: Wertetabelle

Idee:

- Grafische Aufstellung der Wertetabelle so, daß benachbarte Belegungen auch in der Tabelle benachbart sind.
  - Zwei Belegungen X_i und X_j heißen **benachbart**, wenn sie sich in genau einem Bit an der r-ten Stelle unterscheiden, d.h. es gilt:

$$X_i(x_s) = \begin{cases} \overline{X_j(x_r)} & \text{falls} \quad s = r \\ X_j(x_r) & \text{sonst} \end{cases} \qquad (1)$$

  - Elementarkonjunktionen benachbarter Belegungen sind in der Variablen x_r kürzbar zu Fundamental-Konjunktionen entsprechend folgender Kürzungsregeln:

$$h_i(x) \underset{\text{o}}{=} h_i(x)h_j(x) \vee h_i(x)\overline{h_j(x)}$$
$$h_i(x) \underset{\text{o}}{=} x_r h_i(x) \vee \overline{x_r} h_i(x)$$

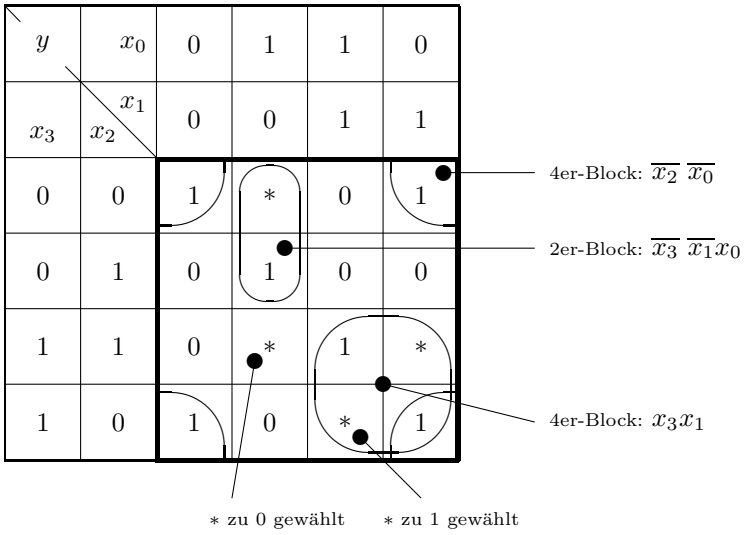
- Grafische Gruppenbildung benachbarter Belegungen

Verfahren:

- zwei im Karnaugh-Plan benachbarte Felder erfüllen die Nachbarschaftsbeziehung (1)
- linker und rechter sowie oberer und unterer Rand des Karnaugh-Planes sind benachbart
- $\Rightarrow$ Minimierung durch Bilden von 2er-, 4er-, 8er-, ... Blöcken untereinander benachbarter Felder
- die Variablen, deren Wert innerhalb eines Blockes konstant ist, bilden den (diese Belegungen repräsentierenden) Minimalausdruck

Beispiel

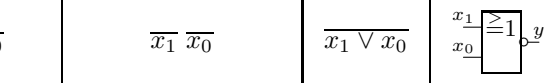
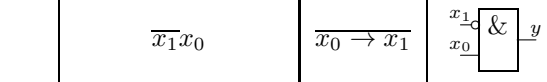
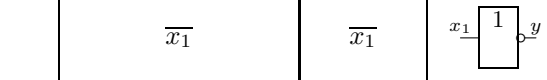
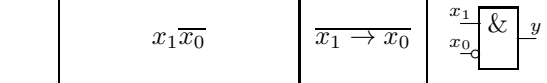
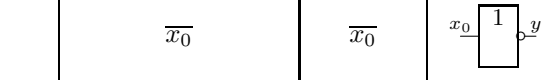
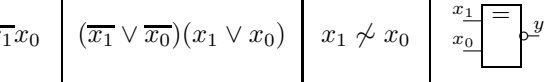
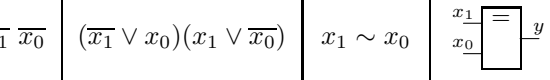
x_3	x_2	x_1	x_0	y
0	0	0	0	1
0	0	0	1	*
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	*
1	1	0	0	0
1	1	0	1	*
1	1	1	0	*
1	1	1	1	1



$y_{min} \underset{\text{o}}{=} \overline{x_2}\ \overline{x_0} \vee x_3x_2x_1x_0 \vee \overline{x_3}x_2\overline{x_1}x_0$ $\quad h^* = \overline{x_3}\ \overline{x_2}\ \overline{x_1}x_0 \vee x_3\overline{x_2}x_1x_0 \vee x_3x_2\overline{x_1}x_0 \vee x_3x_2x_1\overline{x_0}$

$y_{min} \underset{*}{=} \overline{x_2}\ \overline{x_0} \vee x_3x_1 \vee \overline{x_3}\ \overline{x_1}x_0$ $\quad h^\bullet = \overline{x_3}\ \overline{x_2}\ \overline{x_1}x_0 \vee x_3\overline{x_2}x_1x_0 \vee x_3x_2x_1\overline{x_0}$

Beispiele für Repräsentanten h_i der Mengen H^i für Ausdrücke mit 2 Variablen x_0, x_1 in unterschiedlichen Normalformen

x_1 x_0	1 1 0 0 1 0 1 0	Funktionsname	DNF	KNF	weitere NF	Schaltsymbol
y_0	0 0 0 0	Null	0	0	0	
y_1	0 0 0 1	NOR (not or)	$\overline{x_1}\ \overline{x_0}$	$\overline{x_1}\ \overline{x_0}$	$\overline{x_1 \vee x_0}$	
y_2	0 0 1 0	Inhibition von x_0 auf x_1	$\overline{x_1}x_0$	$\overline{x_1}x_0$	$\overline{x_0 \to x_1}$	
y_3	0 0 1 1	NOT (Negation von x_1)	$\overline{x_1}$	$\overline{x_1}$	$\overline{x_1}$	
y_4	0 1 0 0	Inhibition von x_1 auf x_0	$x_1\overline{x_0}$	$x_1\overline{x_0}$	$\overline{x_1 \to x_0}$	
y_5	0 1 0 1	NOT (Negation von x_0)	$\overline{x_0}$	$\overline{x_0}$	$\overline{x_0}$	
y_6	0 1 1 0	Antivalenz (XOR,Exklusiv-Oder)	$x_1\overline{x_0} \vee \overline{x_1}x_0$	$(\overline{x_1} \vee \overline{x_0})(x_1 \vee x_0)$	$x_1 \not\sim x_0$	
y_7	0 1 1 1	NAND (not and)	$\overline{x_1} \vee \overline{x_0}$	$\overline{x_1} \vee \overline{x_0}$	$\overline{x_1 x_0}$	
y_8	1 0 0 0	AND (Konjunktion, Und)	x_1x_0	x_1x_0	x_1x_0	
y_9	1 0 0 1	Äquivalenz	$x_1x_0 \vee \overline{x_1}\ \overline{x_0}$	$(\overline{x_1} \vee x_0)(x_1 \vee \overline{x_0})$	$x_1 \sim x_0$	
y_{10}	1 0 1 0	Identität von x_0	x_0	x_0	x_0	
y_{11}	1 0 1 1	Implikation von x_1 auf x_0	$\overline{x_1} \vee x_0$	$\overline{x_1} \vee x_0$	$x_1 \to x_0$	
y_{12}	1 1 0 0	Identität von x_1	x_1	x_1	x_1	
y_{13}	1 1 0 1	Implikation von x_0 auf x_1	$x_1 \vee \overline{x_0}$	$x_1 \vee \overline{x_0}$	$x_0 \to x_1$	
y_{14}	1 1 1 0	OR (Disjunktion, Oder)	$x_1 \vee x_0$	$x_1 \vee x_0$	$x_1 \vee x_0$	
y_{15}	1 1 1 1	Eins	1	1	1	

Struktur $S = [M, x, p, y, \kappa]$

mit:

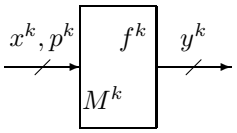
- M Menge von Modulen $M^k = [x^k, p^k, y^k, f^k] \in M$
- x Eingangsvektor; x^k ... Eingangsvektor des Moduls M^k
- p Programmiervektor; p^k ... Programmiervektor des Moduls M^k
- y Ausgangsvektor; y^k ... Ausgangsvektor des Moduls M^k
- κ Koppelfunktion $\bigcup_k x^k \cup \bigcup_k p^k \cup y \Rightarrow \bigcup_k y^k \cup x \cup p$

Koppelfunktion Die Koppelfunktion κ beschreibt die eindeutige Abbildung der Eingänge und Programmiereingänge der Module und der Strukturausgänge auf Modulausgänge sowie auf Eingänge oder Programmiereingänge der Struktur.

Schnittstelle Eingangsvektor x, Programmiervektor p und Ausgangsvektor y bilden die Schnittstelle der Struktur.

Kombinatorische Strukturen bestehen aus kombinatorischen Modulen und enthalten keine Rückführungen von Ausgängen auf Eingänge.

Modul M^k



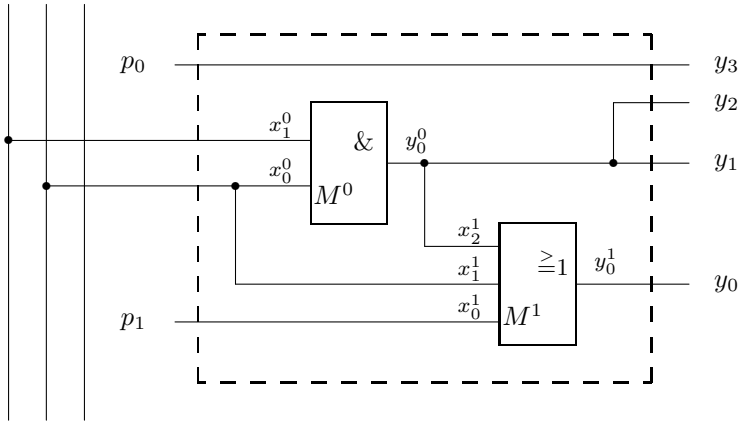
M^k ... Modulbezeichner
x^k ... Eingangsvektor des Moduls M^k
p^k ... Programmiervektor des Moduls M^k
y^k ... Ausgangsvektor des Moduls M^k
f^k ... Funktionsbezeichner des Moduls M^k

Beispiel: $x = [x_2, x_1, x_0],\ y = [y_3, y_2, y_1, y_0],\ p = [p_1, p_0],\ M = \{M^0, M^1\}$

κ : $\{x_1^0, x_0^0\} \cup \{x_2^1, x_1^1, x_0^1\} \cup \{y_3, y_2, y_1, y_0\} \Rightarrow \{y_0^0\} \cup \{y_0^1\} \cup \{x_2, x_1, x_0\} \cup \{p_1, p_0\}$

$\kappa(x_1^0) = x_2$ sprich: *"Die Eingangsvariable x_1 des Moduls M^0 ist gekoppelt mit der Eingangsvariablen x_2 des Eingangsvektors x"*

$\kappa(x_0^0) = x_1$
$\kappa(x_2^1) = y_0^0$
$\kappa(x_1^1) = x_1$
$\kappa(x_0^1) = p_1$
$\kappa(y_2) = y_0^0$
$\kappa(y_1) = y_0^0$
$\kappa(y_0) = y_0^1$
$\kappa(y_3) = p_0$

