ANHANG B-1

ZUSAMMENFASSUNG DER ANWEISUNG DER 8080 ASSEMBLERSPRACHE

Th s appendix provides a summary of 8080 assembly language instructions. Abbreviations used are as follows:

A	The accumulator (register A)
A_n	Bit n of the accumulator contents, where n may have any value from 0 to 7 and 0 is the least significant (rightmost) bit
ADDR	Any memory address
Aux. carry	The auxiliary carry bit
Carry	The carry bit
CODE	An operation code
DATA	8 bits (one byte) of data
DATA16	16 bits (2 bytes) of data
DST	Destination register or memory byte
EXP	A constant or mathematical expression
INTE	The 8080 interrupt enable flip-flop
LABEL:	Any instruction label
M	A memory byte
Parity	The parity bit
PC	Program Counter
PCH	The most significant 8 bits of the program counter
PCL	The least significant 8 bits of the program counter
REGM	Any register or memory byte

REGPR	A register pair. Legal register pair symbols are: B for registers B and C D for registers D and E H for registers H and L SP for the 16 bit stack pointer PSW for register A and flag bits
RP1	The first register of register pair RP
RP2	The second register of register pair RP
Sign	The sign bit
SP	The 16-bit stack pointer register
SRC	Source register or memory byte
Zero	The zero bit
XY	The value obtained by concatenating the values X and Y
[ ]	An optional field enclosed by brackets
( )	Contents of register or memory byte enclosed by parentheses
←	Replace value on lefthand side of arrow with value on righthand side of arrow
$	Present contents of program counter

CARRY BIT INSTRUCTIONS

Format:

```
        [LABEL:]                    CODE
```

CODE	DESCRIPTION	
STC	(Carry) ← 1	Set carry
CMC	(Carry) ← (Carry)	Complement carry

Condition bits affected: Carry

SINGLE REGISTER INSTRUCTIONS

Format:

```
        [LABEL:]        INR         REGM
                        --or--
        [LABEL:]        DCR         REGM
                        --or--
        [LABEL:]        CMA
                        --or--
        [LABEL:]        DAA
```

Rev. C

CODE	DESCRIPTION	
INR	(REGM) ← (REGM)+1	Increment register REGM
DCR	(REGM) ← (REGM)-1	Decrement register REGM
CMA	(A) ← $(\bar{A})$	Complement accumulator
DAA	If $(A_0-A_3) > 9$ or (Aux.Carry)=1, (A) ← (A)+6 Then if $(A_4-A_7) > 9$ or (Carry)= 1 $(A) = (A) + 6 \cdot 2^4$	Convert accumulator contents to form two decimal digits

Condition bits affected:			
	INR,DCR	:	Zero, sign, parity, aux. carry
	CMA	:	None
	DAA	:	Zero, sign, parity, carry, aux. carry

NOP INSTRUCTION

Format:

[LABEL:] NOP

CODE	DESCRIPTION
NOP	No operation

Condition bits affected: None

DATA TRANSFER INSTRUCTIONS

Format:

[LABEL:]	MOV	DST,SRC
	–or–	
[LABEL:]	CODE	REGPR

NOTE: SRC and DST not both = M

NOTE: RP = B or D

CODE	DESCRIPTION	
MOV	(DST) ← (SRC)	Load register DST from register SRC
STAX	((REGPR)) ← (A)	Store accumulator at memory location referenced by the specified register pair
LDAX	(A) ← ((REGPR))	Load accumulator from memory location referenced by the specified register pair

Condition bits affected: None

REGISTER OR MEMORY TO ACCUMULATOR INSTRUCTIONS

Format

[LABEL:] CODE REGM

CODE	DESCRIPTION	
ADD	(A) ← (A)+(REGM)	Add REGM to accumulator
ADC	(A) ← (A)+(REGM)+(Carry)	Add REGM to accumulator with carry
SUB	(A) ← (A)-(REGM)	Subtract REGM from accumulator
SBB	(A) ← (A)-(REGM)-(Carry)	Subtract REGM from accumulator with borrow
ANA	(A) ← (A) AND (REGM)	AND accumulator with REGM
XRA	(A) ← (A) XOR (REGM)	EXCLUSIVE-ORaccumulator with REGM
ORA	(A) ← (A) OR (REGM)	OR accumulator with REGM
CMP	Condition bits set by (A)-(REGM)	Compare REGM with accumulator

Condition bits affected:

ADD, ADC, SUB, SBB: Carry, Sign, Zero, Parity, Aux. Carry
ANA, XRA, ORA: Sign, Zero, Parity. Carry and Aux. Carry are reset.
CMP: Carry, Sign, Zero, Parity, Aux. Carry. Zero set if (A)=(REGM)
Zero reset if (A) $\neq$ (REGM)
Carry set if (A) < (REGM)
Carry reset if (A) $\geqslant$ (REGM)
Note: CMP treats (A) and (REGM) as unsigned 8-bit quantities.

