robotron

3

SOFTWARE HANDBUCH

MIKRORECHNER BETRIEBSSYSTEM SCP

Das ideale Nachschlagewerk für den Programmierer

Assembler-programmierung

Das vorliegende Nachschlagewerk entspricht dem Stand Juni 1986.

Das Softwarehandbuch wurde erarbeitet durch ein Autorenkollektiv der Kammer der Technik im VEB ROBOTRON, Büromaschinenwerk „Ernst Thälmann“ Sömmerda .

Dieses Nachschlagewerk wurde mit dem Textprogramm TP erarbeitet und auf dem Mosaikdrucker robotron K6313 ausgedruckt.

Autorenkollektiv: Dr. Ing. Brode, Mathias
Dipl.-Ing. Hubert, Martin
Dipl.-Math. Klein, Udo
Dipl.-Ing.-Ök. Fahr, Klaus

Herausgeber: VEB ROBOTRON Büromaschinenwerk
„Ernst Thälmann“ Sömmerda
Weißenseer Straße 52
Sömmerda
DDR - 5230

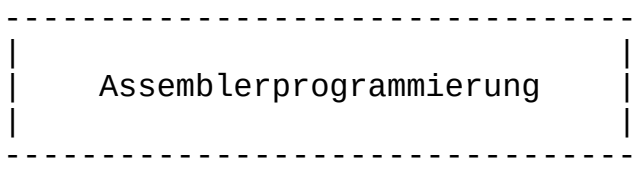
Sömmerda 1986

SOFTWARE - HANDBUCH

für das

Mikrorechner - Betriebssystem SCP

Das ideale Nachschlagewerk für den Programmierer



VEB ROBOTRON Büromaschinenwerk
"Ernst Thälmann" Sömmerda

Vorwort

Das vorliegende Systemhandbuch für Mikrorechner mit dem Betriebssystem SCP ist als ein ideales Nachschlagewerk für Programmierer in Beruf und Hobby geschrieben.

Dieses Handbuch besteht aus den Teilen

- Betriebssystem SCPX

- BASIC-Interpreter/-Compiler

- Assemblerprogrammierung

- Tabellen und Arbeitsblätter

und wird ständig erweitert.

Die Nutzung dieses Nachschlagewerkes setzt Grundkenntnisse über das Betriebssystem SCP sowie der unter Steuerung dieses Systems laufenden Software voraus.

Zur Aneignung von Grundkenntnissen und zur Vertiefung des Wissens über SCP-Software wird vom VEB ROBOTRON, Büromaschinenwerk "Ernst Thälmann", Sömmerda u.a. folgende Dokumentation bereitgestellt.

- Systemhandbuch SCP, Anleitung für den Bediener

- Systemhandbuch SCP, Anleitung für den Programmierer

- Systemhandbuch SCP, Assemblerprogrammierung

- Anwendungsbeschreibung Textprogramm

- Bedienungsanleitung und Sprachbeschreibung BASIC-Interpreter

- Bedienungsanleitung BASIC-Compiler

- Programmieranleitung BASIC

- Bedienungsanleitung und Sprachbeschreibung PASCAL-Compiler

Autorenkollektiv

INHALTSVERZEICHNIS

1. Z80 - Mikrobefehlsliste

In diesem Abschnitt wird ein Überblick über die Z80-Mikrobefehlsliste gegeben.
Die folgende Tabelle zeigt die Beeinflussung des Flag-Registers F mit den einzelnen Flags.

Das Flag-Register ist wie folgt aufgebaut:

	S	Z	X	H	X	P/V	N	C
Bit	7	6	5	4	3	2	1	0

Beschreibung der einzelnen Flags:

S Signum-Flag
   S = 1, wenn das Ergebnis negativ ist (Bit 7 = 1)
   S = 0, wenn das Ergebnis positiv ist (Bit 7 = 0).

Z Zero-Flag
   Z = 1, wenn das Ergebnis der Operation Null ist
   Z = 0, wenn das Ergebnis ungleich Null ist.

H Half-Carry-Flag
   H = 1, wenn durch Addition oder Subtraktion ein Übertrag von der unteren auf die obere Tetrade im Akku erzeugt wird.
   H = 0, sonst

P/V Paritäts- und Überlauf-Flag
   Parität (P) und Überlauf (V) benutzen das gleiche Flag. Logische Operationen beeinflussen dieses Flag entsprechend der Parität des Ergebnisses, während arithmetische Operationen dieses Flag durch Übertrag des Ergebnisses beeinflussen.
   Wenn P/V als Paritäts-Flag wirkt, so ist
      P/V = 1, wenn das Ergebnis der Operation paarig ist;
      P/V = 0, wenn das Ergebnis der Operation unpaarig ist.
   Wenn P/V als Überlauf-Flag wirkt, so ist
      P/V = 1, wenn durch die Operation ein Überlauf entsteht.

N Additions-/Subtraktions-Flag
   N = 1, wenn die vorgegebene Operation eine Subtraktion war. H- und N-Flags werden für die Dezimalkorrektur (DAA) benutzt, um das Ergebnis einer Addition oder Subtraktion von gepackten BCD-Zahlen wieder in das Format gepackter BCD-Zahlen zu wandeln (im Akku).

C Carry-Flag
   C = 1, wenn durch die Operation ein Übertrag erzeugt wird;
   C = 0, wenn kein Übertrag erzeugt wird.

X wird nicht verwendet

In der folgenden Tabelle wird die Beeinflussung der einzelnen Flags durch bestimmte Befehlsgruppen gezeigt.

Es gilt:
* Flag wird durch die Operation beeinflußt
- Flag wird nicht durch die Operation beeinflußt
0 Flag wird durch die Operation gelöscht
1 Flag wird durch die Operation gesetzt
X Flag unbestimmt
V P/V-Flag wird entsprechend bei Überlauf des Ergebnisses durch die Operation gesetzt
P P/V-Flag wird entsprechend der Parität des Ergebnisses durch die Operation gesetzt
IFF Interrupt-Annahme-Flipflop

Befehl	S	Z	H	P/V	N	C	Bemerkungen
ADD A,s	*	*	*	V	0	*	8-Bit-Addition ohne C-Flag
ADC A,s	*	*	*	V	0	*	8-Bit-Addition mit C-Flag
SUB s;	*	*	*	V	1	*	8-Bit-Subtrakt. ohne C-Flag
SBC A,s	*	*	*	V	1	*	8-Bit-Subtraktion mit C-Flag
CP s; NEG	*	*	*	V	1	*	Vergleich, Negation Akku
AND s	*	*	1	P	0	0	Logische Operation
OR s; XOR s	*	*	0	P	0	0	Logische Operationen
INC s	*	*	*	V	0	-	8-Bit-Inkrement
DEC s	*	*	*	V	1	-	8-Bit Dekrement
ADD HL,ss	-	-	X	-	0	*	16-Bit-Addition
ADC HL,ss	*	*	X	V	0	*	16-Bit-Addition mit C-Flag
SBC HL,ss	*	*	X	V	1	*	16-Bit-Subtrakt. mit C-Flag
RLA; RLCA; RRA; RRCA	-	-	0	-	0	*	Verschiebung von A um 1 Bit
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	*	*	0	P	0	*	Verschiebung von s um 1 Bit
RLD; RRD	*	*	0	P	0	*	Verschiebung tetradenweise
DAA	*	*	*	P	-	*	Dezimalkorrektur Akku
CPL	-	-	1	-	1	-	Komplement Akku
SCF	-	-	0	-	0	1	Setzen C-Flag (C=1)
CCF	-	-	X	-	0	*	Komplement C-Flag
IN r,(C)	X	*	0	P	0	-	Input-Register indirekt
INI; IND; OUTI; OUTD	X	*	X	X	1	-	Blockinput- und Blockoutput-befehle
INIR; INDR; OTIR; OTDR	X	1	X	X	1	-	Z=0, wenn B≠0; sonst Z=1
LDI; LDD	X	X	0	*	0	-	Blockübertragungsbefehle
LDIR; LDDR	X	X	0	0	0	-	P/V=1, wenn BC≠0; sonst P/V=0
CPI; CPIR; CPD; CPDR	X	*	X	*	1	-	Blocksuchbefehle Z=1, wenn A=(HL); sonst Z=0 P/V=1, wenn BC≠0; sonst P/V=0
LD A,I; LD A,R	*	*	0	IFF	0	-	Wet des IFF nach P/V-Flag
BIT b,s	X	*	1	X	0	-	Komplement von Bit b von s wird in Z-Flag eingetragen
NEG	*	*	*	V	1	*	Negation des Akkus

Für die verwendeten Registerabkürzungen gilt:

s ein 8-Bit-Speicherplatz, der durch eine der für den jeweiligen Befehl zulässigen Adressierungsarten definiert ist.
ss ein 16-Bit-Speicherplatz, der durch eine der für diesen Befehl zulässigen Adressierungsarten definiert ist.
r eines der Register A, B, C, D, E, H oder L
b Nummer des Bits, $0 \leq b \leq 7$

In der nachfolgenden Z80-Mikrobefehlsliste wird die Gesamtheit aller Mikrobefehle aufgeführt.

Darin werden

- die Befehlsschreibweise (Mnemonik)
- eine symbolische Befehlsbeschreibung
- die Beeinflussung der einzelnen Flags
- der Operationskode (binär)
- die Anzahl der Zyklen für diesen Befehl und
- zusätzliche Erläuterungen bzw. Bemerkungen

angegeben.

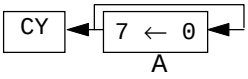
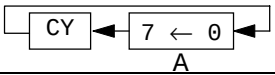
Als Abkürzungen werden verwendet:

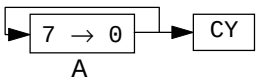
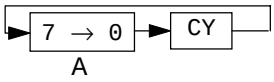
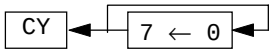
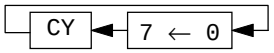
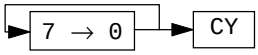
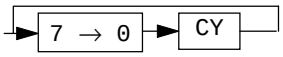
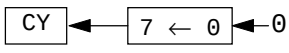
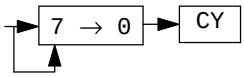
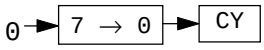
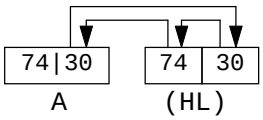
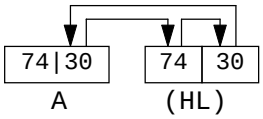
r	eines der Register A, B, C, D, E, H, oder L
r'	eines der Register A',B',C',D',E',H' oder L'
ss	eines der Registerpaare BC, DE, HL oder SP
rr	eines der Registerpaare BC, DE, IY oder SP
b	Bit-Nummer, $0 \leq b \leq 7$
d	Verschiebung bei Adressierung über Indexregister, $0 \leq d \leq 255$
e	relative Sprungadresse, $-128 \leq e \leq 127$
n	Konstante (1 Byte), $0 \leq n \leq 255$
nn	Konstante (2 Byte), $0 \leq nn \leq 65535$
°°H	obere 8 Bits des Registerpaares ss bzw. rr
°°L	untere 8 Bits des Registerpaares ss bzw. rr
-	Flag wird nicht beeinflußt
*	Flag wird entsprechend dem Ergebnis der Operation beeinflußt
0	Flag wird gelöscht (= 0)
1	Flag wird gesetzt (= 1)
X	Flag ist unbestimmt
*a	P/V-Flag = 0 wenn das Ergebnis von BC-1 = 0
	P/V-Flag = 1 sonst
*b	Z-Flag = 1 wenn A = (HL)
	Z-Flag = 0 sonst
*c	Z-Flag = 1 wenn das Ergebnis von BC-1 = 0
	Z-Flag = 0 sonst