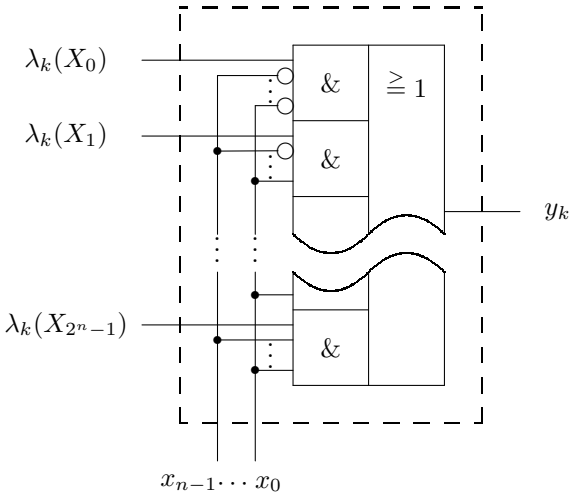


Beispiele für kombinatorische Strukturen

(1) Schaltsymbole auf Seite 11

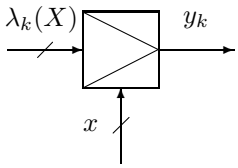
(2) Multiplexer

interne Struktur

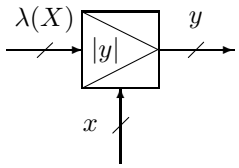


interne Struktur des Multiplexers k

Symbol



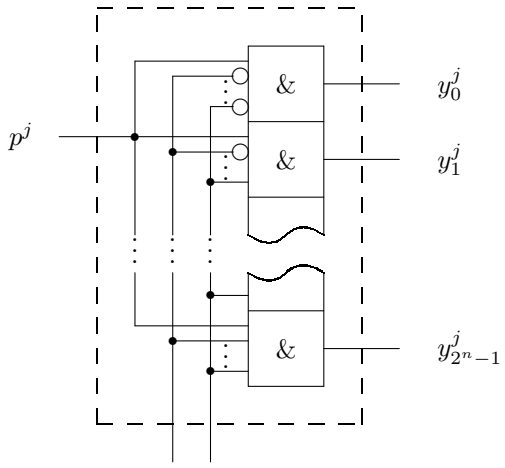
Multiplexer k



Multiplexerblock
aus $|y|$ Multiplexern

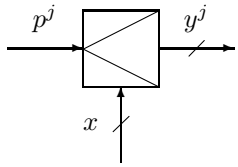
(3) Demultiplexer

interne Struktur

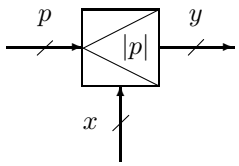


interne Struktur des Demultiplexers j

Symbol

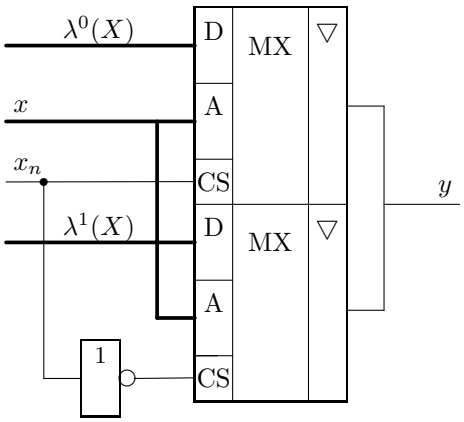


Demultiplexer j



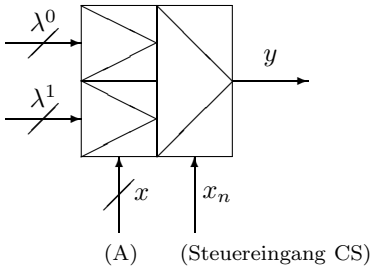
Demultiplexerblock
aus $|p|$ Demultiplexern

(4) Multiplexer-Kaskade



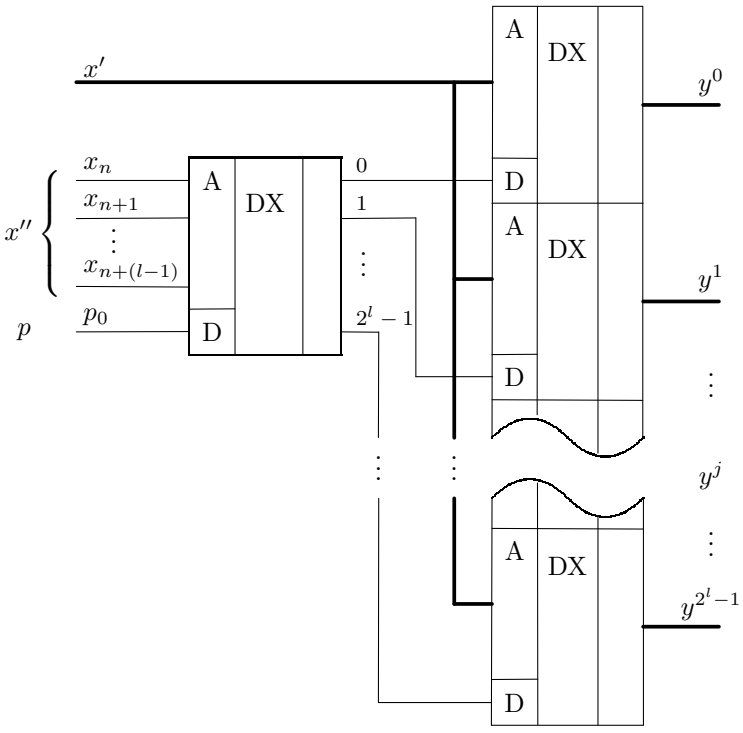
▽ ... Tri-state-Ausgänge

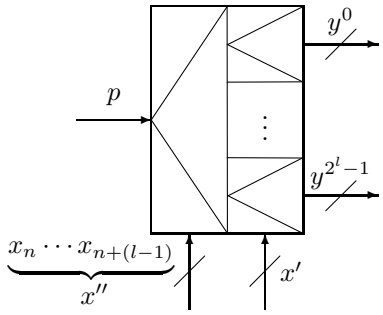
CS ... Chip Select



ohne Tri-state-Ausgänge

(5) Demultiplexer-Kaskade





$x = [x'', x']$

Beispiel mit $n = 2$ **und** $l = 3$

$|y^j| = 2^n = 4$ Elemente

$y^0 = [y_3, y_2, y_1, y_0]$

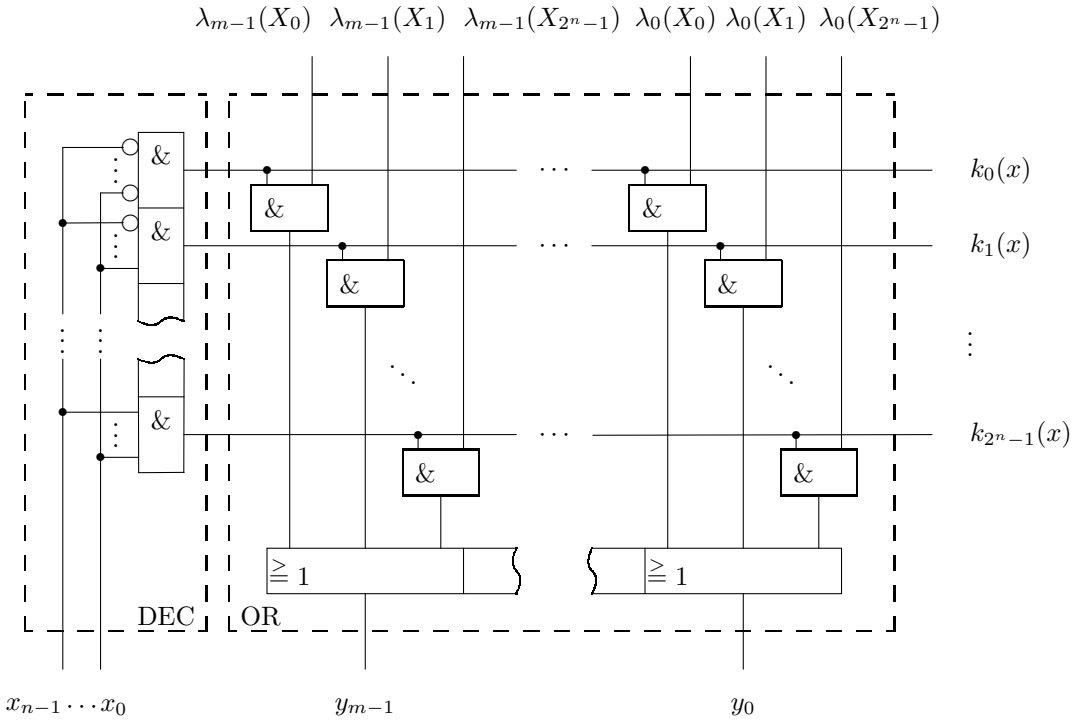
$\vdots$

$y^7 = [y_{31}, y_{30}, y_{29}, y_{28}] = y^{2^l-1}$

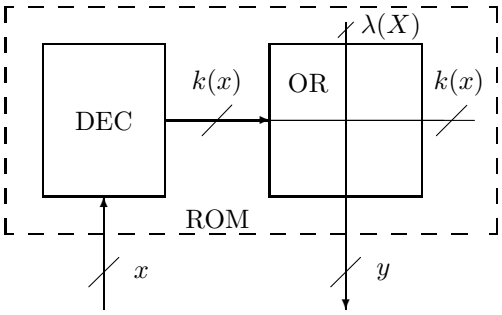
$x' = [x_1, x_0]$ $\quad x'' = [x_4, x_3, x_2]$

$p = [p_0]$

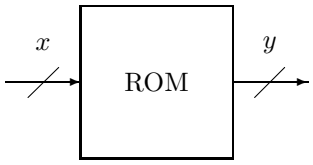
(6) ROM-Struktur



kombinatorische Ersatzstruktur

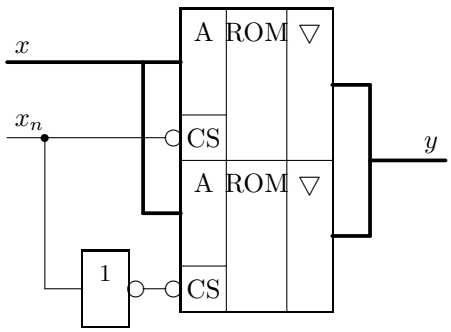


Blockstruktur

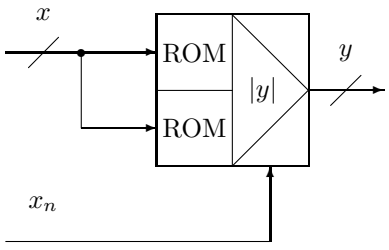


Blocksymbol

(7) ROM-Kaskade



▽ . . . Tri-state-Ausgänge

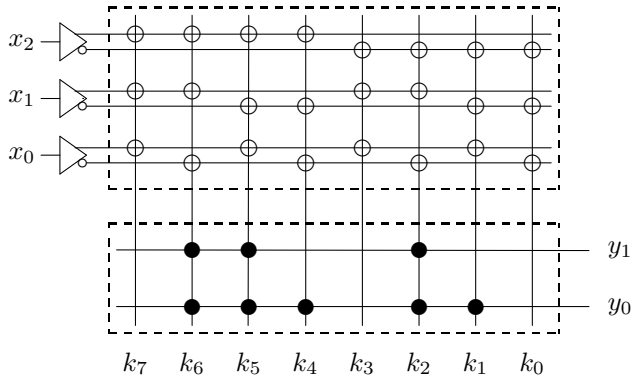


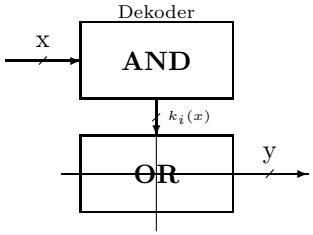
ohne Tri-state-Ausgänge

Programmierbare Strukturen (Hinweis: AND- und OR-Matritzen können auch als NOR-Matritzen realisiert sein)

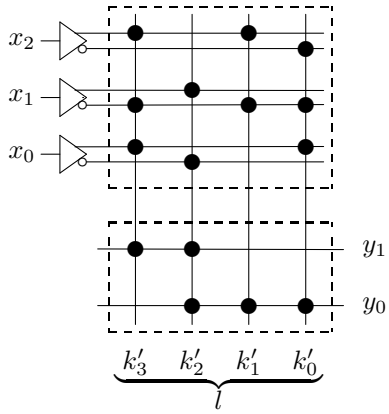
Beispiel: $y_1 = x_2\bar{x_1}x_0 \vee x_1\bar{x_0} \quad y_0 = x_2\bar{x_1} \vee \bar{x_2}\bar{x_1}x_0 \vee x_1\bar{x_0}$

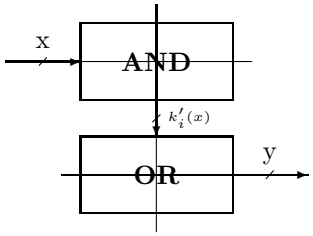
ROM Ausgangspunkt: Elementarkonjunktionen $k_i(x)$ bzw. KDNF



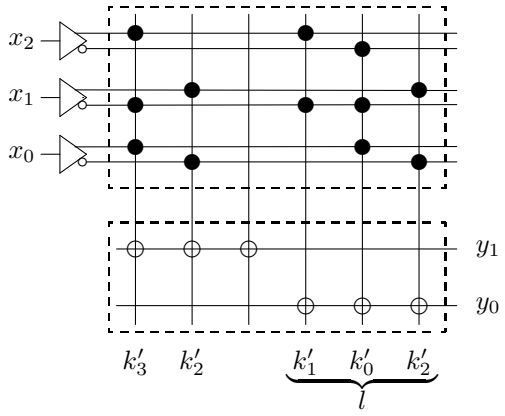


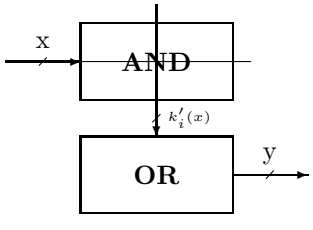
PLA Ausgangspunkt: Gleichungen in DNF mit max. l unterschiedlichen Fundamentalkonjunktionen für alle Gleichungen