ROTATE ACCUMULATOR INSTRUCTIONS

Format:

[LABEL:] CODE

CODE	DESCRIPTION	
RLC	(Carry) ← A_7, A_{n+1} ← A_n, A_0 ← A_7	Set Carry = A_7, rotate accumulator left
RRC	(Carry) ← A_0, A_n ← A_{n+1}, A_7 ← A_0	Set Carry = A_0, rotate accumulator right
RAL	A_{n+1} ← A_n, (Carry) ← A_7, A_0 ← (Carry)	Rotate accumulator left through the Carry
RAR	A_n ← A_{n+1}, (Carry) ← A_0, A_7 ← (Carry)	Rotate accumulator right through Carry

Condition bits affected: Carry

Rev. C

REGISTER PAIR INSTRUCTIONS

Format:

```
        [LABEL:]        CODE1        REGPR
                        —or—
        [LABEL:]        CODE2
```

NOTE: For PUSH and POP, REGPR=B, D, H, or PSW
For DAD, INX, and DCX, REGPR=B, D, H, or SP

CODE1	DESCRIPTION	
PUSH	((SP)-1) ← (REGPR1), ((SP)-2) ← (REGPR2), (SP) ← (SP)-2	Save REGPR on the stack REGPR=PSW saves accumulator and condition bits
POP	(REGPR1) ← ((SP)+1), (REGPR2) ← ((SP)), (SP) ← (SP)+2	Restore REGPR from the stack REGPR=PSW restores accumulator and condition bits
DAD	(HL) ← (HL) + (REGPR)	Add REGPR to the 16-bit number in H and L
INX	(REGPR) ← (REGPR)+1	Increment REGPR by 1
DCX	(REGPR) ← (REGPR)−1	Decrement REGPR by 1
CODE2	**DESCRIPTION**	
XCHG	(H) ⟷ (D), (L) ⟷ (E)	Exchange the 16 bit number in H and L with that in D and E
XTHL	(L) ⟷ ((SP)), (H) ⟷ ((SP)+1)	Exchange the last values saved in the stack with H and L
SPHL	(SP) ← (H):(L)	Load stack pointer from H and L

Condition bits affected:

PUSH, INX, DCX, XCHG, XTHL, SPHL: None
POP : If REGPR=PSW, all condition bits are restored from the stack, otherwise none are affected.
DAD : Carry

IMMEDIATE INSTRUCTIONS

Format:

```
        [LABEL:]        LXI          REGPR, DATA16
                        —or—
        [LABEL:]        MVI          REGM, DATA
                        —or—
        [LABEL:]        CODE         REGM
```

NOTE: REGPR=B, D, H, or SP

Rev. C

CODE	DESCRIPTION	
LXI	(REGPR) ← DATA 16	Move 16-bit immediate Data into REGPR
MVI	(REGM) ← DATA	Move immediate DATA into REGM
ADI	(A) ← (A) + DATA	Add immediate data to accumulator
ACI	(A) ← (A) + DATA + (Carry)	Add immediate data to accumulator with Carry
SUI	(A) ← (A) - DATA	Subtract immediate data from accumulator
SBI	(A) ← (A) - DATA - (Carry)	Subtract immediate data from accumulator with borrow
ANI	(A) ← (A) AND DATA	AND accumulator with immediate data
XRI	(A) ← (A) XOR DATA	EXCLUSIVE-OR accumulator with immediate data
ORI	(A) ← (A) OR DATA	OR accumulator with immediate data
CPI	Condition bits set by (A)-DATA	Compare immediate data with accumulator

Condition bits affected:

LXI, MVI: None
ADI, ACI, SUI, SBI: Carry, Sign, Zero, Parity, Aux. Carry
ANI, XRI, ORI: Zero, Sign, Parity. Carry and Aux. Carry are reset.
CPI: Carry, Sign, Zero, Parity, Aux. Carry. Zero set if (A) = DATA
Zero reset if (A) ≠ DATA
Carry set if (A) < DATA
Carry reset if (A) ≥ DATA
Note: CPI treats (A) and DATA as unsigned 8-bit quantities.