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
LD r,r'	r ← r'	- - - - - -	01 r r'	4	r,r' | Reg.
LD r,n	r ← n	- - - - - -	00 r 110 n	7	000 | B 001 | C
LD r,(HL)	r ← (HL)	- - - - - -	01 r 110	7	010 | D
LD r,(IX+d)	r ← (IX+d)	- - - - - -	11 011 101 01 r 110 d	19	011 | E 100 | H 101 | L
LD r,(IY+d)	r ← (IY+d)	- - - - - -	11 111 101 01 r 110 d	19	111 | A
LD (HL),r	(HL) ← r	- - - - - -	01 110 r	7	
LD (IX+d),r	(IX+d) ← r	- - - - - -	11 011 101 01 110 r d	19	
LD (IY+d),r	(IY+d) ← r	- - - - - -	11 111 101 01 110 r d	19	
LD (HL),n	(HL) ← n	- - - - - -	00 110 110 n	10	
LD (IX+d),n	(IX+d) ← n	- - - - - -	11 011 101 00 110 110 d n	19	
LD (IY+d),n	(IY+d) ← n	- - - - - -	11 111 101 00 110 110 d n	19	
LD A,(BC)	A ← (BC)	- - - - - -	00 001 010	7	
LD A,(DE)	A ← (DE)	- - - - - -	00 011 010	7	
LD A,(nn)	A ← (nn)	- - - - - -	00 111 010 n n	13	
LD (BC),A	(BC) ← A	- - - - - -	00 000 010	7	
LD (DE),A	(DE) ← A	- - - - - -	00 010 010	7	
LD (nn),A	(nn) ← A	- - - - - -	00 110 010 n n	13	
LD A,I	A ← I	* * 0 IFF 0 -	11 101 101 01 010 111	9	I - Indexregister
LD A,R	A ← R	* * 0 IFF 0 -	11 101 101 01 011 111	9	
LD I,A	I ← A	- - - - - -	11 101 101 01 000 111	9	
LD R,A	R ← A	- - - - - -	11 101 101 01 001 111	9	
LD dd,nn	dd ← nn	- - - - - -	00 dd0 001 n n	10	dd | Reg. 00 | BC 01 | DE 10 | HL 11 | SP
LD IX,nn	IX ← nn	- - - - - -	11 011 101 00 100 001 n n	14	
LD IY,nn	IY ← nn	- - - - - -	11 111 101 00 100 001 n n	14	

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
LD HL,(nn)	H ← (nn+1) L ← (nn)	- - - - - -	00 101 010 n n	16	
LD dd,(nn)	dd_H ← (nn+1) dd_L ← (nn)	- - - - - -	11 101 101 01 dd1 011 n n	20	
LD IX,(nn)	IX_H ← (nn+1) IX_L ← (nn)	- - - - - -	11 011 101 00 101 010 n n	20	
LD IY,(nn)	IY_H ← (nn+1) IY_L ← (nn)	- - - - - -	11 111 101 00 101 010 n n	20	
LD (nn),HL	(nn+1) ← H (nn) ← L	- - - - - -	00 100 010 n n	16	
LD (nn),dd	(nn+1) ← dd_H (nn) ← dd_L	- - - - - -	11 101 101 01 dd0 011 n n	20	
LD (nn),IX	(nn+1) ← IX_H (nn) ← IX_L	- - - - - -	11 011 101 00 100 010 n n	20	
LD (nn),IY	(nn+1) ← IY_H (nn) ← IY_L	- - - - - -	11 111 101 00 100 010 n n	20	
LD SP,HL	SP ← HL	- - - - - -	11 111 001	6	
LD SP,IX	SP ← IX	- - - - - -	11 011 101 11 111 001	10	
LD SP,IY	SP ← IY	- - - - - -	11 111 101 11 111 001	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	- - - - - -	11 qq0 101	11	qq | Paar 00 | BC 01 | DE 10 | HL 11 | AF
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	- - - - - -	11 011 101 11 100 101	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	- - - - - -	11 111 101 11 100 101	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	- - - - - -	11 qq0 001	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	- - - - - -	11 011 101 11 100 001	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	- - - - - -	11 111 101 11 100 001	14	
EX DE,HL	DE ↔ HL	- - - - - -	11 101 011	4	
EX AF,AF'	AF ↔ AF'	- - - - - -	00 001 000	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	- - - - - -	11 011 001	4	
EX (SP),HL	H ↔ (SP+1) L ↔ (SP)	- - - - - -	11 100 011	19	

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
EX (SP),IX	$IX_H \leftrightarrow (SP+1)$ $IX_L \leftrightarrow (SP)$	- - - - - -	11 011 101 11 100 011	23	
EX (SP),IY	$IY_H \leftrightarrow (SP+1)$ $IY_L \leftrightarrow (SP)$	- - - - - -	11 111 101 11 100 011	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	X X 0 *a 0 -	11 101 101 10 100 000	16	Laden (HL) nach (DE) mit Erhöhen dieser Zeiger und dekrementieren des Zählers BC
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Wiederhlg. bis BC=0	X X 0 0 0 -	11 101 101 10 110 000	21 16	wenn BC ≠ 0 wenn BC = 0
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	X X 0 *a 0 -	11 101 101 10 101 000	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Wiederhlg. bis BC=0	X X 0 0 0 -	11 101 101 10 111 000	21 16	wenn BC ≠ 0 wenn BC = 0
CPI	A - (HL) HL ← HL+1 BC ← BC-1	X *b X *a 1 -	11 101 101 10 100 001	16	
CPIR	A - (HL) HL ← HL+1 BC ← BC-1 Wdhlg. bis A=(HL) oder BC=0	X *b X *a 1 -	11 101 101 10 110 001	21 16	wenn BC ≠ 0 und A ≠ (HL) wenn BC = 0 oder A = (HL)
CPD	A - (HL) HL ← HL-1 BC ← BC-1	X *b X *a 1 -	11 101 101 10 101 001	16	
CPDR	A - (HL) HL ← HL-1 BC ← BC-1 Wdhlg. bis A=(HL) oder BC=0	X *b X *a 1 -	11 101 101 10 111 001	21 16	wenn BC ≠ 0 und A ≠ (HL) wenn BC = 0 oder A = (HL)
ADD A,r	A ← A+r	* * * V 0 *	10 000 r	4	r | Reg. 000 | B 001 | C 010 | D 011 | E 100 | H 101 | L 111 | A
ADD A,n	A ← A+n	* * * V 0 *	11 000 110 n	7	
ADD A,(HL)	A ← A+(HL)	* * * V 0 *	10 000 110	7	
ADD A,(IX+d)	A ← A+(IX+d)	* * * V 0 *	11 011 101 10 000 110 d	19	
ADD A,(IY+d)	A ← A+(IY+d)	* * * V 0 *	11 111 101 10 000 110 d	19	

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
ADC A,s	A ← A+s+CY	* * * V 0 *	001		s=r,n,(HL),(IX+d)
SUB s	A ← A-s	* * * V 1 *	010		(IY+d) und ana-
SBC A,s	A ← A-s-CY	* * * V 1 *	011		log den ADD-Be-
AND s	A ← A^s	* * 1 P 0 0	100		fehlen. Für 000
OR s	A ← Avs	* * 0 P 0 0	110		im Op.-kode wer-
XOR s	A ← AΘs	* * 0 P 0 0	101		jetzt die ent-
CP s	A - s	* * * V 1 *	111		sprechenden
INC r	r ← r+1	* * * V 0 -	00 r 100	4	3 Bits eingetra-
INC (HL)	(HL) ← (HL)+1	* * * V 0 -	00 110 100	11	gen. CY = C-Flag
INC (IX+d)	(IX+d) ← (IX+d)+1	* * * V 0 -	11 011 101 00 110 100 d	23	
INC (IY+d)	(IY+d) ← (IX+d)+1	* * * V 0 -	11 111 101 00 110 100 d	23	
DEC m	m ← m-1	* * * V 1 -	101		m = r,(HL), (IX+d),(IY+d) und analog den INC-Befehlen. Für 100 im Op.-kode wird jetzt 101 eingetragen.
ADD HL,ss	HL ← HL+ss	- - X - 0 *	00 ss1 001	11	ss | Reg. 00 | BC 01 | DE 10 | HL 11 | SP
ADC HL,ss	HL ← HL+ss+CY	* * X V 0 *	11 101 101 01 ss1 010	15	
SBC HL,ss	HL ← HL-ss-CY	* * X V 1 *	11 101 101 01 ss0 010	15	
ADD IX,pp	IX ← IX+pp	- - X - 0 *	11 011 101 00 pp1 001	15	pp | Reg. 00 | BC 01 | DE 10 | IX 11 | SP
ADD IY,rr	IY ← IY+rr	- - X - 0 *	11 111 101 00 rr1 001		rr | Reg. 00 | BC 01 | DE 10 | IY 11 | SP
INC ss	ss ← ss+1	- - - - - -	00 ss0 011	6	
INC IX	IX ← IX+1	- - - - - -	11 011 101 00 100 011	10	
INC IY	IY ← IY+1	- - - - - -	11 111 101 00 100 011	10	
DEC ss	ss ← ss-1	- - - - - -	00 ss1 011	6	
DEC IX	IX ← IX-1	- - - - - -	11 011 101 00 101 011	10	
DEC IY	IX ← IY-1	- - - - - -	11 111 101 00 101 011	10	
RLCA		- - 0 - 0 *	00 000 111	4	Linksrotation um 1 Bit im Akku
RLA		- - 0 - 0 *	00 010 111	4	Linksrotation um 1 Bit im Akku über C-Flag

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
RRCA		- - 0 - 0 *	00 001 111	4	Rechtsrotation um 1 Bit im Akku
RRA		- - 0 - 0 *	00 011 111	4	Rechtsrotation um 1 Bit im Akku über C-Flag
RLC r		* * 0 P 0 *	11 001 011 00 000 r	8	Linksrotation um 1 Bit durch Reg. r
RLC (HL)		* * 0 P 0 *	11 001 011 00 000 110	15	
RLC (IX+d)		* * 0 P 0 *	11 011 101 11 001 011 d 00 000 110	23	
RLC (IY+d)		* * 0 P 0 *	11 111 101 11 001 011 d 00 000 110	23	
RL m		* * 0 P 0 *	010		Befehlsaufbau ist analog den RLC-Befehlen.
RRC m		* * 0 P 0 *	001		Für 000 im Op.-kode werden jetzt die entspr. 3 Bits eingetragen
RR m		* * 0 P 0 *	011		
SLA m		* * 0 P 0 *	100		
SRA m		* * 0 P 0 *	101		
SRL m		* * 0 P 0 *	111		
RLD		* * 0 P 0 -	11 101 101 01 101 111	18	Tetradenrotation links bzw. rechts zwischen Akku und (HL).
RRD		* * 0 P 0 -	11 101 101 01 100 111	18	Der Wert der oberen Tetrade des Akkus bleibt unverändert.

m = r,(HL),(IX+d),(IY+d)

_r_	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
DAA	Korrektur nach Add./Subtr. mit gepackt. Zahlen	* * * P - *	00 100 111	4	Dezimalkorrektur im Akku
CPL	$A \leftarrow \overline{A}$	- - 1 - 1 -	00 101 111	4	Komplement des Akkus (Einerkomplement)
NEG	$A \leftarrow \overline{A}+1$	* * * V 1 *	11 101 101 01 000 100	8	Negation des Akkus (Zweierkomplement)
CCF	$CY \leftarrow \overline{CY}$	- - X - 0 *	00 111 111	4	Komplement C-Flag
SCF	$CY \leftarrow 1$	- - 0 - 0 1	00 110 111	4	Setzen C-Flag (C=1)
NOP	keine Operation	- - - - - -	00 000 000	4	
HALT	CPU-Halt	- - - - - -	01 110 110	4	Halt (bis Interrupt)
DI	IFF ← 0	- - - - - -	11 110 011	4	Interrupt sperren
EI	IFF ← 1	- - - - - -	11 111 011	4	Interrupt freigeben
IMO	Setzen Interruptmodus 0	- - - - - -	11 101 101 01 000 110	8	
IM1	Setzen Interruptmodus 1	- - - - - -	11 101 101 01 010 110	8	
IM2	Setzen Interruptmodus 2	- - - - - -	11 101 101 01 011 110	8	
BIT b,r	$Z \leftarrow \overline{r_b}$	X * 1 X 0 -	11 001 011 01 b r	8	_r | Reg. 000 | B 001 | C 010 | D
BIT b,(HL)	$Z \leftarrow \overline{(HL)_b}$	X * 1 X 0 -	11 001 011 01 b 110	12	
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	X * 1 X 0 -	11 011 101 11 001 011 d 01 b 110	20	011 | E 100 | H 101 | L 111 | A
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	X * 1 X 0 -	11 111 101 11 001 011 d 01 b 110	20	_b | Testbit 000 | 0 001 | 1
SET b,r	$r_b \leftarrow 1$	- - - - - -	11 001 011 11 b r	8	010 | 2 011 | 3
SET b,(HL)	$(HL)_b \leftarrow 1$	- - - - - -	11 001 011 11 b 110	15	100 | 4 101 | 5
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	- - - - - -	11 011 101 11 001 011 d 11 b 110	23	110 | 6 111 | 7
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	- - - - - -	11 111 101 11 001 011 d 11 b 110	23	
RES b,m	$m_b \leftarrow 1$ m = r,(HL), (IX+d), (IY+d)	- - - - - -	10		Befehlsaufbau analog den SET-Befehlen. Im Op.-kode wird 11 durch 10 ersetzt.
JP nn	PC ← nn	- - - - - -	11 000 011 n n	10	