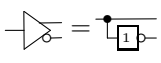




PAL/GAL Ausgangspunkt: Gleichungen in DNF mit max. l Fundamentalkonjunktionen je Gleichung







nicht verbunden fest verbunden programmiert

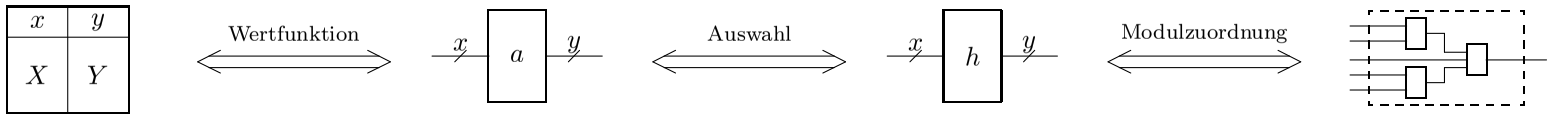
TU Ilmenau
Institut TII
FG IKS

Zusammenfassung Beschreibungsmittel

Techn. Informatik
Wuttke / Henke

Funktion ⟷ (**Synthese** / **Analyse**) Schaltung

BMA	**BAA**	**BAA**	**Struktur**
$\lambda:\ X(x) \Rightarrow Y(y)$	$y = a(x, g)$ $h^*(x)$	$y \underset{*}{=} h(x)$	$S = [M, x, p, y, æ]$



Wertetabelle	**Gleichungssystem**	**strukturorientiertes Gleichungssystem**	**Schaltungsstruktur**
mit $\lvert X\rvert + \lvert Y\rvert$ Spalten und $2^{\lvert x\rvert} = \lvert X\rvert$ Zeilen	aus $\lvert y\rvert$ Gleichungen und h^*	aus $\lvert y\rvert$ Gleichungen	**aus gekoppelten Moduln**

x_2	x_1	x_0	y_1	y_0
0	0	0	1	1
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	g_0
1	1	1	*	*

$y_0 = k_0 \vee k_1 \vee k_4 \vee g_0 k_6$

$y_1 = k_0 \vee k_3 \vee k_4 \vee k_5 \vee k_6$

$h^* = k_7$

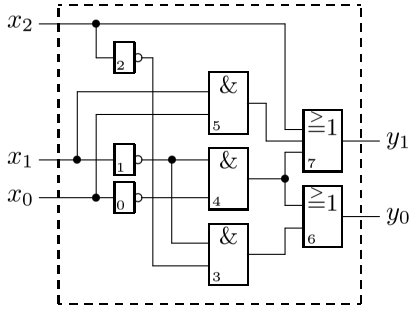
y_0	$\bar{x_1}\bar{x_0}$	$\bar{x_1}x_0$	x_1x_0	$x_1\bar{x_0}$
$\bar{x_2}$	1 k_0	1 k_1	0 k_3	0 k_2
x_2	1 k_4	0 k_5	* k_7	g_0 k_6

y_1	$\bar{x_1}\bar{x_0}$	$\bar{x_1}x_0$	x_1x_0	$x_1\bar{x_0}$
$\bar{x_2}$	1 k_0	0 k_1	1 k_3	0 k_2
x_2	1 k_4	1 k_5	* k_7	1 k_6

für $g_0 = 0$ ergibt sich:

$$y_{0_{min}} \underset{*}{=} \underbrace{\underbrace{\underbrace{\overline{x_1}\ \overline{x_0}}_{M^1} \vee \underbrace{\overline{x_2}\ \overline{x_1}}_{M^3}}_{M^4}}_{M^6}$$

$$y_{1_{min}} \underset{*}{=} \underbrace{x_2 \vee \underbrace{\overline{x_1}\ \overline{x_0}}_{M^4} \vee \underbrace{x_1 x_0}_{M^5}}_{M^7}$$



$y_0 \underset{o}{=} y_{0_{min}} \qquad y_1 \underset{o}{=} y_{1_{min}}$

$M = \{M_0, M_1, \ldots, M_7\}$

$x = [x_2, x_1, x_0]$

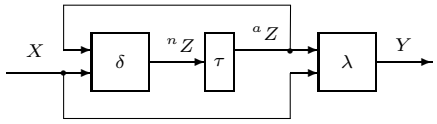
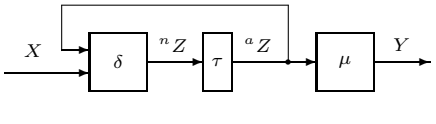
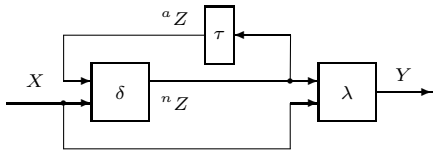
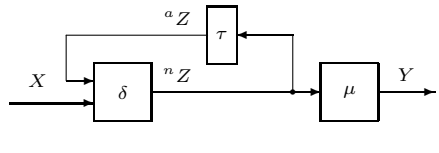
$p = \emptyset$

$y = [y_1, y_0]$

$æ(y_1) = y_0^7, \ldots, æ(x_0^0) = x_0$

Sequentielle Automaten

Automatentypen

	Mealy-Automat $A_\lambda = [X, Y, Z, \delta, \lambda]$	**Moore-Automat** $A_\mu = [X, Y, Z, \delta, \mu]$
Typ a	 $\lambda: {}^aZ \times X \Rightarrow Y$	 $\mu: {}^aZ \Rightarrow Y$
Typ n	 $\lambda: {}^nZ \times X \Rightarrow Y$	 $\mu: {}^nZ \Rightarrow Y$

X ... Eingangsbelegungsmenge — des Eingangsvektors $x = [x_{n-1}, \ldots, x_r, \ldots, x_0]$

Y ... Ausgangsbelegungsmenge — des Ausgangsvektors $y = [y_{m-1}, \ldots, y_k, \ldots, y_0]$

Z ... Zustandsbelegungsmenge — z.B. $Z_1(z) = [0, 0, \ldots, 0, 1] \in Z$ mit $z = [z_{p-1}, \ldots, z_s, \ldots, z_0]$ als p-stelliger Zustandsvektor z aus p binären Zustandsvariablen

^nZ ... neuer Zustand

^aZ ... alter Zustand

τ ... Verzögerung

δ ... Zustandsüberführungsfunktion $\quad \delta: Z \times X \Rightarrow Z$

λ/μ ... Mealy-/Moore-Ausgabefunktion

Automatentabelle für einen Mealy-Automaten vom Typ a

Belegungsindex					0	⋯	i	⋯	2^n-1
				x_0	0	⋯	$X_i(x_0)$	⋯	1
				⋮					
				x_r	0	⋯	$X_i(x_r)$	⋯	1
				⋮					
				x_{n-1}	0	⋯	$X_i(x_{n-1})$	⋯	1
	z_{p-1}	⋯	z_s	⋯	z_0				
0	0	⋯	0	⋯	0				
⋮									
j	$^aZ_j(z_{p-1})$	⋯	$^aZ_j(z_s)$	⋯	$^aZ_j(z_0)$	→	$^nZ_u := \delta(^aZ_j, X_i)$ $Y_t = \lambda(^aZ_j, X_i)$		
⋮									
2^p-1	1	⋯	1	⋯	1				

Zustands- und Automatengraphen

Zustandsgraph $G_\delta = [Z, K, \omega_\delta]$

Mealy-Automatengraph $G_\lambda = [Z, K, \omega_\delta, \omega_\lambda]$

Moore-Automatengraph $G_\mu = [Z, K, \omega_\delta, \omega_\mu]$

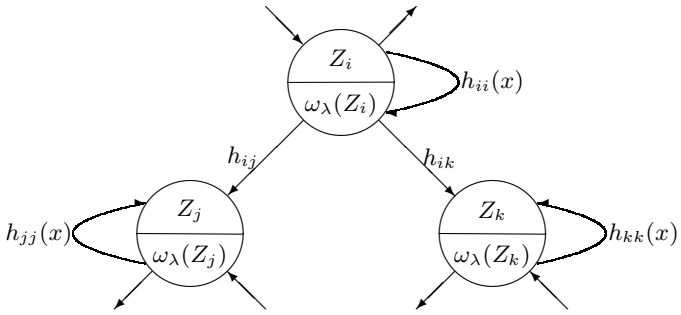
mit:

- Z ... Knotenmenge
- K ... Kantenmenge mit $K \subseteq Z \times Z$
- ω_δ ... Kantengewichtsfunktion (Zustandsüberführungsbedingung)
- ω_λ bzw. ω_μ ... Knotengewichtsfunktionen (Ausgabe)

wobei im einzelnen für Zustände und Kanten gilt:

- $[Z_i, Z_j] \in K \leftrightarrow \overline{h_{ij}(x) = 0}$ $(h_{ij} \neq 0)$
- $\omega_\delta([Z_i, Z_j]) = h_{ij}(x)$
  $h_{ij}(x)$ ... Übergangsausdruck (Kantengewicht der Kante $[Z_i, Z_j]$)
- $\omega_\lambda(Z_i) = \{y_k = h_{ik}(x) \quad | \quad 0 \leq k \leq m-1\}$ (Mealy)
- $\omega_\mu(Z_i) = \{y_k = h_{ik}(0,1) \quad | \quad 0 \leq k \leq m-1\}$ (Moore)
  $h_{ik}(x)$ ... Ausgabeausdruck der y_k-Komponente in Z_i, $m = |y|$

Allgemeine graphische Notationsform für G_λ



Vollständigkeit und Widerspruchsfreiheit

Vollständigkeit $\forall i(\bigvee_{j=0}^{2^p-1} h_{ij}(x) = 1)$

Widerspruchsfreiheit $\forall i(\bigvee_{\substack{j,k=0 \\ j \neq k}}^{2^p-1} h_{ij}(x) \wedge h_{ik}(x) = 0)$

Für die schaltalgebraische Realisierung von Automaten kann jeder Zustand Z_i durch eine Elementarkonjunktion von Zustandsvariablen $k_i(z)$ repräsentiert werden.

Aus dem Automatengraph lassen sich ableiten:

1. die Zustandsüberführungsfunktion δ

- **als Gleichungen für die Elementarkonjunktionen k_j der Zustände Z_j**

$$k_j(z) := \bigvee_{i=0}^{2^p-1} k_i(z) \wedge h_{ij}(x)$$

Anmerkung: h_{ij} ist ein Ausdruck in x-Variablen, der die Menge aller Eingangsbelegungen X_k repräsentiert, für die ein Zustandsübergang von Z_i nach Z_j erfolgt.

$$\mathcal{W}(h_{ij}, X_k) = 1 \Leftrightarrow \delta(Z_i, X_k) = Z_j$$

Sie werden zur Schaltungs*analyse* benutzt.

- **oder als Gleichungen für die Zustandsvariablen z_k**

$$z_k := \bigvee_{j \in M_k} \Big(\bigvee_{i=0}^{2^p-1} k_i(z) \wedge h_{ij}(x) \Big) \qquad \text{mit} \quad j \in M_k \leftrightarrow Z_j(z_k) = 1$$

Diese dienen als Ausgangspunkt der Schaltungs*synthese*.

2. die Ausgabefunktion λ (bzw. μ)