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
JP cc,nn	PC ← nn wenn Bedingung erfüllt, sonst weiter	- - - - - -	11 cc 010 n n	10	cc | Bedingung 000 | NZ non zero 001 | Z zero 010 | NC non carry 011 | C carry 100 | PO parity odd 101 | PE parity even 110 | P sign positiv 111 | M sign negativ
JR e	PC ← PC+e	- - - - - -	00 011 000 e-2	12	
JR C,e	C=0, weiter C=1, PC ← PC+e	- - - - - -	00 111 000 e-2	7 12	Bed. nicht erfüllt Bed. erfüllt
JR NC,e	C=1, weiter C=0, PC ← PC+e	- - - - - -	00 110 000 e-2	7 12	Bed. nicht erfüllt Bed. erfüllt
JR Z,e	Z=0, weiter Z=1, PC ← PC+e	- - - - - -	00 101 000 e-2	7 12	Bed. nicht erfüllt Bed. erfüllt
JR NZ,e	Z=1, weiter Z=0, PC ← PC+e	- - - - - -	00 100 000 e-2	7 12	Bed. nicht erfüllt Bed. erfüllt
JP (HL)	PC ← HL	- - - - - -	11 101 001	4	
JP (IX)	PC ← IX	- - - - - -	11 011 101 11 101 001	8	
JP (IY)	PC ← IY	- - - - - -	11 111 101 11 101 001		
DJNZ e	B ← B-1 B=0, weiter B≠0, PC ← PC+e	- - - - - -	00 010 000 e-2	8 13	wenn B = 0 wenn B ≠ 0
CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L PC ← nn	- - - - - -	11 001 101 n n	17	
CALL cc,nn	wenn Bed. cc nicht erfüllt, dann weiter; sonst wie CALL nn	- - - - - -	11 cc 100 n n	10 17	cc nicht erfüllt cc erfüllt
RET	PC_L ← (SP) PC_H ← (SP+1)	- - - - - -	11 001 001	10	
RET cc	wenn Bed. cc nicht erfüllt dnn weiter; sonst wie RET	- - - - - -	11 cc 000	5 11	cc nicht erfüllt cc erfüllt cc | Bedingung 000 | NZ non zero 001 | Z zero 010 | NC non carry 011 | C carry 100 | PO parity odd 101 | PE parity even 110 | P sign posit. 111 | PM sign negat.
RETI	Return vom Interrupt	- - - - - -	11 101 101 01 001 101	14	
RETN	Return vom nichtmaskier-ten Interrupt		11 101 101 01 000 101	14	

Mnemonik	Operation symbolisch	Flags S Z H P/V N C	OP.-kode 76 543 210	Anz. Zykl	Bemerkungen
RST p	(SP-1) ← PC_H (SP-2) ← PC_L PC_H ← 0 PC_L ← p	- - - - - -	11 t 111	11	t | p 000 | 00H 001 | 08H 010 | 10H 011 | 18H 100 | 20H 101 | 28H 110 | 30H 111 | 38H
IN A,(n)	A ← (n)	- - - - - -	11 011 011 n	11	
IN r,(C)	r ← (C)	* * 0 P 0 -	11 101 101 01 r 000	12	bei r=110 werden Flags beeinflußt
INI	(HL) ← (C) B ← B-1 HL ← HL+1	X $*_C$ X X 1 -	11 101 101 10 100 010	16	
INIR	(HL) ← (C) B ← B-1 HL ← HL+1 Wdhlg bis B=0	X 1 X X 1 -	11 101 101 10 110 010	21 16	wenn BC ≠ 0 wenn BC = 0
IND	(HL) ← (C) B ← B-1 HL ← HL-1	X $*_C$ X X 1 -	11 101 101 10 101 010	16	
INDR	(HL) ← (C) B ← B-1 HL ← HL-1 Wdhlg bis B=0	X 1 X X 1 -	11 101 101 10 111 010	21 16	wenn BC ≠ 0 wenn BC = 0
OUT (n),A	(n) ← A	- - - - - -	11 010 011 n	11	
OUT (C),r	(C) ← r	- - - - - -	11 101 101 01 r 001	12	
OUTI	(C) ← (HL) B ← B-1 HL ← HL+1	X $*_C$ X X 1 -	11 101 101 10 100 011	16	
[5] OTIR	(C) ← (HL) B ← B-1 HL ← HL+1 Wdhlg bis B=0	X 1 X X 1 -	11 101 101 10 110 011	21 16	wenn BC ≠ 0 wenn BC = 0
OUTD	(C) ← (HL) B ← B-1 HL ← HL-1	X $*_C$ X X 1 -	11 101 101 10 101 011	16	
OTDR	(C) ← (HL) B ← B-1 HL ← HL-1 Wdhlg bis B=0	X 1 X X 1 -	11 101 101 10 111 011	21 16	wenn BC ≠ 0 wenn BC = 0

2. Lexikografische Z80-Mikrobefehlsliste

Nachfolgend sind der Z80-Befehlssatz, der lexikografisch geordnet ist, die dazu entsprechende 8080-Mnemonik, der Operationskode und eine Kurzbeschreibung der Operation aufgeführt.

Operationskode	Mnemonik Z80		Mnemonik 8080		Operation
8E	ADC	A,(HL)	ADC	M	Addition von Akku,
DD 8E 05	ADC	A,(IX+d)	---		Operand und C-Flag
FD 8E 05	ADC	A,(IY+d)	---		Ergebnis im Akku
8F	ADC	A,A	ADC	A	
88	ADC	A,B	ADC	B	
89	ADC	A,C	ADC	C	
8A	ADC	A,D	ADC	D	
8B	ADC	A,E	ADC	E	
8C	ADC	A,H	ADC	H	
8D	ADC	A,L	ADC	L	
CE 20	ADC	A,n	ACI	n	
ED 4A	ADC	HL,BC	---		Addition von HL,
ED 5A	ADC	HL,DE	---		Registerpaar und
ED 6A	ADC	HL,HL	---		C-Flag
ED 7A	ADC	HL,SP	---		Ergebnis in HL
86	ADD	A,(HL)	ADD	M	Addition von Akku und
DD 86 05	ADD	A,(IX+d)	---		Operand
FD 86 05	ADD	A,(IY+d)	---		Ergebnis im Akku
87	ADD	A,A	ADD	A	
80	ADD	A,B	ADD	B	
81	ADD	A,C	ADD	C	
82	ADD	A,D	ADD	D	
83	ADD	A,E	ADD	E	
84	ADD	A,H	ADD	H	
85	ADD	A,L	ADD	L	
C6 20	ADD	A,n	ADI	n	
09	ADD	HL,BC	DAD	B	Addition von HL und
19	ADD	HL,DE	DAD	D	Registerpaar
29	ADD	HL,HL	DAD	H	Ergebnis in HL
39	ADD	HL,SP	DAD	SP	
DD 09	ADD	IX,BC	---		Addition von IX und
DD 19	ADD	IX,DE	---		Registerpaar
DD 29	ADD	IX,IX	---		Ergebnis in IX
DD 39	ADD	IX,SP	---		
FD 09	ADD	IY,BC	---		Addition von IY und
FD 19	ADD	IY,DE	---		Registerpaar
FD 29	ADD	IY,IY	---		Ergebnis in IY
FD 39	ADD	IY,SP	---		
A6	AND	(HL)	ANA	M	Logische 'UND'-Ver-
DD A6 05	AND	(IX+d)	---		knüpfung von Akku
FD A6 05	AND	(IY+d)	---		und Operand
A7	AND	A	ANA	A	Ergebnis im Akku
A0	AND	B	ANA	B	
A1	AND	C	ANA	C	
A2	AND	D	ANA	D	
A3	AND	E	ANA	E	

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
A4	AND	H	ANA	H	
A5	AND	L	ANA	L	
E6 20	AND	n	ANI	n	
CB 46	BIT	0,(HL)	---		Test von Bit b
DD CB 05 46	BIT	0,(IX+d)	---		(b = 0,1,...,7)
FD CB 05 46	BIT	0,(IY+d)	---		von Speicher oder
CB 47	BIT	0,A	---		Register
CB 40	BIT	0,B	---		
CB 41	BIT	0,C	---		
CB 42	BIT	0,D	---		
CB 43	BIT	0,E	---		
CB 44	BIT	0,H	---		
CB 45	BIT	0,L	---		
CB 4E	BIT	1,(HL)	---		
DD CB 05 4E	BIT	1,(IX+d)	---		
FD CB 05 4E	BIT	1,(IY+d)	---		
CB 4F	BIT	1,A	---		
CB 48	BIT	1,B	---		
CB 49	BIT	1,C	---		
CB 4A	BIT	1,D	---		
CB 4B	BIT	1,E	---		
CB 4C	BIT	1,H	---		
CB 4D	BIT	1,L	---		
CB 56	BIT	2,(HL)	---		
DD CB 05 56	BIT	2,(IX+d)	---		
FD CB 05 56	BIT	2,(IY+d)	---		
CB 57	BIT	2,A	---		
CB 50	BIT	2,B	---		
CB 51	BIT	2,C	---		
CB 52	BIT	2,D	---		
CB 53	BIT	2,E	---		
CB 54	BIT	2,H	---		
CB 55	BIT	2,L	---		
CB 5E	BIT	3,(HL)	---		
DD CB 05 5E	BIT	3,(IX+d)	---		
FD CB 05 5E	BIT	3,(IY+d)	---		
CB 5F	BIT	3,A	---		
CB 58	BIT	3,B	---		
CB 59	BIT	3,C	---		
CB 5A	BIT	3,D	---		
CB 5B	BIT	3,E	---		
CB 5C	BIT	3,H	---		
CB 5D	BIT	3,L	---		
CB 66	BIT	4,(HL)	---		
DD CB 05 66	BIT	4,(IX+d)	---		
FD CB 05 66	BIT	4,(IY+d)	---		
CB 67	BIT	4,A	---		
CB 60	BIT	4,B	---		
CB 61	BIT	4,C	---		
CB 62	BIT	4,D	---		
CB 63	BIT	4,E	---		
CB 64	BIT	4,H	---		
CB 65	BIT	4,L	---		
CB 6E	BIT	5,(HL)	---		