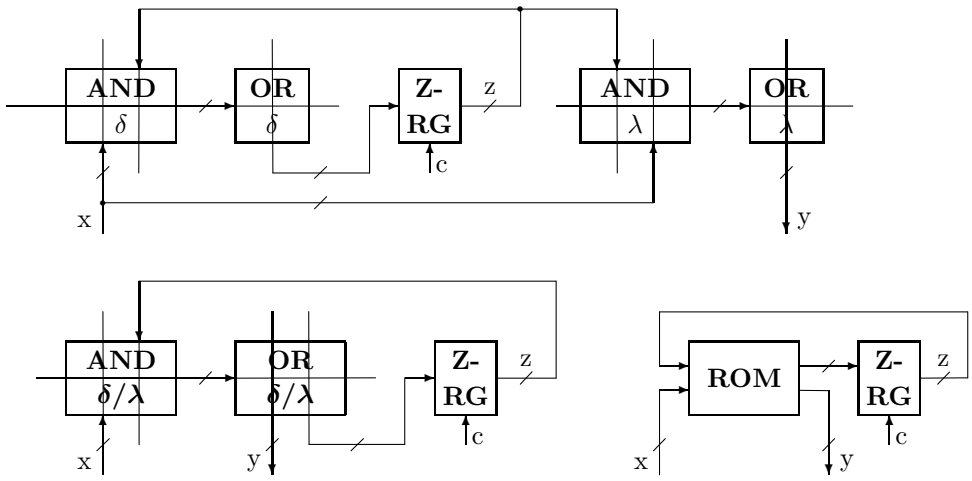
- **für Mealy-Automaten**

$$y_k = \bigvee_{i=0}^{2^p-1} k_i(z) \wedge h_{ik}(x)$$

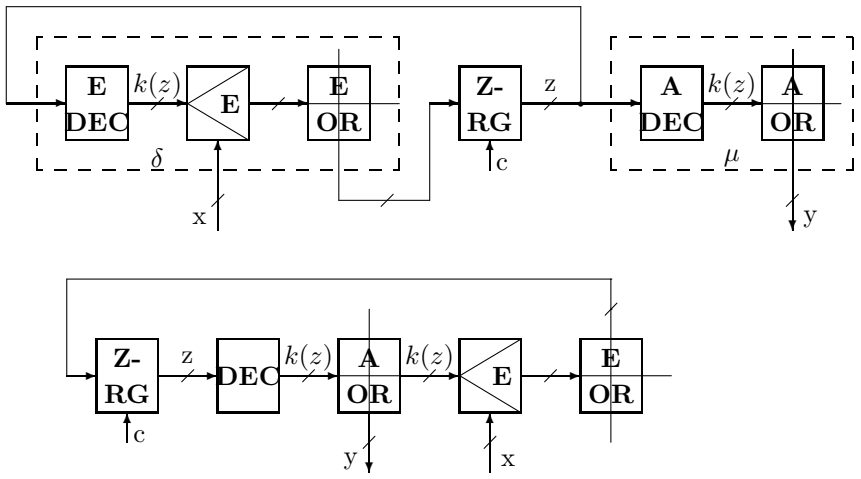
- **für Moore-Automaten**

$$y_k = \bigvee_{i=0}^{2^p-1} k_i(z) \wedge h_{ik}(0,1)$$

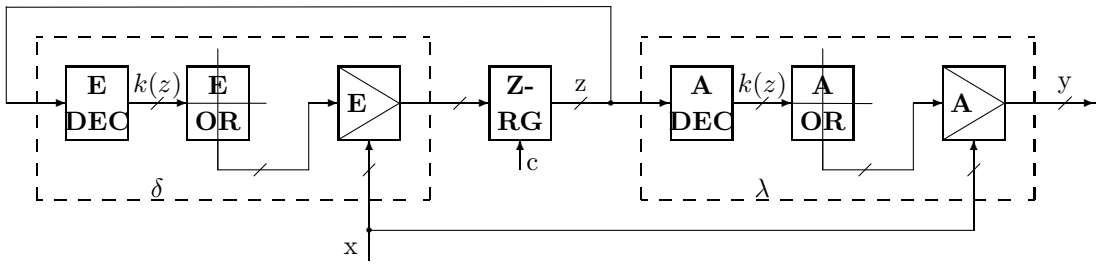
Automatenstruktur mit PLA/ROM



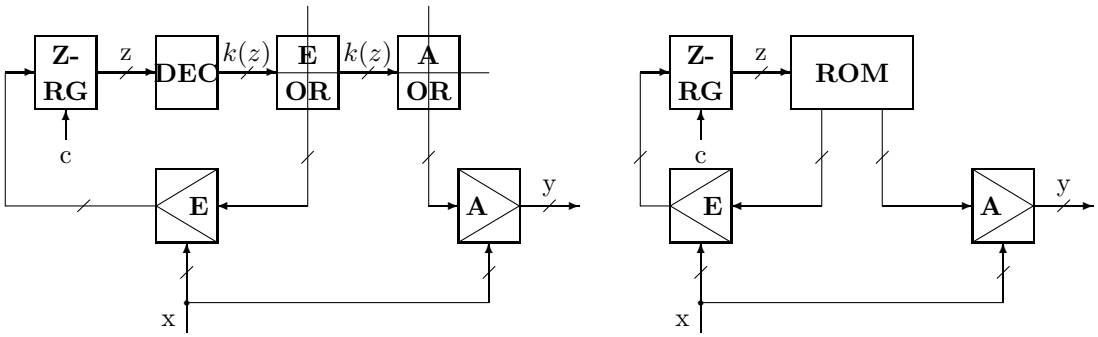
Vertikale Mikroprogrammsteuerung



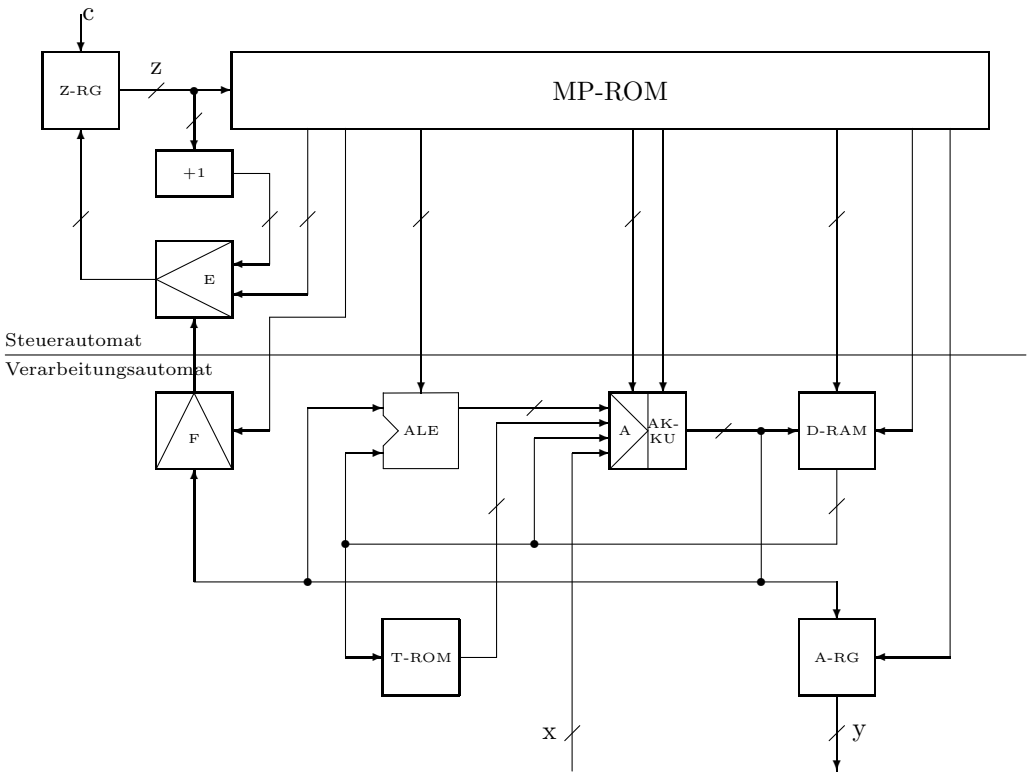
Horizontale Mikroprogrammsteuerung (Teil I)



Horizontale Mikroprogrammsteuerung (Teil II)



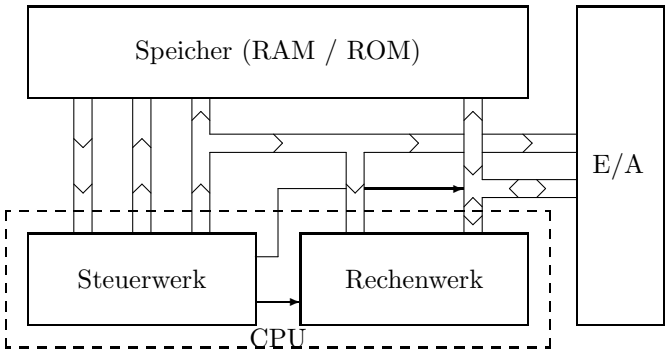
Einfache Digitale Maschine



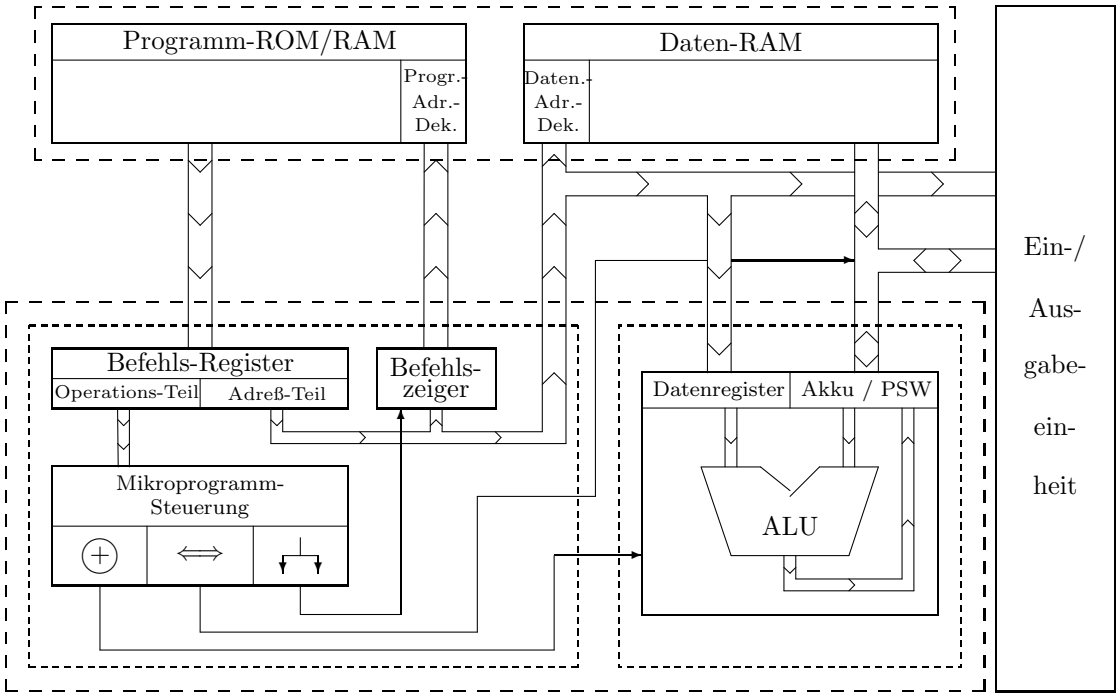
Z-RG	Zustandsregister mit dem Takt c
A-RG	Ausgangsregister
MP-ROM	Mikroprogramm-Speicher
+1	Zähler
AKKU	Akkumulator
A	Akku-Multiplexer
E	Eingangs-Multiplexer
F	Flag-Multiplexer
ALE	arithmetisch-logische Einheit
D-RAM	Daten-RAM
T-ROM	Text-ROM

Mikrorechner-Architektur

Allgemeine Struktur

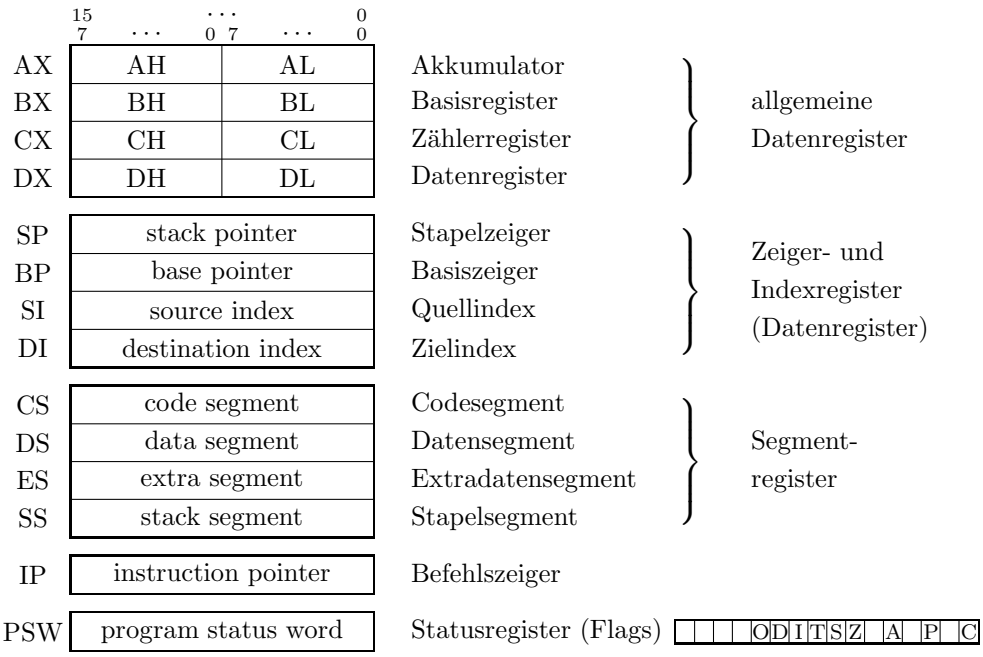


Detaillierte Struktur



Harvard-Architektur : Programm- und Datenspeicher getrennt
von-Neumann-Architektur : Programm- und Datenspeicher gemeinsam

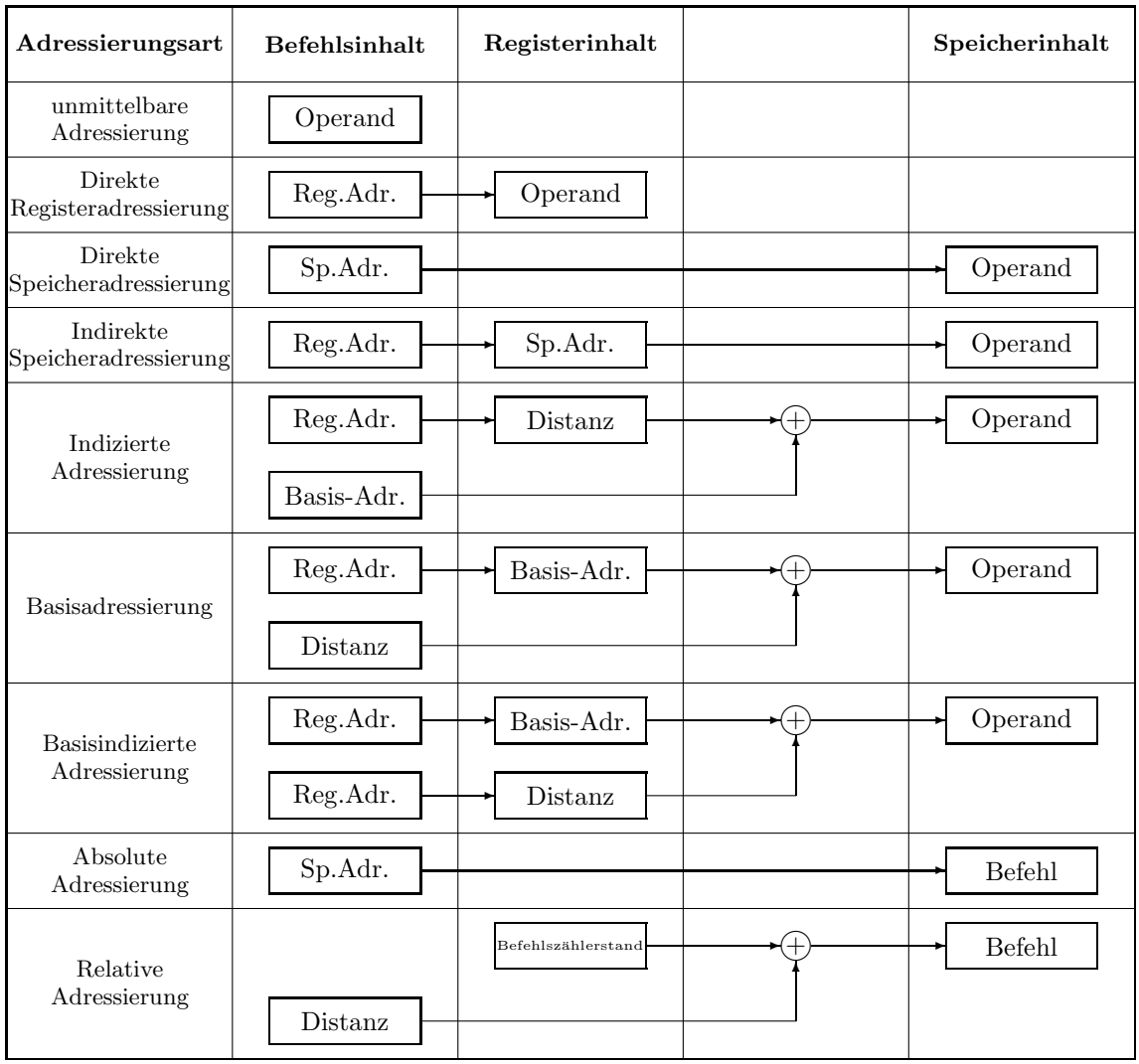
Der Registersatz des i8086



Bezeichnung	Kürzel	Struktur	vorrangige Verwendung
allgemeine Datenregister	AX	$2 \cdot 8$ bit	Akkumulator
	BX	$2 \cdot 8$ bit	Basisregister
	CX	$2 \cdot 8$ bit	Zählerregister
	DX	$2 \cdot 8$ bit	Datenregister
Zeigerregister	SP	16 bit	Zeiger auf den Stapelanfang
	BP	16 bit	Zeiger auf den Anfang des Adreßbereiches
Indexregister	SI	16 bit	Quelladresse bei Stringoperationen
	DI	16 bit	Zieladresse bei Stringoperationen
Segmentregister	CS	16 bit	Anfangsadresse des Programmbereiches
	DS	16 bit	Anfangsadresse des Datenbereiches
	ES	16 bit	Anfangsadresse des Extradatenbereiches
	SS	16 bit	Anfangsadresse des Stapelbereiches
Befehlszeiger	IP	16 bit	Zeiger auf den nächsten auszuführenden Befehl
Overflow-Flag	O	1 bit	Überlauf
Direction-Flag	D	1 bit	Zählrichtung
Interrupt-Flag	I	1 bit	Interruptfreigabe
Trap-Flag	T	1 bit	Einzelschritt für Testzwecke
Sign-Flag	S	1 bit	MSB (höchstwertiges Bit)
Zero-Flag	Z	1 bit	Null
Auxiliary Carry-Flag	A	1 bit	Halbbyteübertrag
Parity-Flag	P	1 bit	Parität des Low-Bytes
Carry-Flag	C	1 bit	Übertrag
Speicher	RAM	1 Mbyte	Daten- und Programmspeicher
Ein-/Ausgabe	I/O	$64K \cdot 8$bit	Peripherieanschlüsse

Adressierungsarten

Während der Programmabarbeitung sind Operanden für Verknüpfungsoperationen bereitzustellen. Je nach den vom Algorithmus geforderten Zugriffsarten auf die Operanden ist es für eine effektive Programmabarbeitung günstig, verschiedene Zugriffsmechanismen auf die Operanden zu haben, um die vorhandenen Strukurelemente (Register, Speicher) effizient adressieren zu können. Die unterschiedlichen Zugriffsmechanismen auf die Inhalte der Strukturelemente werden als **Adressierungsarten** bezeichnet.
Typisch für Mikrorechner sind die im folgenden dargestellten Adressierungsarten.



Reg.Adr.. . . Register-Adresse Sp.Adr. . . . Speicher-Adresse

Praktisch werden die genannten Adressierungsarten auch kombiniert verwendet. In den folgenden Beispielen wird die Adressierungsart des jeweils zweiten Operanden erläutert.
Als Syntaxschreibweise wurde die des Turbo-Assemblers bzw. des Turbo-Debuggers (/Turbo-Debugger/) gewählt.

Unmittelbare Adressierung

Der Operand befindet sich unmittelbar im Adreßteil des Befehls.
Bei der Programmierung findet diese Adressierungsart vorwiegend zur Festlegung von Anfangswerten (Initialisierung) sowie zur schnellen Bereitstellung konstanter Operanden Anwendung.

Beispiele:

- MOV AL,2FH
- ADD BX,2512H

Direkte Registeradressierung

Der Adreßteil beinhaltet die Adresse eines Universalregisters, das den Operanden direkt enthält.
Diese Adressierungsart ermöglicht eine im Vergleich zur Speicheradressierung wesentlich schnellere Operandenbereitstellung.

Beispiele:

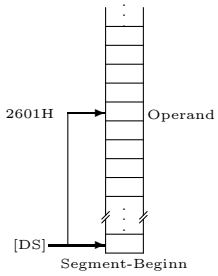
- ADD AL,DL
- XCHG BX,DX

Direkte Speicheradressierung

Die Adresse des Operanden steht direkt im Adreßteil.
Diese Adressierungsart wird benutzt, wenn Operanden an feste Speicheradressen gebunden sind, die sich im Programm nicht ändern.

Beispiel:

- MOV AX,[2601H]

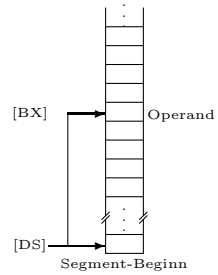


Indirekte Speicheradressierung

Durch den Adreßteil wird ein Universalregister bestimmt, das indirekt über seinen Inhalt den Operanden im Speicher adressiert.
Damit besteht die Möglichkeit, durch Veränderung des Registerinhalts unterschiedliche Operanden unter Nutzung des gleichen Programms zu adressieren. Die im Vergleich zur Speicheradresse kürzere Registeradresse führt zu kürzeren Befehlswörtern.

Beispiel:

- SUB AH,[BX]

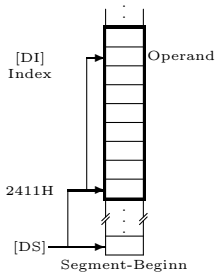


Indizierte Adressierung

Der Adreßteil enthält eine Basisadresse und eine Indexregister-Adresse. Die Basisadresse zeigt auf den Anfang eines interessierenden Speicherbereichs mit homogener Struktur. Im Indexregister befindet sich eine (positive oder negative) Distanzangabe, die den Abstand relativ zum Anfang des ausgewählten Speicherbereichs angibt.
Diese Adressierungsart ist vorteilhaft zur Adressierung von Vektoren. Die Basisadresse zeigt auf den Vektoranfang. Um auf bestimmte Elemente innerhalb des Vektors zugreifen zu können, ist der Registerinhalt entsprechend zu verändern.

Beispiel:

- ADD AX,[DI+2411H]

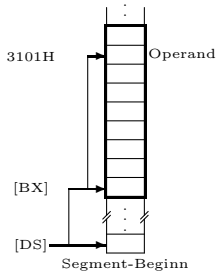


Basisadressierung

Es werden hierbei eine Registeradresse und eine Distanzangabe im Adreßteil geführt. Der Inhalt des adressierten Registers enthält die Basisadresse, d.h. den Anfang eines interessierenden Speicherbereichs, der auch eine inhomogene Struktur aufweisen kann. Die Distanzangabe beschreibt den relativen Abstand zur Basisadresse.

Beispiel:

- MOV AL,[BX+3101H]



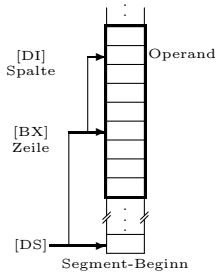
Basisindizierte Adressierung

Sie stellt eine Kombination aus indizierter und Basisadressierung dar, d.h durch den Adreßteil werden zwei Register adressiert, wobei das eine die Basisadresse und das andere die Distanzangabe enthält.

Durch Änderung der Registerelemente ist ein wahlfreier Zugriff auf beliebige Tabellenemente möglich.

Beispiel:

- CMP AX,[BX][DI]

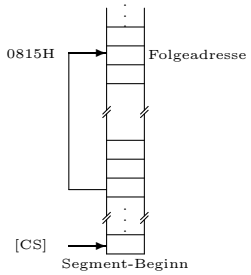


Absolute Befehls-Adressierung

Der Adreßteil des Befehls enthält eine Sprungadresse, die unverändert (absolut) in den Befehlszähler übernommen wird und damit einen Sprung in der Programmfortsetzung bewirkt.

Beispiel:

- JMP 0815H

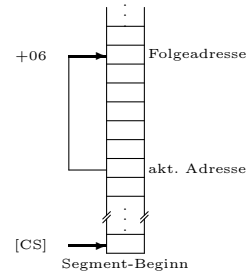


Relative Befehls-Adressierung

Relativ in bezug auf den Inhalt des Befehlszählers befindet sich im Adreßteil eine Distanzangabe. Die Summe aus Distanz und aktuellem Befehlszählerstand ergibt den in den Befehlszähler zu übernehmenden Wert.
Diese Adressierung wird in Sprungbefehlen verwendet, die eine Fortsetzung des Programms unabhängig von seiner Bindung an einen Speicherplatz gewährleisten sollen.

Beispiel:

- JNZ 06



Informationskodierung $\mathcal{K} : Vb \Longleftrightarrow Nb$

Vorbereich (Vb)	Befehle, Daten
Nachbereich (Nb)	Menge von n-Tupeln binärer Werte, wobei $N_n = \{0,1\}^n$ ... Menge aller n-Tupel
n-stelliger Kode	k heißt n-stelliger Kode von z, falls $z \in$ Vb, $k \in N_n$ und $\mathcal{K}(z) = k$
Kodiervorschrift KV	Eine Kodiervorschrift besteht aus Angaben zur Struktur sowie einem Algorithmus für die Kode-Erzeugung.