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
DD CB 05 6E	BIT	5,(IX+d)	---		
FD CB 05 6E	BIT	5,(IY+d)	---		
CB 6F	BIT	5,A	---		
CB 68	BIT	5,B	---		
CB 69	BIT	5,C	---		
CB 6A	BIT	5,D	---		
CB 6B	BIT	5,E	---		
CB 6C	BIT	5,H	---		
CB 6D	BIT	5,L	---		
CB 76	BIT	6,(HL)	---		
DD CB 05 76	BIT	6,(IX+d)	---		
FD CB 05 76	BIT	6,(IY+d)	---		
CB 77	BIT	6,A	---		
CB 70	BIT	6,B	---		
CB 71	BIT	6,C	---		
CB 72	BIT	6,D	---		
CB 73	BIT	6,E	---		
CB 74	BIT	6,H	---		
CB 75	BIT	6,L	---		
CB 7E	BIT	7,(HL)	---		
DD CB 05 7E	BIT	7,(IX+d)	---		
FD CB 05 7E	BIT	7,(IY+d)	---		
CB 7F	BIT	7,A	---		
CB 78	BIT	7,B	---		
CB 79	BIT	7,C	---		
CB 7A	BIT	7,D	---		
CB 7B	BIT	7,E	---		
CB 7C	BIT	7,H	---		
CB 7D	BIT	7,L	---		
DC 84 05	CALL	C,nn	CC	nn	Aufruf der Programm-
FC 84 05	CALL	M,nn	CM	nn	routine an der
D4 84 05	CALL	NC,nn	CNC	nn	Stelle nn, wobei die
C4 84 05	CALL	NZ,nn	CNZ	nn	Bedingung erfüllt ist.
F4 84 05	CALL	P,nn	CP	nn	
EC 84 05	CALL	PE,nn	CPE	nn	
E4 84 05	CALL	PO,nn	CPO	nn	
CC 84 05	CALL	Z,nn	CZ	nn	
CD 84 05	CALL	nn	CALL	nn	Die an der Stelle nn stehende Programmrou- tine wird aufgerufen
3F	CCF		CMC		Komplement Carry-Flag
BE	CP	(HL)	CMP	M	Vergleich von Akku
DD BE 05	CP	(IX+d)	---		und Operand
FD BE 05	CP	(IY+D)	---		(A - Operand)
BF	CP	A	CMP	A	Ergebnis im Akku
B8	CP	B	CMP	B	
B9	CP	C	CMP	C	
BA	CP	D	CMP	D	
BB	CP	E	CMP	E	
BC	CP	H	CMP	H	
BD	CP	L	CMP	L	
FE 20	CP	n	CPI	n	

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
ED A9	CPD		---		Vergleich von Akku und (HL), Dekrement von HL und BC
ED B9	CPDR		---		Vergleich von Akku und (HL), Dekrement von HL und BC Vergleich wird wiederderholt, bis A = (HL) oder BC = 0
ED A1	CPI		---		Vergleich von Akku und (HL), Inkrement HL und Dekrement BC
ED B1	CPIR		---		Vergleich von Akku und (HL), Inkrement HL und Dekrement BC Vergleich wird wiederholt, bis A = (HL) oder BC = 0
2F	CPL		CMA		Komplement Akku (Einerkomplement)
27	DAA		DAA		Dezimalkorrektur Akku
35	DEC	(HL)	DCR	M	Dekrement von Operand
DD 35 05	DEC	(IX+d)	---		
FD 35 05	DEC	(IY+d)	---		
3D	DEC	A	DCR	A	
05	DEC	B	DCR	B	
0B	DEC	BC	DCX	B	
0D	DEC	C	DCR	C	
15	DEC	D	DCR	D	
1B	DEC	DE	DCX	D	
1D	DEC	E	DCR	E	
25	DEC	H	DCR	H	
2B	DEC	HL	DCX	H	
DD 2B	DEC	IX	---		
FD 2B	DEC	IY	---		
2D	DEC	L	DCR	L	
3B	DEC	SP	DCX	SP	
F3	DI		DI		Interrupt sperren
10 2E	DJNZ	e	---		Dekrement von B und relativer Sprung, wenn B ≠ 0
FB	EI		EI		Interrupt freigeben
E3	EX	(SP),HL	XTHL		Tausch von (SP) und HL bzw. IX oder IY
DD E3	EX	(SP),IX	---		
FD E3	EX	(SP),IY	---		

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
08	EX	AF,AF	---		Tausch der Werte von AF und AF'
EB	EX	DE,HL	XCHG		Tausch der Werte von DE und HL
D9	EXX		---		Tausch der Werte von BC,DE,HL mit den Wertten von BC',DE',HL'
76	HALT		HLT		Halt, bis Interrupt
ED 46	IM	0	---		Setzen des Interrupt-modus
ED 56	IM	1	---		
ED 5E	IM	2	---		
ED 78	IN	A,(C)	---		Laden des Registers mit Input von (C)
ED 40	IN	B,(C)	---		
ED 48	IN	C,(C)	---		
ED 50	IN	D,(C)	---		
ED 58	IN	E,(C)	---		
ED 60	IN	H,(C)	---		
ED 68	IN	L,(C)	---		
34	INC	(HL)	INR	M	Inkrement des Operanden
DD 34 05	INC	(IX+d)	---		
FD 34 05	INC	(IY+d)	---		
3C	INC	A	INR	A	
04	INC	B	INR	B	
03	INC	BC	INX	B	
0C	INC	C	INR	C	
14	INC	D	INR	D	
13	INC	DE	INX	D	
1C	INC	E	INR	E	
24	INC	H	INR	H	
23	INC	HL	INX	H	
DD 23	INC	IX	---		
FD 23	INC	IY	---		
2C	INC	L	INR	L	
33	INC	SP	INX	SP	
DB 20	IN	A,(n)	IN	n	Laden des Akkus mit Input von n
ED AA	IND		---		Laden (HL) mit Input von Port (C) Dekrement von HL und B
ED BA	INDR		---		Laden (HL) mit Input von Port (C) Dekrement von HL und B Wiederholung bis B = 0
ED A2	INI		---		Laden (HL) mit Input von Port (C) Inkrement HL und Dekrement B

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
ED B2	INIR		---		Laden (HL) mit Input von Port (C) Inkrement HL und Dekrement B Wiederholung bis B = 0
E9	JP	(HL)	PCHL		Unbedingter Sprung an die adressierte Stelle
DD E9	JP	(IX)	---		
C3 84 05	JP	nn	JMP	nn	
FD E9	JP	(IY)	---		
DA 84 05	JP	C,nn	JC	nn	Bedingter Sprung an die Stelle nn, wenn die Bedingung erfüllt ist
FA 84 05	JP	M,nn	JM	nn	
D2 84 05	JP	NC,nn	JNC	nn	
C2 84 05	JP	NZ,nn	JNZ	nn	
F2 84 05	JP	P,nn	JP	nn	
EA 84 05	JP	PE,nn	JPE	nn	
E2 84 05	JP	PO,nn	JPO	nn	
CA 84 05	JP	Z,nn	JZ	nn	
38 2E	JR	C,e	---		Bedingter relativer Sprung nach PC + e, wenn die Bedingung erfüllt ist
30 2E	JR	NC,e	---		
20 2E	JR	NZ,e	---		
28 2E	JR	Z,e	---		
18 2E	JR	e	---		Unbedingter relativer Sprung nach PC + e
02	LD	(BC),A	STAX	B	Laden Ziel mit Quelle
12	LD	(DE),A	STAX	D	
77	LD	(HL),A	MOV	M,A	
70	LD	(HL),B	MOV	M,B	
71	LD	(HL),C	MOV	M,C	
72	LD	(HL),D	MOV	M,D	
73	LD	(HL),E	MOV	M,E	
74	LD	(HL),H	MOV	M,H	
75	LD	(HL),L	MOV	M,L	
36 20	LD	(HL),n	MVI	M,n	
DD 77 05	LD	(IX+d),A	---		
DD 70 05	LD	(IX+d),B	---		
DD 71 05	LD	(IX+d),C	---		
DD 72 05	LD	(IX+d),D	---		
DD 73 05	LD	(IX+d),E	---		
DD 74 05	LD	(IX+d),H	---		
DD 75 05	LD	(IX+d),L	---		
DD 36 05 20	LD	(IX+d),n	---		
FD 77 05	LD	(IY+d),A	---		
FD 70 05	LD	(IY+d),B	---		
FD 71 05	LD	(IY+d),C	---		
FD 72 05	LD	(IY+d),D	---		
FD 73 05	LD	(IY+d),E	---		
FD 74 05	LD	(IY+d),H	---		
FD 75 05	LD	(IY+d),L	---		
FD 36 05 20	LD	(IY+d),n	---		
32 84 05	LD	(nn),A	STA	nn	
ED 43 84 05	LD	(nn),BC	---		
ED 53 84 05	LD	(nn),DE	---		

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
22 84 05	LD	(nn),HL	SHLD	nn	
DD 22 84 05	LD	(nn),IX	---		
FD 22 84 05	LD	(nn),IY	---		
ED 73 84 05	LD	(nn),SP	---		
0A	LD	A,(BC)	LDAX	B	
1A	LD	A,(DE)	LDAX	D	
7E	LD	A,(HL)	MOV	A,M	
DD 7E 05	LD	A,(IX+d)	---		
FD 7E 05	LD	A,(IY+d)	---		
3A 84 05	LD	A,(nn)	LDA	nn	
7F	LD	A,A	MOV	A,A	
78	LD	A,B	MOV	A,B	
79	LD	A,C	MOV	A,C	
7A	LD	A,D	MOV	A,D	
7B	LD	A,E	MOV	A,E	
7C	LD	A,H	MOV	A,H	
ED 57	LD	A,I	---		
7D	LD	A,L	MOV	A,L	
3E 20	LD	A,n	MVI	A,n	
ED 5F	LD	A,R	---		
46	LD	B,(HL)	MOV	B,M	
DD 46 05	LD	B,(IX+d)	---		
FD 46 05	LD	B,(IY+d)	---		
47	LD	B,A	MOV	B,A	
40	LD	B,B	MOV	B,B	
41	LD	B,C	MOV	B,C	
42	LD	B,D	MOV	B,D	
43	LD	B,E	MOV	B,E	
44	LD	B,H	MOV	B,H	
45	LD	B,L	MOV	B,L	
06 20	LD	B,n	MVI	B,n	
ED 4B 84 05	LD	BC,(nn)	---		
01 84 05	LD	BC,nn	LXI	B,nn	
4E	LD	C,(HL)	MOV	C,M	
DD 4E 05	LD	C,(IX+d)	---		
FD 4E 05	LD	C,(IY+d)	---		
4F	LD	C,A	MOV	C,A	
48	LD	C,B	MOV	C,B	
49	LD	C,C	MOV	C,C	
4A	LD	C,D	MOV	C,D	
4B	LD	C,E	MOV	C,E	
4C	LD	C,H	MOV	C,H	
4D	LD	C,L	MOV	C,L	
0E 20	LD	C,n	MVI	C,n	
56	LD	D,(HL)	MOV	D,M	
DD 56 05	LD	D,(IX+d)	---		
FD 56 05	LD	D,(IY+d)	---		
57	LD	D,A	MOV	D,A	
50	LD	D,B	MOV	D,B	
51	LD	D,C	MOV	D,C	
52	LD	D,D	MOV	D,D	
53	LD	D,E	MOV	D,E	
54	LD	D,H	MOV	D,H	

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
55	LD	D,L	MOV	D,L	
16 20	LD	D,n	MVI	D,n	
ED 5B 84 05	LD	DE,(nn)	---		
11 84 05	LD	DE,nn	LXI	D,nn	
5E	LD	E,(HL)	MOV	E,M	
DD 5E 05	LD	E,(IX+d)	---		
FD 5E 05	LD	E,(IY+d)	---		
5F	LD	E,A	MOV	E,A	
58	LD	E,B	MOV	E,B	
59	LD	E,C	MOV	E,C	
5A	LD	E,D	MOV	E,D	
5B	LD	E,E	MOV	E,E	
5C	LD	E,H	MOV	E,H	
5D	LD	E,L	MOV	E,L	
1E 20	LD	E,n	MVI	E,n	
66	LD	H,(HL)	MOV	H,M	
DD 66 05	LD	H,(IX+d)	---		
FD 66 05	LD	H,(IY+d)	---		
67	LD	H,A	MOV	H,A	
60	LD	H,B	MOV	H,B	
61	LD	H,C	MOV	H,C	
62	LD	H,D	MOV	H,D	
63	LD	H,E	MOV	H,E	
64	LD	H,H	MOV	H,H	
65	LD	H,L	MOV	H,L	
26 20	LD	H,n	MVI	H,n	
2A 84 05	LD	HL,(nn)	LHLD	nn	
21 84 05	LD	HL,nn	LXI	H,nn	
ED 47	LD	I,A	---		
DD 2A 84 05	LD	IX,(nn)	---		
DD 21 84 05	LD	IX,nn	---		
FD 2A 84 05	LD	IY,(nn)	---		
FD 21 84 05	LD	IY,nn	---		
6E	LD	L,(HL)	MOV	L,M	
DD 6E 05	LD	L,(IX+d)	---		
FD 6E 05	LD	L,(IY+d)	---		
6F	LD	L,A	MOV	L,A	
68	LD	L,B	MOV	L,B	
69	LD	L,C	MOV	L,C	
6A	LD	L,D	MOV	L,D	
6B	LD	L,E	MOV	L,E	
6C	LD	L,H	MOV	L,H	
6D	LD	L,L	MOV	L,L	
2E 20	LD	L,n	MVI	L,n	
ED 4F	LD	R,A	---	L,M	
ED 7B 84 05	LD	SP,(nn)	---		
F9	LD	SP,HL	SPHL		
DD F9	LD	SP,IX	---		
FD F9	LD	SP,IY	---		
31 84 05	LD	SP,nn	LXI	SP,nn	
ED A8	LDD		---		Laden (DE) mit (HL) Dekrement DE, HL, BC