I. Zeichen-Kodierung

ASCII-Kodierung

- Vb: α-numerische Zeichen (in der Tabelle: Spalte α-n)
- Nb: N_8 ... Menge von Bytes (dargestellt als Hexadezimalzahlen $(ASC)_{16}$ bzw. Dezimalzahlen $(ASC)_{10}$)
- Kodiervorschrift: siehe nachfolgende Tabelle

$(ASC)_{16}$	$(ASC)_{10}$	α-n	$(ASC)_{16}$	$(ASC)_{10}$	α-n	$(ASC)_{16}$	$(ASC)_{10}$	α-n	$(ASC)_{16}$	$(ASC)_{10}$	α-n
00	00	NUL	20	32	SP	40	64	@	60	96	‘
01	01	SOH	21	33	!	41	65	A	61	97	a
02	02	STX	22	34	”	42	66	B	62	98	b
03	03	ETX	23	35	#	43	67	C	63	99	c
04	04	EOT	24	36	$	44	68	D	64	100	d
05	05	ENQ	25	37	%	45	69	E	65	101	e
06	06	ACK	26	38	&	46	70	F	66	102	f
07	07	BEL	27	39	’	47	71	G	67	103	g
08	08	BS	28	40	(	48	72	H	68	104	h
09	09	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

II. Zahlen-Kodierung

(1) BCD-Zahlen

- Vb: Z ... Menge der Dezimalziffern z_i mit $z_i \in \{0, 1, 2, \ldots, 9\}$
- Nb: $T = N_4 \backslash P$ mit:
  - T ... Menge der Tetraden t_j
  - P ... Menge der Pseudotetraden p
- KV: $\mathcal{K}(z_i) = t_j$ (Indexbestimmung: siehe Tabelle)

Ausgewählte BCD-Kodes

z_i (i ... Dezimalziffernwert)				t_j	
BCD (direkt)	**Aiken**	**3xS**	**Gray**	$2^3 2^2 2^1 2^0$	j
0	0	-	0	0 0 0 0	0
1	1	-	1	0 0 0 1	1
2	2	-	3	0 0 1 0	2
3	3	0	2	0 0 1 1	3
4	4	1	7	0 1 0 0	4
5	-	2	6	0 1 0 1	5
6	-	3	4	0 1 1 0	6
7	-	4	5	0 1 1 1	7
8	-	5	9	1 0 0 0	8
9	-	6	-	1 0 0 1	9
-	-	7	-	1 0 1 0	A
-	5	8	-	1 0 1 1	B
-	6	9	8	1 1 0 0	C
-	7	-	-	1 1 0 1	D
-	8	-	-	1 1 1 0	E
-	9	-	-	1 1 1 1	F
$j := i$	(a)	$j := i + 3$		$j \ldots Dualzahlenwert$	

(a): $j := \begin{cases} i & \text{falls} \quad 0 \le i \le 4 \\ i+6 & \text{falls} \quad 5 \le i \le 9 \end{cases}$

Operationen mit BCD-Zahlen

BCD-Zahlen werden bei Addition und Subtraktion entsprechend der Gesetze der Dualzahlen-Arithmetik verknüpft.
Da hierbei je Dezimalziffer der Zahlenbereich vierstelliger Dualzahlen genutzt wird, sind die Ergebnisse in Abhängigkeit von auftretenden **Pseudotetraden p** bzw. **Tetradenüberträgen ü** kodeabhängig mit Hilfe einer **Konstanten C** folgendermaßen zu korrigieren:

- bei Addition $\mathcal{K}(z_i + z_i') = \mathcal{K}(z_i) + \mathcal{K}(z_i') + \mathbf{C}$
- bei Subtraktion $\mathcal{K}(z_i - z_i') = \mathcal{K}(z_i) - \mathcal{K}(z_i') - \mathbf{C}$

	BCD	Aiken	3xS
Korrektur erforderlich	bei **ü** oder **p**	nur bei **p**, dann abhängig von **ü**	immer, aber abhängig von **ü**
Konstante C	0110	−0110 bei **ü** +0110 sonst	+0011 bei **ü** −0011 sonst

(2) Vorzeichen-Betrags-Zahlen

- Vb: $-2^{n-1} < z < 2^{n-1}$
- Nb: N_n
- KV: $MSB = 0 \quad \Leftrightarrow \quad z \geq 0$
  $MSB = 1 \quad \Leftrightarrow \quad z \leq 0$
  $Rest := |z_{n-1}|$

- Struktur:

MSB	$Betrag$
2^{n-1}	$2^{n-2} \quad \ldots \quad 2^0$

MSB ... Most Significant Bit
$Betrag$ ... $n-1$-stellige Darstellung der Dualzahl von $|z|$

(3) Konegative Zahlen

- Vb: $-2^{n-1} < z < 2^{n-1}$
- Nb: N_n (üblich: N_8, N_{16}, N_{32})
- KV:
  $\mathcal{K} = (z) = \begin{cases} z_n \leftrightarrow z \geq 0 \\ \overline{z_n} \text{ sonst} \end{cases}$
  z_n ... n-stellige Dualzahl von z
  $\overline{z_n}$ ... Komplement der Dualzahl

Komplementbildung

Man unterscheidet Einer- und Zweierkomplemente und dementsprechend als konegative Zahlen 1K- und 2K-Zahlen

- 1K-Zahlen: $\overline{z_n} = 2^n - 1 - z_n$
- 2K-Zahlen: $\overline{z_n} = 2^n - z_n$

Operationen mit Konegativen Zahlen

Für die Addition bzw. Subtraktion zweier Dualzahlen z_{n1} und z_{n2} mit

- $z_{n1} \geq z_{n2}$
- $s_n = z_{n1} + z_{n2}$ als Summe
- $d_n = z_{n1} - z_{n2}$ als Differenz

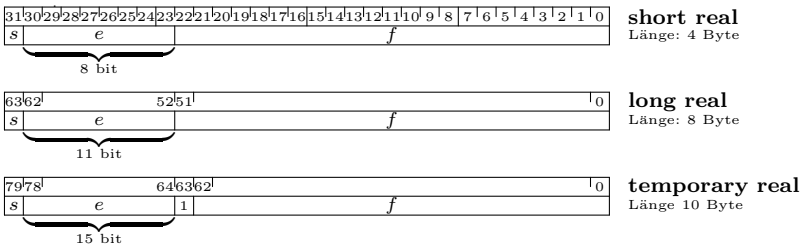
gilt bei Ersetzung der Komplemente entsprechend den Vorschriften zur Komplementbildung:

Operation	1k-Zahlen		2K-Zahlen	
	Ergebnis	Korrektur	Ergebnis	Korrektur
$z_{n1} + z_{n2}$	s_n	-	$z_{n1} + z_{n2} = s_n$	-
$z_{n1} + \overline{z_{n2}}$	$d_n + 2^n - 1$	$-2^n + 1$	$z_{n1} - z_{n2} + 2^n = d_n + 2^n$	-2^n
$\overline{z_{n1}} + z_{n2}$	$\overline{d_n}$	-	$-(z_{n1} - z_{n2}) + 2^n = \overline{d_n}$	-
$\overline{z_{n1}} + \overline{z_{n2}}$	$\overline{s_n} + 2^n - 1$	$-2^n + 1$	$-(z_{n1} + z_{n2}) + 2^n + 2^n = \overline{s_n} + 2^n$	-2^n

(4) Gleitkomma-Zahlen (IEEE-Standard)

- Vb: Menge rationaler Zahlen z
- Nb: N_{32} . . .short, N_{64} . . .long, N_{80} . . .temporary
- KV: s . . . sign (Vorzeichen der Zahl)
  e . . . biased exponent (vorzeichenloser Exponent)
  f . . . fractional part (gebrochener Anteil)

- Struktur:



Kodierung

1. Bestimmung von s $\quad s = \begin{cases} 1 & \Leftrightarrow z < 0 \\ 0 & \text{sonst} \end{cases}$
2. Umwandlung von z in eine Dualzahl $\quad |z| \rightarrow z_n$
3. normierte halblogarithmische Darstellung von $z_n = M \cdot (10)_2{}^E$
   mit: M . . . Mantisse, E . . . Exponent, so daß gilt: $1 \leq M < (10)_2$
4. Abspaltung von f aus $M = 1, f$
5. Berechnung von $e = E + (bias)_2 \quad (bias)_{16} = \begin{cases} 7F & \text{für short} \\ 3FF & \text{für long} \\ 3FFF & \text{für temporary} \end{cases}$
6. Formatanpassung gemäß der Struktur

Wertebereich

Datentyp	Bits	Wertebereich	Genauigkeit
short real	32	$\pm 1,2 * 10^{-38} \ldots \pm 3,4 * 10^{38}$	
long real	64	$\pm 2,2 * 10^{-308} \ldots \pm 1,8 * 10^{308}$	
temporary real	80	$\pm 1,1 * 10^{-4932} \ldots \pm 1,2 * 10^{4932}$	

spezielle Werte (für N_{32})

s	e	f	Wert
1/0	$= 00$	$= 0$	± 0
1/0	$= 00$	$\neq 0$	denormalisiert
1/0	$= FF$	$= 0$	$\pm \infty$
1/0	$= FF$	$\neq 0$	NaN (not a number)

Operationen

- getrennte Vorzeichen-, Exponenten- und Mantissenberechnung
- Exponentenanpassung bei Addition und Subtraktion
- Rundungen

III. Maschineninterne Datentypen

Kodes, die übereinstimmen in

- dem Vor- und Nachbereich ihrer Kodierung
- der Kodiervorschrift (Algorithmus und Struktur)
- den anwendbaren Operationen
- der semantischen Interpretation ihres Vorbereiches

können zu einem **Datentyp** zusammengefaßt werden.

Maschineninterne Datentypen werden vom Befehlssatz des Rechners direkt unterstützt, d.h. es existieren Befehle für ihre Kodierung und Verarbeitung.

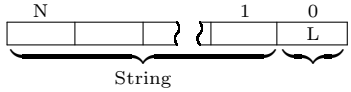
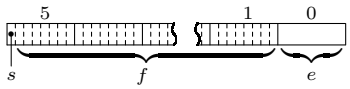
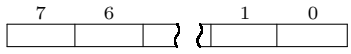
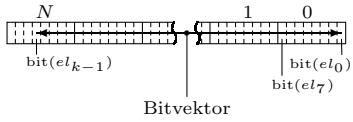
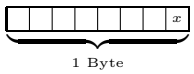
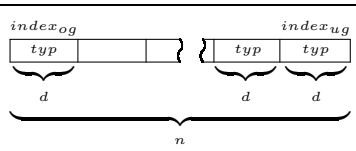
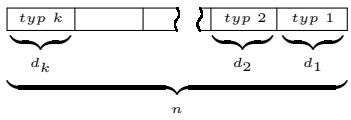
Bezeichnung	Vb (Interpretation)	Nb	KV	Operationen
bit	$\{true, false\}$	$\{0,1\}$	$true = 1$, $false = 0$	BOOLEsche Algebra
bitfield	Bitfolge	$\{0,1\}^n$, $(1 \leq n \leq 32)$	Struktur: n Bits	Vergleich
bcd	Dezimalziffern	$N_4 \backslash p$	siehe Arbeitsblatt	BCD-Arithmetik
ordinal	natürliche Zahlen	$N_8, N_{16}, N_{32}, N_{64}$	$z = z_n$	Dualzahlen-Arithmetik
integer	ganze Zahlen	$N_8, N_{16}, N_{32}, N_{64}$	siehe Arbeitsblatt	Arithm. mit koneg. Zahlen
real	reelle Zahlen	N_{32}, N_{64}, N_{80}	siehe Arbeitsblatt	Gleitkomma-Arithmetik
ASCII	α-numerische Zeichen	N_8	siehe Arbeitsblatt	Vergleich
string	Zeichenketten	N_8^n	byteweise ASCII	Verkettung, Vergleich
pointer	Adresse bzw. Offset	N_{16}, N_{32}	siehe Arbeitsblatt	Speicheradressierung

IV. Maschineninterne Darstellung ausgewählter Pascal-Datentypen

1. Auf maschineninterne Datentypen abbildbare Datentypen

Pascal-Typ	abgebildet auf Maschinentyp	Länge
BYTE	ordinal	1 byte
WORD	ordinal	2 byte
INTEGER	integer	2 byte
SHORTINT	integer	1 byte
LONGINT	integer	4 byte
SINGLE	short real	4 byte
DOUBLE	long real	8 byte
EXTENDED	temporary real	10 byte
CHAR	ASCII	1 byte
^datentyp	pointer	4 byte

2. Vom Laufzeitsystem zu verwaltende Pascal-Datentypen

Pascal-Typ	Struktur	Erläuterung
STRING[N]		L... tatsächliche Länge der Zeichenkette N... Länge gemäß Typvereinbarung
REAL		siehe unten
COMP		ganze Zahl $\pm 9,2 \cdot 10^{18}$ darstellbar
SET OF $el_0 \dots el_{k-1}$		$N = \text{int}(\frac{k-1}{8})$ $N \leq 31$ $k \leq 256$ k ... Anzahl der Elemente
BOOLEAN		$x = \begin{cases} 0 \dots false \\ 1 \dots true \end{cases}$
ARRAY $[index_{ug} \dots index_{og}]$ OF typ		$n = k \cdot d$ n ... benötigte Byteanzahl k ... Anzahl der Elemente d ... Byteanzahl von typ
RECORD $Komponente\ 1 : typ\ 1$; $Komponente\ 2 : typ\ 2$; ⋮ $Komponente\ k : typ\ k$ END;		$n = \sum_{i=1}^{k} d_i$ d_i ... Byteanzahl von $typ\ i$ n ... benötigte Byteanzahl

der Pascal-Typ REAL

- Vb: Menge rationaler Zahlen z
- Nb: N_{48}
- KV: s ... sign (Vorzeichen der Zahl)
  e ... biased exponent (vorzeichenloser Exponent)
  f ... fractional part (gebrochener Anteil)

Kodierung

1. Bestimmung von s $\quad s = \begin{cases} 1 & \Leftrightarrow z < 0 \\ 0 & \text{sonst} \end{cases}$
2. Umwandlung von z in eine Dualzahl $\quad |z| \rightarrow z_n$
3. normierte halblogarithmische Darstellung von $z_n = M \cdot (10)_2{}^E$
   mit: M ... Mantisse, E ... Exponent, so daß gilt: $1 \leq M < (10)_2$
4. Abspaltung von f aus $M = 1, f$
5. Berechnung von $e = E + (bias)_2 \quad (bias)_{16} = 81$
6. Formatanpassung gemäß der Struktur

Anhang

Mathematische Grundlagen

Aussagen

Definitionen

elementare Aussagen — sind Sätze zur Beschreibung von Eigenschaften (Prädikaten) p, q, ... von Individuen $x_0, x_1, \ldots$ aus einem bestimmten Individuenbereich $x = \{x_0, x_1, \ldots\}$.
z.B.: x_0 hat die Eigenschaft p.
Aussagen sind entweder $wahr(w)$ oder $falsch(f)$.

Aussagenvariable — $A, B, \ldots$ sind Symbole für Aussagen.

aussagenlogische Ausdrücke — sind folgendermaßen induktiv definiert:

1. die Aussagenvariablen $A, B, \ldots$ und die Wahrheitswerte w und f sind Ausdrücke,
2. wenn A und B Ausdrücke sind, so sind auch $\overline{A}, (A \wedge B)$, $(A \vee B), (A \rightarrow B)$ und $(A \leftrightarrow B)$ Ausdrücke,
3. nur die nach (1) und (2) gebildeten Zeichenketten sind Ausdrücke.

Elementare und zusammengesetzte Aussagen lassen sich mit Hilfe aussagenlogischer Ausdrücke beschreiben.

Wahrheitswerte zusammengesetzter Aussagen

A	B	Negation (A nicht) $\overline{A}$	Negation (B nicht) $\overline{B}$	Konjunktion (A und B) $A \wedge B$	Disjunktion (A oder B) $A \vee B$	Implikation (wenn A dann B) $A \rightarrow B$	Äquivalenz (A genau dann wenn B) $A \leftrightarrow B$
f	f	w	w	f	f	w	w
f	w	w	f	f	w	w	f
w	f	f	w	f	w	f	f
w	w	f	f	w	w	w	w

Priorität

Negation

Konjunktion

Disjunktion

alle weiteren

Wichtige *Äquivalenzen* aussagenlogischer Ausdrücke:

$A \to B$	äquivalent	$\overline{A} \vee B$
$A \leftrightarrow B$	äquivalent	$(A \wedge B) \vee (\overline{A} \wedge \overline{B})$
$\overline{A \wedge B}$	äquivalent	$\overline{A} \vee \overline{B}$
$\overline{A \vee B}$	äquivalent	$\overline{A} \wedge \overline{B}$

Kontradiktion ist ein aussagenlogischer Ausdruck, welcher nach beliebiger Wertzuweisung zu Aussagevariablen immer den Wert f hat.
Beispiele:

- $A \wedge \overline{A}$
- $A \wedge f$
- $(A \to B) \wedge (A \wedge \overline{B})$

Tautologie ist ein aussagenlogischer Ausdruck, welcher nach beliebiger Wertzuweisung zu Aussagenvariablen immer den Wert w hat.
Beispiele:

- $A \vee \overline{A}$
- $A \vee w$
- $(A \to B) \vee A$

$HORN - Klauseln$ sind aussagenlogische Ausdrücke der Form:

- $A_1 \wedge A_2 \wedge \ldots A_n \to B$ "Regel"
- $w \to B$ "Fakt"

Notizen

Prädikate

Prädikat — Teil einer Aussage, der eine klassifizierende Eigenschaft $(p, q, \ldots)$ beinhaltet.

abhängige Aussage — Der Wahrheitswert abhängiger Aussagen (z.B. $p(x), q(y), \ldots$) kann erst bestimmt werden, wenn die Individuensymbole (z.B. $x, y, \ldots$) durch konkrete Individuen eines Individuenbereiches ersetzt werden.

	abhängige Aussage	*Individuenbereich von x,y*
$p(x)$	x ist durch 25 teilbar	ganze Zahlen
$q(y)$	y ist ein Sommertag	alle Tage des Dezember
$r(x,y)$	x ist Hauptstadt von y	alle Städte > 1Mio. EW

wobei z.B. p das Symbol für das Prädikat *"ist durch 25 teilbar"* darstellt

Prädikatenlogische Ausdrücke und deren Sprechweise

Durch Einbeziehung der Individuensymbole in den Wirkungsbereich von Quantifikatoren (Allquantor $\forall$, *Existenzquantor* $\exists$) können prädikatenlogische Ausdrücke formuliert werden.

(a) $\exists x(r(x))$ — es existiert (mindestens) ein Element im Individuenbereich von x, für das gilt, das Prädikat r ist für x *wahr* (sprich: *"r von x ist wahr"*).

(b) $\overline{\forall x(\overline{r(x)})}$ — es gilt nicht, daß für alle Elemente aus dem Individuenbereich von x $r(x)$ nicht *wahr* ist – gleicher Sachverhalt wie unter (a);

(c) $\forall y(s(y)) = \overline{\exists y(\overline{s(y)})}$ — für alle Elemente des Individuenbereiches von y gilt, die Aussage $s(y)$ ist *wahr* bzw.: es gibt kein y, für das $s(y)$ nicht wahr ist;

(d) $\forall y \exists x(t(x,y))$ — für alle Elemente des Individuenbereiches von y existiert (mindestens) ein Element aus dem Individuenbereich von x, für welches gilt, daß $t(x,y)$ *wahr* ist;

(e) $\exists y \forall x(r(x) \rightarrow s(y))$ — es existiert (mindestens) ein Element im Individuenbereich von y, für das gilt, daß für alle Elemente des Individuenbereiches von x die Implikation $r(x) \rightarrow s(y)$ (sprich: *"aus r(x) folgt s(y)"*) *wahr* ist.

In allen gezeigten Beispielen für prädikatenlogische Ausdrücke kommen die Individuensymbole x und y *gebunden* vor, da sie im Wirkungsbereich des *Allquantors* $\forall$ bzw. des *Existenzquantors* $\exists$ stehen. Ansonsten sind die Individuensymbole *frei*.

Definitionen

Term — es gilt:

1. die Wahrheitswerte w und f und alle Individuensymbole $x, \ldots$ sind Terme;
2. wenn f ein n-stelliges Funktionssymbol und $t_1, \ldots, t_n$ Terme sind, so ist auch $f(t_1, \ldots, t_n)$ ein Term;
3. andere Terme existieren nicht.

Elementarausdruck — es gilt:

1. wenn p ein n-stelliges Prädikatensymbol und $t_1, \ldots, t_n$ Terme sind, so ist auch $p(t_1, \ldots, t_n)$ ein elementarer Ausdruck;
2. wenn t_1 und t_2 Terme sind, so ist auch $t_1 = t_2$ ein elementarer Ausdruck.

prädikatenlogischer Ausdruck — es gilt:

1. jeder Elementarausdruck ist Ausdruck;
2. sind A und B Ausdrücke, so sind auch $\overline{A}, A \wedge B, A \vee B, A \rightarrow B$ und $A \leftrightarrow B$ Ausdrücke;
3. wenn $A(x)$ Ausdruck und x ein in $A(x)$ vorkommendes Individuensymbol ist, wobei in $A(x)$ keine Symbolfolge der Art $\forall x$ oder $\exists x$ vorkommt, so sind auch $\forall x A(x)$ oder $\exists x A(x)$ Ausdrücke;
4. nur die nach (1) bis (3) gebildeten Zeichenreihen sind prädikatenlogische Ausdrücke.

Zusammenhang zwischen Aussagen und prädikatenlogischen Ausdrücken

Prädikatenlogische Ausdrücke, in denen nur gebundene Individuensymbole vorkommen, stellen Aussagen dar.
Im folgenden werden diese als Mittel zur Definition genutzt.

Mengen

Mengendefinition	$\forall a(a \in A \leftrightarrow p_A(a))$ sprich: Für alle a gilt: a ist Element der Menge A genau dann, wenn $p_A(a)$ gilt (d.h. die Aussage *"Für a gilt $p_A(a)$"* ist *wahr*)
	oder $A = \{a \mid p_A(a)\}$ oder $A = \{a_0, a_1, \ldots, a_{m-1}\}$
Mengenmächtigkeit	$\lvert A\rvert = m \leftrightarrow A$ enthält m Elemente
speziell	$\lvert\emptyset\rvert = 0$ $\quad\emptyset$: die leere Menge
disjunkte Menge	A disjunkt $B \leftrightarrow \overline{\exists a(a \in A \wedge a \in B)}$ $\quad A \cap B = \emptyset$
Teilmenge	$A \subseteq B \leftrightarrow \forall a(a \in A \rightarrow a \in B) \vee A = \emptyset$
echte Teilmenge	$A \subset B \leftrightarrow \forall a(a \in A \rightarrow a \in B) \wedge \exists a(\overline{a \in A} \wedge a \in B)$
Mengengleichheit	$A = B \leftrightarrow \forall a(a \in A \leftrightarrow a \in B)$
$n-Tupel$	$C = [c_{n-1}, c_{n-2}, \ldots, c_0]$ $\quad$ (geordnete Menge)
Mengenprodukt (*Kreuzprodukt*)	$\forall a, b([a, b] \in A \times B \leftrightarrow a \in A \wedge b \in B)$ $A \times B$ (sprich: *"A kreuz B"*)
Mengenpotenz	$A^m = A^{m-1} \times A$ $\quad$ mit: $\quad A^1 = A$
Potenzmenge	$\forall B(B \in \mathcal{P}(A) \leftrightarrow B \subseteq A)$
Mengenvereinigung	$\forall a(a \in A \cup B \leftrightarrow a \in A \vee a \in B)$
Mengenschnitt	$\forall a(a \in A \cap B \leftrightarrow a \in A \wedge a \in B)$
allgemein	$\bigcup_{i=1}^{m} A_i = \bigcup_{i=1}^{m-1} A_i \cup A_m$ $\quad$ bzw. $\quad \bigcap_{i=1}^{m} A_i = \bigcap_{i=1}^{m-1} A_i \cap A_m$
Mengendifferenz	$\forall a(a \in B \setminus A \leftrightarrow a \in B \wedge \overline{a \in A})$
Komplement	$\mathcal{K}_M(A) = M \setminus A$ $\quad$ Komplement der Menge A bzgl. Grundmenge M
kurz	$\mathcal{K}_M(A) = \bar{A}$ $\quad$ bei allgemein bekannter Grundmenge
Partition	Für $\Pi(A)$ gilt: 1. $\lvert\Pi(A)\rvert > 1$ 2. $\forall M, N(M, N \in \Pi(A) \wedge M \neq N \rightarrow \overline{M = \emptyset} \wedge \overline{N = \emptyset} \wedge M \cap N = \emptyset)$ 3. $\forall a(a \in A \rightarrow \exists M(M \in \Pi(A) \wedge a \in M))$

Relationen

$n-stellige\ Relation$	$\mathcal{R} \subseteq A^n, \quad \forall t(t \in \mathcal{R} \leftrightarrow t \in A^n \wedge r(t))$
$Infixnotation$	$a\ r\ b$ sprich *"a ist in Relation r zu b"*

Eigenschaften zweistelliger Relationen $r \in \mathcal{R}$ mit $\mathcal{R} \subseteq A^2$; $a, b, c \in A$

$Reflexivität$	$\forall a(a\ r\ a)$
$Irreflexivität$	$\forall a\overline{(a\ r\ a)}$
$Transitivität$	$\forall a, b, c(a\ r\ b \wedge b\ r\ c \rightarrow a\ r\ c)$
$Symmetrie$	$\forall a, b(a\ r\ b \rightarrow b\ r\ a)$
$Antisymmetrie$	$\forall a, b(a\ r\ b \wedge b\ r\ a \rightarrow a = b)$
$Asymmetrie$	$\forall a, b(a\ r\ b \rightarrow \overline{b\ r\ a})$
$Linearität$	$\forall a, b(a\ r\ b \vee b\ r\ a)$
$Konnexität$	$\forall a, b(a\ r\ b \vee a = b \vee b\ r\ a)$

Abbildungen

Abbildung $\mathcal{A}$	$\forall m, n([m,n] \in \mathcal{A} \leftrightarrow [m,n] \in M \times N \wedge p_A([m,n]))$
$Vorbereich(Vb)$	$\forall m(m \in Vb(\mathcal{A}) \leftrightarrow m \in M \wedge \exists n([m,n] \in \mathcal{A}))$
$Nachbereich(Nb)$	$\forall n(n \in Nb(\mathcal{A}) \leftrightarrow n \in N \wedge \exists m([m,n] \in \mathcal{A}))$
partielle Abbildung	$\mathcal{A}$ ist partiell $\leftrightarrow \exists m(m \in M \wedge \overline{m \in Vb(\mathcal{A})})$
$Funktion(\mathcal{F})$	eindeutige Abbildung, d.h. es gilt zusätzlich: $\forall m, n([m,n] \in \mathcal{F} \rightarrow \overline{\exists l(n \neq l \wedge [m,l] \in \mathcal{F})})$
symbolisch	$\mathcal{F} : M \Rightarrow N$ bzw. $n = \mathcal{F}(m)$ mit $n \in N$ und $m \in M$
$Operation(\mathcal{O})$	Funktion mit $Vb = M^n$ und $Nb = M$
symbolisch	$\mathcal{O} : M^n \Rightarrow M$ bzw. $m = \mathcal{O}[m_{n-1}, m_{n-2}, \ldots, m_0]$
$Kodierung(\mathcal{K})$	eindeutige Funktion, d.h. es gilt zusätzlich: $\forall m, n([m,n] \in \mathcal{K} \rightarrow \overline{\exists l(m \neq l \wedge [l,n] \in \mathcal{K})})$
symbolisch	$\mathcal{K} : M \Leftrightarrow N$ bzw. $n = \mathcal{K}(m)$ und $m = \mathcal{K}^{-1}(n)$
$Transformation(\mathcal{T})$	Kodierung mit $Vb = Nb = M$
symbolisch	$\mathcal{T} : M \Leftrightarrow M$