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
ED B8	LDDR		---		Laden (DE) mit (HL) Dekrement DE, HL, BC Wiederholung bis BC=0
ED A0	LDI		---		Laden (DE) mit (HL) Inkrement DE, HL Dekrement BC
ED B0	LDIR		---		Laden (DE) mit (HL) Inkrement DE, HL Dekrement BC Wiederholung bis BC=0
ED 44	NEG		---		Negation Akku (Zweierkomplement)
00	NOP		NOP		keine Operation
B6	OR	(HL)	ORA	M	Logische ODER-Ver- knüpfung von Operand und Akku Ergebnis im Akku
DD B6 05	OR	(IX+d)	---		
FD B6 05	OR	(IY+d)	---		
B7	OR	A	ORA	A	
B0	OR	B	ORA	B	
B1	OR	C	ORA	C	
B2	OR	D	ORA	D	
B3	OR	E	ORA	E	
B4	OR	H	ORA	H	
B5	OR	L	ORA	L	
F6 20	OR	n	ORI	n	
ED BB	OTDR		---		Laden Output-Port (C) mit (HL) Dekrement HL und B Wiederholung bis B=0
ED B3	OTIR		---		Laden Output-Port (C) mit (HL), Inkrement HL, Dekrement B Wiederholung bis B=0
ED 79	OUT	(C),A	---		Laden Output-Port (C) mit Register
ED 41	OUT	(C),B	---		
ED 49	OUT	(C),C	---		
ED 51	OUT	(C),D	---		
ED 59	OUT	(C),E	---		
ED 61	OUT	(C),H	---		
ED 69	OUT	(C),L	---		
D3 20	OUT	(n),A	OUT	n	Laden Output-Port (n) mit Akku
ED AB	OUTD		---		Laden Output-Port (C) mit (HL) Dekrement HL und B
ED A3	OUTI		---		Laden Output-Port (C) mit (HL); Inkrement HL Dekrement B
F1	POP	AF	POP	PSW	Laden des Register- paares aus dem Stapel
C1	POP	BC	POP	B	
D1	POP	DE	POP	D	
E1	POP	HL	POP	H	

Operations-kode	Mnemonik Z80	Mnemonik 8080	Operation
DD E1	POP IX	---	Laden des Register-
FD E1	POP IY	---	paares aus dem Stapel
F5	PUSH AF	PUSH PSW	Laden des Stapels
C5	PUSH BC	PUSH B	durch das Registerpaar
D5	PUSH DE	PUSH D	
E5	PUSH HL	PUSH H	
DD E5	PUSH IX	---	
FD E5	PUSH IY	---	
CB 86	RES 0,(HL)	---	Löschen des Bits b
DD CB 05 86	RES 0,(IX+d)	---	(b = 0,1,...,7)
FD CB 05 86	RES 0,(IY+d)	---	des Operanden
CB 87	RES 0,A	---	
CB 80	RES 0,B	---	
CB 81	RES 0,C	---	
CB 82	RES 0,D	---	
CB 83	RES 0,E	---	
CB 84	RES 0,H	---	
CB 85	RES 0,L	---	
CB 8E	RES 1,(HL)	---	
DD CB 05 8E	RES 1,(IX+d)	---	
FD CB 05 8E	RES 1,(IY+d)	---	
CB 8F	RES 1,A	---	
CB 88	RES 1,B	---	
CB 89	RES 1,C	---	
CB 8A	RES 1,D	---	
CB 8B	RES 1,E	---	
CB 8C	RES 1,H	---	
CB 8D	RES 1,L	---	
CB 96	RES 2,(HL)	---	
DD CB 05 96	RES 2,(IX+d)	---	
FD CB 05 96	RES 2,(IY+d)	---	
CB 97	RES 2,A	---	
CB 90	RES 2,B	---	
CB 91	RES 2,C	---	
CB 92	RES 2,D	---	
CB 93	RES 2,E	---	
CB 94	RES 2,H	---	
CB 95	RES 2,L	---	
CB 9E	RES 3,(HL)	---	
DD CB 05 9E	RES 3,(IX+d)	---	
FD CB 05 9E	RES 3,(IY+d)	---	
CB 9F	RES 3,A	---	
CB 98	RES 3,B	---	
CB 99	RES 3,C	---	
CB 9A	RES 3,D	---	
CB 9B	RES 3,E	---	
CB 9C	RES 3,H	---	
CB 9D	RES 3,L	---	
CB A6	RES 4,(HL)	---	
DD CB 05 A6	RES 4,(IX+d)	---	
FD CB 05 A6	RES 4,(IY+d)	---	
CB A7	RES 4,A	---	
CB A0	RES 4,B	---	
CB A1	RES 4,C	---	

Operations-kode	Mnemonik Z80		Mnemonik 8080	Operation
CB A2	RES	4,D	---	
CB A3	RES	4,E	---	
CB A4	RES	4,H	---	
CB A5	RES	4,L	---	
CB AE	RES	5,(HL)	---	
DD CB 05 AE	RES	5,(IX+d)	---	
FD CB 05 AE	RES	5,(IY+d)	---	
CB AF	RES	5,A	---	
CB A8	RES	5,B	---	
CB A9	RES	5,C	---	
CB AA	RES	5,D	---	
CB AB	RES	5,E	---	
CB AC	RES	5,H	---	
CB AD	RES	5,L	---	
CB B6	RES	6,(HL)	---	
DD CB 05 B6	RES	6,(IX+d)	---	
FD CB 05 B6	RES	6,(IY+d)	---	
CB B7	RES	6,A	---	
CB B0	RES	6,B	---	
CB B1	RES	6,C	---	
CB B2	RES	6,D	---	
CB B3	RES	6,E	---	
CB B4	RES	6,H	---	
CB B5	RES	6,L	---	
CB BE	RES	7,(HL)	---	
DD CB 05 BE	RES	7,(IX+d)	---	
FD CB 05 BE	RES	7,(IY+d)	---	
CB BF	RES	7,A	---	
CB B8	RES	7,B	---	
CB B9	RES	7,C	---	
CB BA	RES	7,D	---	
CB BB	RES	7,E	---	
CB BC	RES	7,H	---	
CB BD	RES	7,L	---	
C9	RET		RET	Return vom Unter-programm
D8	RET	C	RC	Bedingter Return vom
F8	RET	M	RM	Unterprogramm, wenn
D0	RET	NC	RNC	die Bedingung erfüllt
C0	RET	NZ	RNZ	ist
F0	RET	P	RP	
E8	RET	PE	RPE	
E0	RET	PO	RPO	
C8	RET	Z	RZ	
ED 4D	RETI		---	Return vom Interrupt
ED 45	RETN		---	Return vom nicht-maskierten Interrupt
CB 16	RL	(HL)	---	Linksrotation um 1
DD CB 05 16	RL	(IX+d)	---	über C-Flag
FD CB 05 16	RL	(IY+d)	---	
CB 17	RL	A	---	
CB 10	RL	B	---	
CB 11	RL	C	---	
CB 12	RL	D	---	

Operations-kode	Mnemonik Z80		Mnemonik 8080	Operation
CB 13	RL	E	---	
CB 14	RL	H	---	
CB 15	RL	L	---	
17	RLA		RAL	Linksrotation um 1 Bit im Akku über C-Flag
CB 06	RLC	(HL)	---	Linksrotation um 1 Bit
DD CB 05 06	RLC	(IX+d)	---	
FD CB 05 06	RLC	(IY+d)	---	
CB 07	RLC	A	---	
CB 00	RLC	B	---	
CB 01	RLC	C	---	
CB 02	RLC	D	---	
CB 03	RLC	E	---	
CB 04	RLC	H	---	
CB 05	RLC	L	---	
07	RLCA		RLC	Linksrotation um 1 Bit im Akku
ED 6F	RLD		---	Tetradenrotation links und rechts zwischen Akku u. (HL)
CB 1E	RR	(HL)	---	Linksrotation um 1 Bit
DD CB 05 1E	RR	(IX+d)	---	über C-Flag
FD CB 05 1E	RR	(IY+d)	---	
CB 1F	RR	A	---	
CB 18	RR	B	---	
CB 19	RR	C	---	
CB 1A	RR	D	---	
CB 1B	RR	E	---	
CB 1C	RR	H	---	
CB 1D	RR	L	---	
1F	RRA		RAR	Rechtsrotation um 1 Bit im Akku über C-Flag
CB 0E	RRC	(HL)	---	Rechtsrotation um
DD CB 05 0E	RRC	(IX+d)	---	1 Bit
FD CB 05 0E	RRC	(IY+d)	---	
CB 0F	RRC	A	---	
CB 08	RRC	B	---	
CB 09	RRC	C	---	
CB 0A	RRC	D	---	
CB 0B	RRC	E	---	
CB 0C	RRC	H	---	
CB 0D	RRC	L	---	
0F	RRCA		RRC	Rechtsrotation um 1 Bit im Akku
ED 67	RRD		---	Tetradenoperation rechts und links zwischen Akku u. (HL)

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
C7	RST	00H	RST	0	Restart nach Adresse
CF	RST	08H	RST	1	
D7	RST	10H	RST	2	
DF	RST	18H	RST	3	
E7	RST	20H	RST	4	
EF	RST	28H	RST	5	
F7	RST	30H	RST	6	
FF	RST	38H	RST	7	
9E	SBC	A,(HL)	SBB	M	Subtraktion von
DD 9E 05	SBC	A,(IX+d)	---		Operand und C-Flag
FD 9E 05	SBC	A,(IY+d)	---		vom Akku
9F	SBC	A,A	SBB	A	Ergebnis im Akku
98	SBC	A,B	SBB	B	
99	SBC	A,C	SBB	C	
9A	SBC	A,D	SBB	D	
9B	SBC	A,E	SBB	E	
9C	SBC	A,H	SBB	H	
9D	SBC	A,L	SBI	L	
DE 20	SBC	A,n	SBI	n	
ED 42	SBC	HL,BC	---		Subtraktion von
ED 52	SBC	HL,DE	---		Registerpaar und
ED 62	SBC	HL,HL	---		C-Flag von HL
ED 72	SBC	HL,SP	---		Ergebnis in HL
37	SCF		STC		Setzen des Carry-Flags (C=1)
CB C6	SET	0,(HL)	---		Setzen von Bit b
DD CB 05 C6	SET	0,(IX+d)	---		(b = 0,1,...,7)
FD CB 05 C6	SET	0,(IY+d)	---		des Operanden
CB C7	SET	0,A	---		
CB C0	SET	0,B	---		
CB C1	SET	0,C	---		
CB C2	SET	0,D	---		
CB C3	SET	0,E	---		
CB C4	SET	0,H	---		
CB C5	SET	0,L	---		
CB CE	SET	1,(HL)	---		
DD CB 05 CE	SET	1,(IX+d)	---		
FD CB 05 CE	SET	1,(IY+d)	---		
CB CF	SET	1,A	---		
CB C8	SET	1,B	---		
CB C9	SET	1,C	---		
CB CA	SET	1,D	---		
CB CB	SET	1,E	---		
CB CC	SET	1,H	---		
CB CD	SET	1,L	---		
CB D6	SET	2,(HL)	---		
DD CB 05 D6	SET	2,(IX+d)	---		
FD CB 05 D6	SET	2,(IY+d)	---		
CB D7	SET	2,A	---		
CB D0	SET	2,B	---		
CB D1	SET	2,C	---		
CB D2	SET	2,D	---		
CB D3	SET	2,E	---		

Operations-kode	Mnemonik Z80		Mnemonik 8080	Operation
CB D4	SET	2,H	---	
CB D5	SET	2,L	---	
CB DE	SET	3,(HL)	---	
DD CB 05 DE	SET	3,(IX+d)	---	
FD CB 05 DE	SET	3,(IY+d)	---	
CB DF	SET	3,A	---	
CB D8	SET	3,B	---	
CB D9	SET	3,C	---	
CB DA	SET	3,D	---	
CB DB	SET	3,E	---	
CB DC	SET	3,H	---	
CB DD	SET	3,L	---	
CB E6	SET	4,(HL)	---	
DD CB 05 E6	SET	4,(IX+d)	---	
FD CB 05 E6	SET	4,(IY+d)	---	
CB E7	SET	4,A	---	
CB E0	SET	4,B	---	
CB E1	SET	4,C	---	
CB E2	SET	4,D	---	
CB E3	SET	4,E	---	
CB E4	SET	4,H	---	
CB E5	SET	4,L	---	
CB EE	SET	5,(HL)	---	
DD CB 05 EE	SET	5,(IX+d)	---	
FD CB 05 EE	SET	5,(IY+d)	---	
CB EF	SET	5,A	---	
CB E8	SET	5,B	---	
CB E9	SET	5,C	---	
CB EA	SET	5,D	---	
CB EB	SET	5,E	---	
CB EC	SET	5,H	---	
CB ED	SET	5,L	---	
CB F6	SET	6,(HL)	---	
DD CB 05 F6	SET	6,(IX+d)	---	
FD CB 05 F6	SET	6,(IY+d)	---	
CB F7	SET	6,A	---	
CB F0	SET	6,B	---	
CB F1	SET	6,C	---	
CB F2	SET	6,D	---	
CB F3	SET	6,E	---	
CB F4	SET	6,H	---	
CB F5	SET	6,L	---	
CB FE	SET	7,(HL)	---	
DD CB 05 FE	SET	7,(IX+d)	---	
FD CB 05 FE	SET	7,(IY+d)	---	
CB FF	SET	7,A	---	
CB F8	SET	7,B	---	
CB F9	SET	7,C	---	
CB FA	SET	7,D	---	
CB FB	SET	7,E	---	
CB FC	SET	7,H	---	
CB FD	SET	7,L	---	
CB 26	SLA	(HL)	---	Linksverschiebung des
DD CB 05 26	SLA	(IX+d)	---	Operanden um 1 Bit

Operations-kode	Mnemonik Z80		Mnemonik 8080		Operation
FD CB 05 26	SLA	(IY+d)	---		mit Bit 7 nach C-Flag
CB 27	SLA	A	---		und 0 nach Bit 0
CB 20	SLA	B	---		
CB 21	SLA	C	---		
CB 22	SLA	D	---		
cb 23	SLA	E	---		
CB 24	SLA	H	---		
CB 25	SLA	L	---		
CB 2E	SRA	(HL)	---		Rechtsverschiebung des
DD CB 05 2E	SRA	(IX+d)	---		Operanden um 1 Bit mit
FD CB 05 2E	SRA	(IY+d)	---		Bit 7 bleibt erhalten
CB 2F	SRA	A	---		und Bit 0 nach C-Flag
CB 28	SRA	B	---		
CB 29	SRA	C	---		
CB 2A	SRA	D	---		
CB 2B	SRA	E	---		
CB 2C	SRA	H	---		
CB 2D	SRA	L	---		
CB 3E	SRL	(HL)	---		Rechtsverschiebung des
DD CB 05 3E	SRL	(IX+d)	---		Operanden um 1 Bit mit
FD CB 05 3E	SRL	(IY+d)	---		0 nach Bit 7 und Bit 0
CB 3F	SRL	A	---		nach C-Flag
CB 38	SRL	B	---		
CB 39	SRL	C	---		
CB 3A	SRL	D	---		
CB 3B	SRL	E	---		
CB 3C	SRL	H	---		
CB 3D	SRL	L	---		
96	SUB	(HL)	SUB	M	Subtraktion von Akku
DD 96 05	SUB	(IX+d)	---		und Operand
FD 96 05	SUB	(IY+d)	---		Ergebnis im Akku
97	SUB	A	SUB	A	
90	SUB	B	SUB	B	
91	SUB	C	SUB	C	
92	SUB	D	SUB	D	
93	SUB	E	SUB	E	
94	SUB	H	SUB	H	
95	SUB	L	SUB	L	
D6 20	SUB	n	SUI	n	
AE	XOR	(HL)	XRA		Exklusiv-ODER von
DD AE 05	XOR	(IX+d)	---		Akku und Operand
FD AE 05	XOR	(IY+d)	---		Ergebnis im Akku
AF	XOR	A	XRA	A	
A8	XOR	B	XRA	B	
A9	XOR	C	XRA	C	
AA	XOR	D	XRA	D	
AB	XOR	E	XRA	E	
AC	XOR	H	XRA	H	
AD	XOR	L	XRA	L	
EE 20	XOR	n	XRI	n	

Für Konstanten verwendete Werte:

```
      nn  EQU   584H          |    n    EQU   20H
      d   EQU   5             |    e    EQU   30H
```

3. Übersicht über Assembler und Binder

Zur Syntax

- Unterstrichene Begriffe müssen durch einen aktuellen Wert ersetzt werden, z.B. "exp" durch "addr+1" oder "string" durch " 'FEHLER 3' ".
- In eckigen Klammern stehende Begriffe können ggf. weggelassen werden.
- Drei Punkte ("...") bedeuten, daß die Liste beliebig fortgesetzt werden kann.
- Alle sonstigen Begriffe oder Zeichen müssen direkt eingegeben werden. So sind z.B. die spitzen Klammern (< >) um die Argumente bei bedingten Pseudooperationen mit anzugeben. Groß- und Kleinschreibung sind im gesamten Quelltext gleichbedeutend.

Begriffe:

exp — numerischer Ausdruck
string — "zeichenkette" oder 'zeichenkette'
par — string oder exp
name — Zeichenkette aus A-Z 0-9 $. ? @ _;
Das erste Zeichen muß ein Buchstabe sein und nur die ersten 6 Zeichen sind signifikant.
delim — Begrenzerzeichen

3.1. Aufruf des Assemblers

ASM asmkommando
oder
ASM

Das ASM-Kommando kann auch erst nach Aufruf von ASM eingegeben werden. Dann werden allerdings nur noch große Buchstaben anerkannt.

ASM-Kommando

Ausführliche Schreibweise des ASM-Kommandos:

objektdatei,druckdatei=quelldatei[asmschalter,...]
Von der Quelldatei werden eine Druckdatei und eine Objektdatei erzeugt.

Verschiedene verkürzte Schreibweisen des ASM-Kommandos:

druckdatei=quelldatei[asmschalter,...]
Von Quelldatei wird nur Druckdatei erzeugt.

=quelldatei[asmschalter...]
Quelldatei wird in Objektdatei mit gleichem Namen übersetzt

,=quelldatei[asmschalter...]
Übersetzt Quelldatei, ohne daß Objektdatei oder Druckdatei angelegt wird.

Für Quelldatei wird der Typ "MAC", für Objektdatei der Typ "REL" und für Druckdatei der Typ "PRN" als Standard angenommen.

steht für druckdatei "TTY:" - Ausgabe auf Konsole (Bildschirm)
steht für druckdatei "LST:" - Ausgabe auf Listgerät (Drucker)
steht für quelldatei "TTY:" - Eingabe von Tastatur

ASM-Schalter

/M - Der mit DEFS (DS) definierte Bereich wird mit 00 beschrieben.
/X - setzt den Initialwert der Standardannahme für die Ausgabe von bedingten Blöcken bei nicht erfüllter Bedingung auf off (stfcond=off).
/O - In Druckdatei werden Adressen und Befehlskode oktal ausgegeben.
/H - In Druckdatei werden Adressen und Befehlskode hexadezimal ausgegeben.
/R - Erzeugung einer Objektkodedatei
/L - Erzeugung einer Druckdatei
/C - Erzeugung einer Symbolnachweisdatei (Cross-Referenz)
/Z - Übersetzung der Z80-Operationskodes
/I - Übersetzung der 8080-Operationskodes (Standard)
/P - Reservierung von 256 Byte zusätzlichen Stapels (wenn bei Standardstapel Überlauffehler auftreten)

Beispiele für ASM-Kommando:

A>asm bsp1,bsp2=bsp3
Die Datei "BSP1.MAC" wird übersetzt; es entstehen die Druckdatei "BSP2.PRN" und die Objektdatei "BSP3.REL"

B>a:asm =bsp1/l/m
Die Datei "BSP1.MAC" wird übersetzt; es entsteht die Objektdatei "BSP1.REL. Der L-Schalter bewirkt, daß außerdem eine Druckdatei "BSP1.PRN" erzeugt wird. Durch den M-Schalter werden die reservierten Speicherbereiche auf 00H gesetzt.

A>asm
*,TTY:=TTY:
Quelldaten werden von der Tastatur erwartet (1. und 2. Durchlauf beachten); die Druckdaten werden auf die Konsole ausgegeben.

3.2. Pseudooperationen zur Auswahl der Anweisungsliste

.Z80 - Z80 Mnemonik
.8080 - 8080-Mnemonik

3.3. Pseudooperationen zur Daten- und Symboldefinition

DB par[,par...]	definiere Byte
DEFB par[,par...]	wie DB
DEFM par[,par...]	wie DB
DC string[,string...]	wie DB mit Bit8 = 1 bei letztem Zeichen jedes strings
DS exp1[,exp2]	Reservierung Speicherbereich, exp1-Byte mit Byte exp2
DEFS exp1[,exp2]	wie DS
DW exp[,exp...]	definiere Wort (2 Byte-Größe)
DEFW exp[,exp...]	wie DW
name **EQU** exp	Zuweisung name:=exp; Wert für name darf nicht verändert werden
name **DEFL** exp	Zuweisung name:=exp; Wert für name kann verändert werden
name **ASET** exp	wie DEFL
name **SET** exp	wie DEFL, jedoch nicht für .Z80-Modus
EXT name[,name...]	extern definierte Namen
EXTRN name[,name...]	wie EXT
EXTERNAL name[,name...]	wie EXT
BYTE EXT name	extern definierte Bytes
BYTE EXTRN name	wie BYTE EXT
BYTE EXTERNAL name	wie BYTE EXT
ENTRY name[,name...]	global gültige Namen
GLOBAL name[,name...]	wie ENTRY
PUBLIC name[,name...]	wie ENTRY
opcode param,name##	extern definierter Name
name::opcode param,...	global gültiger Name

3.4. Pseudooperationen für Verschieben vor dem Laden

.PHASE exp	aktiviert den Phasenmodus und setzt den Phasenzeiger auf exp.
.DEPHASE	setzt den Phasenmodus außer Kraft

3.5. Pseudooperationen für den Zuordnungszähler

ASEG aktiviert den Zuordnungszähler für den absoluten Bereich

CSEG aktiviert den Zuordnungszähler für den programmrelativen Bereich

DSEG aktiviert den Zuordnungszähler für den datenrelativen Bereich

COMMON aktiviert den Zuordnungszähler für einen von mehreren Programmen benutzten Bereich

ORG exp ordnet dem aktivierten Zuordnungszähler exp zu.

3.6. Pseudooperationen zur Formatsteuerung

PAGE [exp] neue Ausgabeseite; exp = Seitengröße
***EJECT** [exp] wie PAGE
$EJECT [exp] wie PAGE

TITLE text text wird auf der ersten Zeile jeder Seite ausgegeben

3.7. Pseudooperationen zur Listensteuerung

.PRINTX delim text delim
text (einzeilig), einschließlich der Begrenzerzeichen delim, wird beim ersten und beim zweiten Durchlauf auf Konsole ausgegeben.
Beispiel:
.printx /falscher Parameter /

.LIST Es wird in Druckdatei ausgegeben (Standard).
.XLIST Ausgabe in Druckdatei wird unterdrückt.