Literaturliste zur Lehrveranstaltung

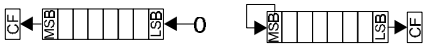
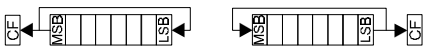
"Technische Informatik"

H.-D. Wuttke; K. Henke	Schaltsysteme – Eine automatenorientierte Einführung, Pearson-Education Deutschland, eBook, URL: http://ebooks.pearson-studium.de/schaltsysteme.html
Th. Flick	Mikroprozessortechnik und Rechnerstrukturen, Springer Verlag, Berlin 2005
H.-J. Zander	Logischer Entwurf binärer Systeme, Verlag Technik, Berlin 1992
S. Hentschke	Grundzüge der Digitaltechnik, Teubner-Verlag, Stuttgart 1988
Informatik-Duden	Duden-Verlag, Mannheim, Wien, Zürich 2002
H.-D. Wuttke; K. Henke	Arbeitsblätter zur Lehrveranstaltung "Rechnerorganisation", TU Ilmenau, Fakultät IA, Ilmenau 2020, http://www.tu-ilmenau.de/iks
H.-D. Wuttke; K. Henke	Online-Materialien zur Lehrveranstaltung "Rechnerorganisation", TU Ilmenau, Fakultät IA, Ilmenau 2020, http://www.tu-ilmenau.de/iks
moodle	Technische Informatik, Rechnerorganisation, Schaltsysteme Studienbegleitendes Online-Material, TU Ilmenau, Fakultät IA, Ilmenau 2020, https://moodle2.tu-ilmenau.de/course/view.php?id=1576
GOLDi	Grid of Online Lab Devices Ilmenau, Remote Lab des Fachgebietes IKS, TU Ilmenau, Fakultät IA, Ilmenau 2020, http://www.goldi-labs.net

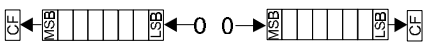
CodeTable 1/2

TRANSFER				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
MOV	Move (copy)	MOV Dest,Source	Dest:=Source									
XCHG	Exchange	XCHG Op1,Op2	Op1:=Op2 , Op2:=Op1									
STC	Set Carry	STC	CF:=1									1
CLC	Clear Carry	CLC	CF:=0									0
CMC	Complement Carry	CMC	CF:= ¬CF									±
STD	Set Direction	STD	DF:=1 (string op s downwards)		1							
CLD	Clear Direction	CLD	DF:=0 (string op s upwards)		0							
STI	Set Interrupt	STI	IF:=1			1						
CLI	Clear Interrupt	CLI	IF:=0			0						
PUSH	Push onto stack	PUSH Source	DEC SP, [SP]:=Source									
PUSHF	Push flags	PUSHF	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL									
PUSHA	Push all general registers	PUSHA	AX, CX, DX, BX, SP, BP, SI, DI									
POP	Pop from stack	POP Dest	Dest:=[SP], INC SP									
POPF	Pop flags	POPF	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL	±	±	±	±	±	±	±	±	±
POPA	Pop all general registers	POPA	DI, SI, BP, SP, BX, DX, CX, AX									
CBW	Convert byte to word	CBW	AX:=AL (signed)									
CWD	Convert word to double	CWD	DX:AX:=AX (signed)	±				±	±	±	±	±
CWDE	Conv word extendeddouble	CWDE 386	EAX:=AX (signed)									
IN *i*	Input	IN Dest, Port	AL/AX/EAX := byte/word/double of specified port									
OUT *i*	Output	OUT Port, Source	Byte/word/double of specified port := AL/AX/EAX									

i for more informations see instruction specifications Flags: ±=affected by this instruction ?=undefined after this instruction

ARITHMETIC				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
ADD	Add	ADD Dest,Source	Dest:=Dest+Source	±				±	±	±	±	±
ADC	Add with Carry	ADC Dest,Source	Dest:=Dest+Source+CF	±				±	±	±	±	±
SUB	Subtract	SUB Dest,Source	Dest:=Dest-Source	±				±	±	±	±	±
SBB	Subtract with borrow	SBB Dest,Source	Dest:=Dest-(Source+CF)	±				±	±	±	±	±
DIV	Divide (unsigned)	DIV Op	Op=byte: AL:=AX / Op AH:=Rest	?				?	?	?	?	?
DIV	Divide (unsigned)	DIV Op	Op=word: AX:=DX:AX / Op DX:=Rest	?				?	?	?	?	?
DIV 386	Divide (unsigned)	DIV Op	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	?				?	?	?	?	?
IDIV	Signed Integer Divide	IDIV Op	Op=byte: AL:=AX / Op AH:=Rest	?				?	?	?	?	?
IDIV	Signed Integer Divide	IDIV Op	Op=word: AX:=DX:AX / Op DX:=Rest	?				?	?	?	?	?
IDIV 386	Signed Integer Divide	IDIV Op	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	?				?	?	?	?	?
MUL	Multiply (unsigned)	MUL Op	Op=byte: AX:=AL*Op if AH=0 ♦	±				?	?	?	?	±
MUL	Multiply (unsigned)	MUL Op	Op=word: DX:AX:=AX*Op if DX=0 ♦	±				?	?	?	?	±
MUL 386	Multiply (unsigned)	MUL Op	Op=double: EDX:EAX:=EAX*Op if EDX=0 ♦	±				?	?	?	?	±
IMUL *i*	Signed Integer Multiply	IMUL Op	Op=byte: AX:=AL*Op if AL sufficient ♦	±				?	?	?	?	±
IMUL	Signed Integer Multiply	IMUL Op	Op=word: DX:AX:=AX*Op if AX sufficient ♦	±				?	?	?	?	±
IMUL 386	Signed Integer Multiply	IMUL Op	Op=double: EDX:EAX:=EAX*Op if EAX sufficient ♦	±				?	?	?	?	±
INC	Increment	INC Op	Op:=Op+1 (Carry not affected !)	±				±	±	±	±	
DEC	Decrement	DEC Op	Op:=Op-1 (Carry not affected !)	±				±	±	±	±	
CMP	Compare	CMP Op1,Op2	Op1-Op2	±				±	±	±	±	±
SAL	Shift arithmetic left (≡ SHL)	SAL Op,Quantity		*i*				±	±	?	±	±
SAR	Shift arithmetic right	SAR Op,Quantity		*i*				±	±	?	±	±
RCL	Rotate left through Carry	RCL Op,Quantity		*i*								±
RCR	Rotate right through Carry	RCR Op,Quantity		*i*								±
ROL	Rotate left	ROL Op,Quantity		*i*								±
ROR	Rotate right	ROR Op,Quantity		*i*								±

i for more informations see instruction specifications ♦ then CF:=0, OF:=0 else CF:=1, OF:=1

LOGIC				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
NEG	Negate (two-complement)	NEG Op	Op:=0-Op if Op=0 then CF:=0 else CF:=1	±				±	±	±	±	±
NOT	Invert each bit	NOT Op	Op:=¬Op (invert each bit)									
AND	Logical and	AND Dest,Source	Dest:=Dest∧Source	0				±	±	?	±	0
OR	Logical or	OR Dest,Source	Dest:=Dest∨Source	0				±	±	?	±	0
XOR	Logical exclusive or	XOR Dest,Source	Dest:=Dest (exor) Source	0				±	±	?	±	0
SHL	Shift logical left (≡ SAL)	SHL Op,Quantity		*i*				±	±	?	±	±
SHR	Shift logical right	SHR Op,Quantity		*i*				±	±	?	±	±

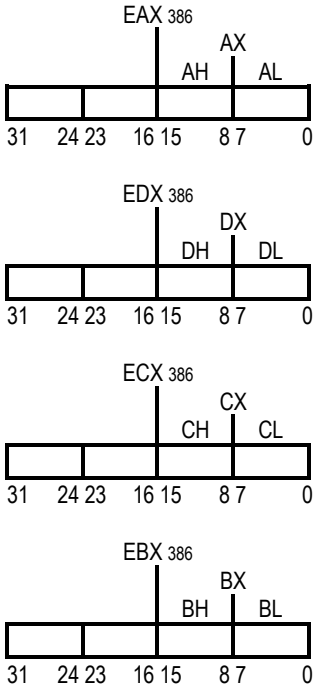
CodeTable 2/2

MISCELLANEOUS				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
NOP	No operation	NOP	No operation									
LEA	Load effective adress	LEA Dest,Source	Dest := address of Source									
INT	Interrupt	INT Nr	interrupts current program, runs spec. int-program			0	0					

JUMPS (flags remain unchanged)							
Name	**Comment**	**Code**	**Operation**	**Name**	**Comment**	**Code**	**Operation**
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(≡ JZ)	JNE	Jump if not Equal	JNE Dest	(≡ JNZ)
JZ	Jump if Zero	JZ Dest	(≡ JE)	JNZ	Jump if not Zero	JNZ Dest	(≡ JNE)
JCXZ	Jump if CX Zero	JCXZ Dest		JECXZ	Jump if ECX Zero	JECXZ Dest	386
JP	Jump if Parity (Parity Even)	JP Dest	(≡ JPE)	JNP	Jump if no Parity (Parity Odd)	JNP Dest	(≡ JPO)
JPE	Jump if Parity Even	JPE Dest	(≡ JP)	JPO	Jump if Parity Odd	JPO Dest	(≡ JNP)

Unsigned (Cardinal)				signed (Integer)			
JA	Jump if Above	JA Dest	(≡ JNBE)	JG	Jump if Greater	JG Dest	(≡ JNLE)
JAE	Jump if Above or Equal	JAE Dest	(≡ JNB ≡ JNC)	JGE	Jump if Greater or Equal	JGE Dest	(≡ JNL)
JB	Jump if Below	JB Dest	(≡ JNAE ≡ JC)	JL	Jump if Less	JL Dest	(≡ JNGE)
JBE	Jump if Below or Equal	JBE Dest	(≡ JNA)	JLE	Jump if Less or Equal	JLE Dest	(≡ JNG)
JNA	Jump if not Above	JNA Dest	(≡ JBE)	JNG	Jump if not Greater	JNG Dest	(≡ JLE)
JNAE	Jump if not Above or Equal	JNAE Dest	(≡ JB ≡ JC)	JNGE	Jump if not Greater or Equal	JNGE Dest	(≡ JL)
JNB	Jump if not Below	JNB Dest	(≡ JAE ≡ JNC)	JNL	Jump if not Less	JNL Dest	(≡ JGE)
JNBE	Jump if not Below or Equal	JNBE Dest	(≡ JA)	JNLE	Jump if not Less or Equal	JNLE Dest	(≡ JG)
JC	Jump if Carry	JC Dest		JO	Jump if Overflow	JO Dest	
JNC	Jump if no Carry	JNC Dest		JNO	Jump if no Overflow	JNO Dest	
				JS	Jump if Sign (= negative)	JS Dest	
				JNS	Jump if no Sign (= positive)	JNS Dest	

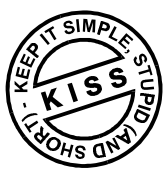
General Registers:



Example:

```
                .DOSSEG                 ; Demo program
                .MODEL SMALL
                .STACK 1024
Two             EQU 2                   ; Const
                .DATA
VarB            DB ?                    ; define Byte, any value
VarW            DW 1010b                ; define Word, binary
VarW2           DW 257                  ; define Word, decimal
VarD            DD 0AFFFFh              ; define Doubleword, hex
S               DB  Hello ! ,0          ; define String
                .CODE
main:           MOV AX,DGROUP           ; resolved by linker
                MOV DS,AX               ; init datasegment reg
                MOV [VarB],42           ; init VarB
                MOV [VarD],-7           ; set VarD
                MOV BX,Offset[S]        ; addr of  H  of  Hello !
                MOV AX,[VarW]           ; get value into accumul.
                ADD AX,[VarW2]          ; add VarW2 to AX
                MOV [VarW2],AX          ; store AX in VarW2

                MOV AX,4C00h            ; back to system
                INT 21h
                END main
```



Flags:					
ZF:	Zero	1 = ZR = result is Zero	0 = NZ = non Zero		
CF:	Carry	1 = CY = Carry / Borrow	0 = NC = no Carry	(unsigned = Cardinal)	
OF:	Overflow	1 = OV = Overflow / Underflow	0 = NV = no Overflow	(signed = Integer)	
SF:	Sign	1 = NG = Negative (resonable for Integer)	0 = PL = Plus	copy of highest bit of the result, even if it s a Cardinal !	