.SFCOND FCOND-Flag := ON
.LFCOND FCOND-Flag := OFF
.TFCOND FCOND-Flag := /STFCOND
FCOND = ON - bedingte Blöcke ausgeben
FCOND = OFF - bedingte Blöcke nicht ausgeben
STFCOND - abhängig vom Schalter /X

.XALL Bei Makroaufruf werden nur die Zeilen ausgegeben, die Objektkode erzeugen (Standard).
.LALL Bei Makroaufruf werden alle Zeilen ausgegeben.
.SALL Bei Makroaufruf werden keine Zeilen ausgegeben.

.CREF Es wird in Symbolnachweisdatei ausgegeben (Standard).
.XCREF Ausgabe in Symbolnachweisdatei wird unterdrückt.

3.8. Pseudooperationen für bedingte Assemblierung

Der Anweisungsblock (bis ENDC, ENDIF oder ELSE) wird übersetzt, wenn:

IF exp	exp ungleich 0,
IFT exp	wie IF,
COND exp	wie IF,
IFE exp	exp gleich 0,
IFF exp	wie IFE,
IF1	1. Durchlauf,
IF2	2. Durchlauf,
IFDEF name	Symbol bereits definiert wurde,
IFNDEF name	Symbol noch nicht definiert wurde,
IFB <arg>	arg leer,
IFNB <arg>	arg nicht leer,
IFIDN <arg1>,<arg2>	wenn arg1 identisch arg2,
IFDIF <arg1>,<arg2>	wenn arg1 nicht identisch arg2,
ENDC	Ende bedingter Anweisungsblock,
ENDIF	wie ENDC
ELSE	Nachfolgender Anweisungsblock wird übersetzt, wenn obige Bedingung nicht erfüllt ist.

3.9. Operatoren

NUL [exp]	exp ist angegeben - 0000H exp ist nicht angegeben - 0FFFFH
TYPE exp	Der Operator liefert ein Statusbyte mit folgender Bedeutung: Bit 1,9 0 0 - absolut 0 1 - koderelativ 1 0 - datenrelativ 1 1 - commonrelativ Bit 7 1 - exp ist external 0 - exp ist lokal
LOW exp	niederwertige 8 Bit
HIGH exp	höherwertige 8 Bit
exp ***** exp	Multiplikation
exp **/** exp	Division
exp **MOD** exp	Modulo-Berechnung
exp1 **SHR** exp2	Rechtsverschiebung von exp1 um exp2 Bitpositionen
exp1 **SHL** exp2	Linksverschiebung von exp1 um exp2 Bitpositionen
- exp	negativer Wert
exp **+** exp	Addition
exp **-** exp	Subtraktion
exp **EQU** exp	gleich, ja - 0FFFFH, nein - 0000H
exp **NE** exp	ungleich, ja - 0FFFFH, nein - 0000H
exp **LT** exp	kleiner, ja - 0FFFFH, nein - 0000H
exp **LE** exp	kleiner gleich, ja - 0FFFFH, nein - 0000H
exp **GT** exp	größer, ja - 0FFFFH, nein - 0000H
exp **GE** exp	größer gleich, ja - 0FFFFH, nein - 0000H
exp **NOT** exp	Negation ja - 0FFFFH, nein - 0000H
exp **AND** exp	UND
exp **OR** exp	ODER
exp **XOR** exp	exklusiv ODER

3.10. Makros und Blockpseudooperationen

Begriffe:

dummy	Pseudoparameter
dummylist	Liste von dummys (durch Kommas getrennt)
arglist	Liste von Argumenten (durch Kommas getrennt)
parlist	Liste aktueller Parameter (durch Kommas getrennt)

name **MACRO** dummylist Beginn einer Makrodefinition

name parlist	Makroaufruf
REPT exp	Anweisungsblock wird exp-mal übersetzt
IRP dummy,<arglist>	Anweisungsblock wird für jedes Argument einmal übersetzt. Bei jeder Wiederholung wird das nächste Argument für dummy substituiert.
IRPC dummy,string	Anweisungsblock wird für jedes Zeichen der Zeichenkette einmal übersetzt. Bei jeder Wiederholung wird das nächste Zeichen für dummy substituiert.
ENDM	Ende Anweisungsblock; Anweisungsblöcke für REPT, IRP, IRPC sowie MACRO müssen mit ENDM abgeschlossen werden.
EXITM	Diese Pseudooperation ist notwendig, um eine REPT-, IRP- oder IRPC-Schleife "vorzeitig" zu verlassen.
LOCAL dummylist	Die in dummylist aufgeführten Symbole sind nur innerhalb der MACRO-Definition gültig.

3.11. Spezielle Makrooperatoren und -formen

&	um Texte oder Symbole zu verketten
;;	Ein Kommentar, welchem ";;" vorangestellt ist, belastet den Speicher nicht.
'	zur literalen Übersetzung des nächsten Zeichens; hiermit können z.B. Kommentare mit den Parametern übergeben werden.
%	wird lediglich in einem Makroargument verwendet. Es wird nicht der nachfolgende Name sondern eine Zeichenkette, die dem numerischen Wert des Namens entspricht, übergeben.

3.12. Sonstige Pseudooperationen

NAME ('modulname') definiert den Namen für den Modul

.COMMENT delim text delim
Kommentartext (auch mehrzeilig)

END [exp] Ende Programm ; exp = Startadresse

INCLUDE dateiname dateiname (Dateiname.Typ, große Buchstaben) wird in die aktuelle Quellkodedatei eingefügt

$INCLUDE dateiname wie INCLUDE

MACLIB dateiname wie INCLUDE

.RADIX exp Zahlenbasis für Konstanten (Standard dezimal)
$2 \leq$ exp ≤ 16 (Standard-Zahlenbasis für diese exp ist immer dezimal.)

.REQUEST dateiname[,dateiname...]
Anforderung an LINK, die Liste der dateiname nach externen Symbolen zu durchsuchen.

3.13 Fehlermeldungen

A - Fehler im Argument
C - Fehler in der Bedingungsschachtelung
E - Verwendung eine Externals, welches in diesem Zusammenhang nicht erlaubt ist
M - mehrfach definiertes Symbol
N - Fehler in einer Zahl
O - fehlerhafter Operationskode oder nicht einwandfreie Syntax
P - Phasenfehler
(Der Wert einer Marke oder eines EQU-Namens ist im zweiten Durchlauf unterschiedlich zum ersten Durchlauf.)
Q - fragwürdige Quelltextzeile
(z.B. Warnung, wenn Zeile nicht ordnungsgemäß abgeschlossen ist.)
R - nicht erlaubte Verwendung einer Verschiebung
U - undefiniertes Symbol
V - Wertfehler
(Wert ist im ersten Durchlauf noch nicht bekannt.)

3.14 Kennzeichen im ASM-Protokoll

' - koderelativ
" - datenrelativ
! - COMMON-relativ
space - absolut
***** - extern
C - Zeile aus INCLUDE-Datei
+ - Bestandteil einer MACRO-Erweiterung

3.15 Symboltabelle

U - undefiniertes Symbol
C - COMMON-Blockname
***** - externes Symbol
space - absoluter Wert
' - programmrelativer Wert
" - datenrelativer Wert
! - COMMON-relativer Wert
I - globales Symbol

3.16. LINK-Schalter

/R Rücksetzen

/E Rückkehr zum Betriebssystem; wenn /N gesetzt ist, wird die Zieldatei auf Diskette übertragen.

/E:name wie /E; Startadresse des Zielprogramms ist gleich dem Wert von name

/G Start des Programms; wenn /N gesetzt ist, wird zuvor die Zieldatei auf Diskette übertragen.

/G:name wie /G; Startadresse des Programms ist gleich dem Wert von name.

dateiname/N Name der bei /E oder /G zu sichernden Datei (Standardtyp ist .COM)

/P:adr Zeiger Programmbereich auf adr einstellen

/D:adr Zeiger Datenbereich auf adr einstellen

/U Ausgabe aller nichtdefinierten Symbole

/M Ausgabe aller definierter und nichtdefinierter Globals

dateiname/S Suchen in der Datei dateiname, um nicht definierte Globals aufzulösen

/X Sichern der Datei im ASCII-HEX-Format

/Y Erzeugen einer SYM-Datei (Symbol-Datei)

/H Zahlendarstellung hexadezimal (Standard)

/O Zahlendarstellung oktal

4. Der Debugger

4.1. Aufrufformate des Debuggers aus CCP

- Mit dem Auruf des Debuggers wird der vorherige Inhalt des TPA zerstört.
- Der Debugger lädt sich selbst an das obere Ende des TPA und reduziert den TPA um diesen Speicherraum.
- Sind im CCP-Kommando noch Parameter angegeben, werden außerdem die Debugger-Funktionen I und R automatisch auf diese Parameter angewendet.
- Mit dem Debugger-Kommando CTRL-C (Warmstart) wird der Debugger verlassen.

Aufruf: DU <CCP-PAR>

<CCP-PAR> ::=	ohne	keine Dateien automatisch nachladbar
	<edb1>	Nachladen Datendatei
	<edb1> <edb2>	Nachladen Daten- und Symboldatei
	* <edb2>	Nachladen Symboldatei
		<edb..> - eindeutige Dateibezeichnungen

4.2. Debugger-Kommandostruktur

\# Aufforderung des Debuggers an den Bediener zur Eingabe eines Debuggerkommandos
? Abbruchmeldung des Debuggers bei zurückgewiesenen Kommandos

Kommandoformat:
[-]<funktionskennzeichen>[W][<paramet>[,<paramet>[,<paramet>]]]

Debugger-Operanden: 16-Bit-Werte

(a) <hexzahl>	- Wert der Hexziffernfolge modulo 10000
(b) * <dezzahl>	- Wert der Dezimalziffernfolge modulo #65536
(c) '<ascii kette>'	- ASCII-Wert der letzten beiden Zeichen ('AB':=4142)
(d) ^.....^	- Stapelinhalt n-ten Niveaus
(e) * <cpu reg>	- Inhalt des CPU-Wortregisters
	<cpu reg> ::= B, D, H, B', D', H', S, P, X, Y
(f) !	- letzter eingegebener Parameterwert

- Symbole: bis zu 16-stellige alphanumerische Marken (keine Kleinbuchstaben)

(g) . <symbol>	- Referenzadresse
(h) @ <symbol>	- Wort-Inhalt der Referenzadresse
(i) = <symbol>	- Byte-Inhalt der Referenzadresse

- Symbolfolgen: s1/s2/.../sn
  Suche in der Reihenfolge s1,s2,...,sn und Referenz des so ermittelten sn

(j) . <sym folge> - Adresse der bedingten Referenz
(k) @ <sym folge> - Wort-Wert der bedingten Referenz
(l) = <sym folge> - Byte-Wert der bedingten Referenz

Parameter:
- Verknüpfung einer beliebigen Anzahl von Operanden durch Addition und Subtraktion
- Sonderfälle
  - Beginnt der Ausdruck mit "+", wird der letzte Parameterwert voreingestellt.
  - Beginnt der Ausdruck mit "-", wird der Wert Null voreingestellt.
  - Wird als Parameter nur ein Bytewert benutzt, wird der LOW-Teil des Parameterwertes herangezogen.

    z.B. Parameter: DE00+#127,+7 = DE7F,DE86
    DE00-300,-#512 = DB00,FE00

  - Ausnahmsweise ist als Parameter auch eine beliebig lange ASCII-Zeichenkette bei den Kommandos I und B zugelassen.

4.3. Kommandos

**** A ****
Fkt: Konsolen-Edition von Befehlsfolgen mnemonischer Darstellung im TPA

Aa Assemblieren ab a
A Assemblieren ab aktueller Adresse
-A entfernt Reassemblierungsmodul aus dem Debugger (die Kommandos A + L sind nicht mehr ausführbar, die Kommandos T + U nur eingeschränkt, Speichergewinn 1,5K)

**** C ****
Fkt: START eines System-Unterprogrammes

Cp Start des ab p stehenden Unterprogrammes (mit BC,DE=00)
Cp,c obiges mit BC = c, DE = 00
Cp,c,e obiges mit BC = c, DE = e

Bei Abarbeitung werden die Register der Test-CPU nicht verändert.

**** D ****
Fkt: Anzeige des Speicherinhaltes auf der Konsole in HEX- und ASCII-Darstellung

Da Anzeige ein Bild ab a
Da,b Anzeige Speicherinhalt von a bis b
D Anzeige ein Bild ab aktueller Adresse
D,b Anzeige ab aktueller Adresse bis b
DW... analoge Anzeige im Wortformat
-D... wie D... , aber ASCII-Darstellung unterdrückt

**** E ****
Fkt: Vergleich der Inhalte zweier Speicherbereiche (Blöcke)

E,a,b,c Vergleich der Speicherinhalte zweier gleichgroßer Blöcke, deren erster sich von a bis einschließlich b erstreckt und deren zweiter bei b beginnt

Bei unterschiedlicher Bytebelegung erfolgt die Protokollierung im Format

```
MMMM XX  NNNN YY
```

mit der Bedeutung:

```
MMMM   - Adresse im ersten Bereich
XX     - Bytebelegung der Speicheradresse MMMM
NNNN   - Adresse im zweiten Bereich
YY     - Belegung der zugeordneten Speicherstelle NNNN
```

**** F ****
Fkt: Belegung eines Speicherbereiches mit konstanten Werten

Fa,b,c Füllen des Bereiches von a bis einschließlich b mit Bytebelegung c

**** G ****
Fkt: Teststart für Echtzeitbetrieb

```
G       Testlauf ab aktuellem PC
Gp      Testlauf ab p
G,e     Testlauf ab aktuellem PC mit Setzen des Unterbrechungs-
        punktes e
Gp,e    Testlauf ab p mit Setzen des Unterbrechungspunktes e
G,e,f   Testlauf ab aktuellem PC mit Setzen von e oder f
-G...   wie oben, aber Durchgänge durch Paßpunkte werden gezählt,
        aber nicht protokolliert (außer dem letzten)
```

Abbruch des Testlaufes und Rückkehr in den Debuggermode über RST 7 (Unterbrechungs- und Paßpunkte)

**** H ****
Fkt: Hilfsfunktionen

```
Ha,b    Berechnung von Summe a+b und Differenz a-b
          Anzeigeformat: <summe>  <diff>
Ha      Konvertierungen von a
          Anzeigeformat: HHHH  #DDDD  'C'  .SSSSSSSS
          Mit den Bedeutungen:
          HHHH      Hexwert
          DDDD      Dezimalwert
          C         zugeordnetes abbildbares ASCII-Zeichen
          SSSSS     zugeordnetes Symbol der Symboltabelle
-Ha     Konvertierung mit Unterdrückung der Symboldarstellung
H       Anzeige aller Referenzen der Symboltabelle
          Anzeigeformat: HHHH  SSS...S
          mit den Bedeutungen:
          HHHH      Hexwert
          SSS...S   zugeordnetes Symbol
```

**** I ****

Fkt: Eintragung der CCP-Parameter in den Verständigunsbereich

Isss...s	sss...s (max. 63) wird ab 80H eingetragen: 80H +0 +1 +2 ... +N +N+1 N s1 s2 ... sN 00 sowie FCB1 (ab 5CH) bzw. FCB2 (ab 6CH) vorbereitet

Zur Ausführung des R-Kommandos müssen immer FCB1 und FCB2 entsprechend vorbereitet sein.

**** K ****

Fkt: Suchen einer vorgegebenen Bytekette im Speicher

Ka,b,c Suche einer Bytekette, die innerhalb des Bereiches von a bis einschließlich b beginnt und ab c definiert ist.

Kettendefinition:	c+0 +1 +2 ... +N N k1 k2 ... kN
Protokollformat:	AAAA BBBB CCCC .....
AAAA BBBB ...	Adressen, auf denen äquivalente gefundene Ketten beginnen

**** L ****

Fkt: Listen des disassemblierten Maschinenprogramms

La	Listen ein Konsolenbild des disassemblierten Programmes ab a
La,b	Listen der Disassemblierung von a bis einschließlich b (b ist bytegenau anzugeben)
L	Listen ein Konsolenbild ab aktueller Adresse
-L...	Unterdrückung der Symboldarstellung beim Listen

**** M ****

Fkt: Blocktransport innerhalb des Speichers

Ms,h,d Kopieren des Bereiches von s bis einschließlich h in den Bereich ab d in aufsteigender Adressenfolge

**** P ****

Fkt: Setzen, Löschen und Anzeigen von Paßpunkten

P	Anzeige aller gesetzten Paßpunkte
Pa,n	Paßpunkt auf Adresse a mit Paßzählerwert n (max. 255) setzen
Pa	Paßpunkt auf Adresse a setzen (mit n=1)
-Pa	den Paßpunkt auf Adresse a löschen
-P	alle Paßpunkte löschen

Im Echtzeitbetrieb wird nach Durchlauf eines Paßpunktes der CPU-Zustand protokolliert und der Paßzähler dekrementiert. Bei Erreichen von n=0 wird n=1 gesetzt und der Echtzeitlauf beendet, anderenfalls wird der Testlauf fortgesetzt.

**** R ****

Fkt: Einlesen von Dateien in den TPA

R	Einlesen der mit Kommando I in FCB1 und FCB2 spezifizierten Dateien
Rv	Einlesen der in FCB1 spezifizierten Datei mit Verschiebung v gegenüber 100H (TPA-Beginn)

x.HEX	im FCB1 wird als Datei im HEX-Fomat geladen.
x.UTL	im FCB1 wird als Utility geladen und aktiviert (muß vor der Symboltabelle geladen werden, reduziert den TPA, überschreibt den TPA).
x.y	im FCB2 wird in die Symboltabelle eingetragen (Laden zerstört den TPA nicht; reduziert den TPA).

**** S ****

Fkt: byteweise Belegung des Speichers

Sa	ein Byte oder "Zeichenkette ab Adresse a eintragen
SWa	Doppelbyte (Datenwort) oder "Zeichenkette ab Adresse a eintragen
.	Beenden der Kommandoausführung

Die Speicheradresse wird entsprechend Byte, Wort oder Zeichenkette automatisch mit verwaltet.

```
Protokollformat:    AAAA  BB  CCCC
          oder      AAAA  BB  "KK...K
          oder      DDDD  EEEE  FFFF
          oder      DDDD  EEEE  "KK...K
          mit Bedeutung:
          AAAA      Byte-Adresse
          BB        alte Belegung von AAAA
          CCCC      Eingabe neue Byte-Belegung
          DDDD      Wort-Adresse
          EEEE      alte Wort-Belegung
          FFFF      Eingabe neue Wort-Belegung
          KK...K    ASCII-Zeichenkette
```

**** T ****

Fkt: schrittweise Abarbeitung des Programmes mit Protokoll der Test-CPU in jedem Testschritt

Tn	Step-Test n Befehlsschritte
T	Test einen Befehl (n=1)
-T...	Protokoll ohne Symbolreferenz
TW...	wie T...; Unterprogrammaufruf wird als ein Programmschritt abgearbeitet

**** U ****

Fkt: schrittweise Abarbeitung des Programmes ohne Protokollierung jedes Schrittes

Un	Test n Befehlsschritte
UWn	Test n Befehlsschritte, UP-Aufrufe werden als ein Schritt bearbeitet
-U	wie U, aber ohne Symbolreferenzen

**** X ****
Fkt: Modifikation der Test-CPU

X	Anzeige des Zustandes der Test-CPU
Xf	Änderung eines Flags der Test-CPU (f=C,Z,M,E,I,C',Z',M',E',I')
Xr	Änderung eines Registers (r=A,A') bzw. eines Registerpaares (r = B,B',D,D',H,H',S,P,X,Y) der Test-CPU

Erläuterungen:

- Parameter werden durch Komma oder Leerzeichen getrennt.
- Jeder Befehlsablauf kann durch eine beliebige Taste abgebrochen werden

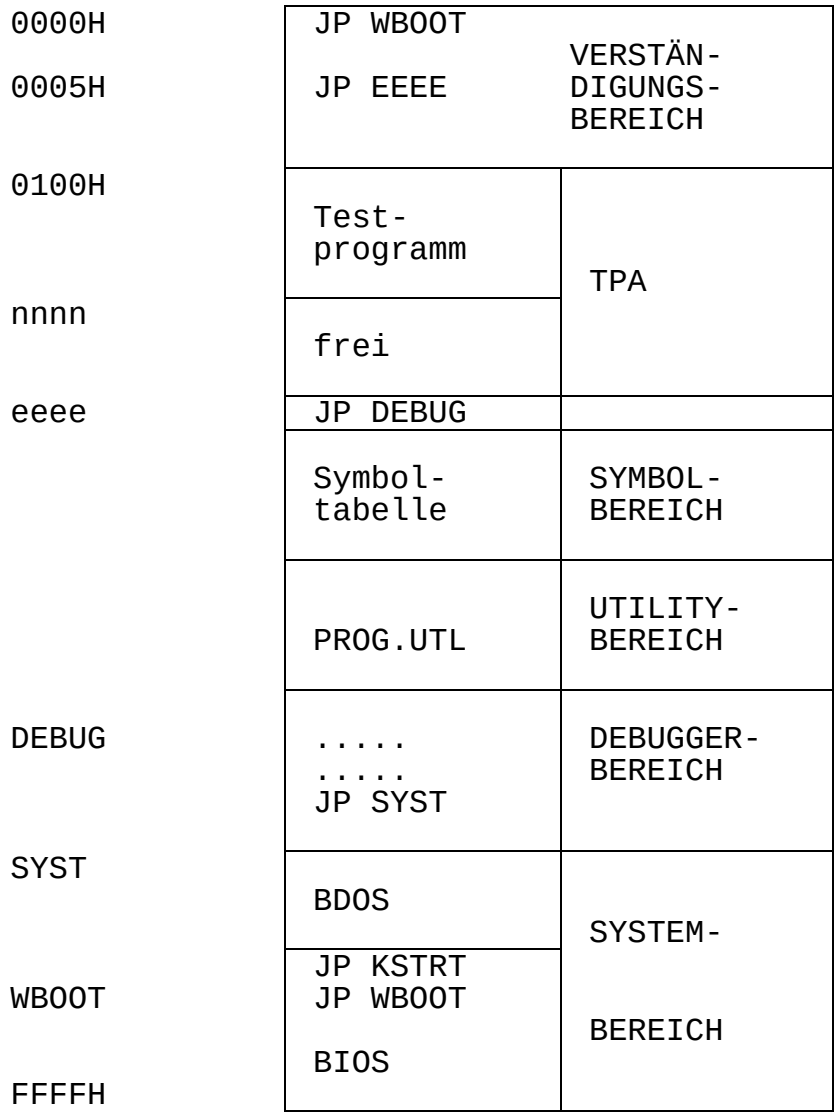
4.4. Debugger - Utilities

- Aktivierung erfolgt über I-R-Kommandofolge bzw. Debuggeraufruf
- reduziert den TPA
- wird nach dem Debugger und vor der Symboltabelle geladen
- immer nur letzte Utility über Symbole ansprechbar

TRACE	- Sammlung der Adressen der letzten 128 Befehle für Backtrace
HIST	- Erzeugung eines Histogrammes über Befehlshäufigkeit im vorgewählten Bereich

C.INITIAL	- Initialisierung bzw. Reinitialisierung von TRACE bzw. HIST
Tn,.COLLECT	- Datensammlung TRACE bzw. HIST im T-Kommando
Un,.COLLECT	- Datensammlung TRACE bzw. HIST im U-Kommando
C.DISPLAY	- Protokoll von Backtrace bzw. Histogramm

4.5. Belegung des Arbeitsspeichers



Druck: VEB Dienstleistungsbetrieb Erfurt V/4/56 R1 01/86

robotron

VEB Robotron Büromaschinenwerk
»Ernst Thälmann« Sömmerda
Weißenseer Straße 52
Sömmerda
DDR – 5230

Exporteur:
Robotron Export–Import
Volkseigener
Außenhandelsbetrieb
der Deutschen
Demokratischen Republik
Allee der Kosmonauten 24
Berlin
DDR – 1140