DIRECT ADDRESSING INSTRUCTIONS

Format:

[LABEL:] CODE ADDR

CODE	DESCRIPTION	
STA	(ADDR) ← (A)	Store accumulator at location ADDR
LDA	(A) ← (ADDR)	Load accumulator from location ADDR
SHLD	(ADDR) ← (L), (ADDR+1) ← (H)	Store L and H at ADDR and ADDR+1
LHLD	(L) ← (ADDR), (H) ← (ADDR+l)	Load L and H from ADDR and ADDR+1

Condition bits affected: None

JUMP INSTRUCTIONS

Format:

[LABEL:] PCHL
—or—
[LABEL:] CODE ADDR

Rev. C

CODE	DESCRIPTION	
PCHL	(PC) ← (HL)	Jump to location specified by register H and L
JMP	(PC) ← ADDR	Jump to location ADDR
JC	If (Carry) = 1, (PC) ← ADDR If (Carry) = 0, (PC) ← (PC)+3	Jump to ADDR if Carry set
JNC	If (Carry) = 0, (PC) ← ADDR If (Carry) = 1, (PC) ← (PC)+3	Jump to ADDR if Carry reset
JZ	If (Zero) = 1, (PC) ← ADDR If (Zero) = 0, (PC) ← (PC)+3	Jump to ADDR if Zero set
JNZ	If (Zero) = 0, (PC) ← ADDR If (Zero) = 1, (PC) ← (PC)+3	Jump to ADDR if Zero reset
JP	If (Sign) = 0, (PC) ← ADDR If (Sign) = 1, (PC) ← (PC)+3	Jump to ADDR if plus
JM	If (Sign) = 1, (PC) ← ADDR If (Sign) = 0, (PC) ← (PC)+3	Jump to ADDR if minus
JPE	If (Parity) = 1, (PC) ← ADDR If (Parity) = 0, (PC) ← (PC)+3	Jump to ADDR if parity even
JPO	If (Parity) = 0, (PC) ← ADDR If (Parity) = 1, (PC) ← (PC)+3	Jump to ADDR if parity odd

Condition l its affected: None

CALL INSTRUCTIONS

Format:

[LABEL:] CODE ADDR

CODE	DESCRIPTION
CALL	((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR Call subroutine and push return address onto stack
CC	If (Carry) = 1, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR If (Carry) = 0, (PC) ← (PC)+3 Call subroutine if Carry set
CNC	If (Carry) = 0, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR If (Carry) = 1, (PC) ← (PC)+3 Call subroutine if Carry reset
CZ	If (Zero) = 1, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC ← ADDR If (Zero) = 0, (PC) ←(PC)+3 Call subroutine if Zero set
CNZ	If (Zero) = 0, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC ← ADDR If (Zero) = 1, (PC) ← (PC)+3 Call subroutine if Zero reset
CP	If (Sign) = 0, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR If (Sign) = 1, (PC) ← (PC)+3 Call subroutine if Sign plus
CM	If (Sign) = 1, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR If (Sign) = 0, (PC) ← (PC)+3 Call subroutine if Sign minus
CPE	If (Parity) = 1, ((SP) -1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR If (Parity) = 0, (PC) ← (PC)+3 Call subroutine if Parity even
CPO	If (Parity) = 0, ((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2, (PC) ← ADDR If (Parity) = 1, (PC) ← (PC)+3 Call subroutine if Parity odd

Condition bits a fected: None

Rev. C

RETURN INSTRUCTIONS

Format:

[LABEL:] CODE

CODE	DESCRIPTION
RET	(PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 Return from subroutine
RC	If (Carry) = 1, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP) +2 If (Carry) = 0, (PC) ← (PC)+1 Return if Carry set
RNC	If (Carry) = 0, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 If (Carry) = 1, (PC) ← (PC)+1 Return if Carry reset
RZ	If (Zero) = 1, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 If (Zero) = 0, (PC) ← (PC)+1 Return if Zero set
RNZ	If (Zero) = 0, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP) ← (SP)+2 If (Zero) = 1, (PC) ← (PC)+1 Return if Zero reset
RM	If (Sign) = 1, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 If (Sign) = 0, (PC) ← (PC)+1 Return if minus
RP	If (Sign) = 0, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 If (Sign) = 1, (PC) ← (PC)+1 Return if plus
RPE	If (Parity) = 1, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 If (Parity) = 0, (PC) ← (PC)+1 Return if parity even
RPO	If (Parity) = 0, (PCL) ← ((SP)), (PCH) ← ((SP)+1), (SP) ← (SP)+2 If (Parity) = 1, (PC) ← (PC)+1 Return if parity odd

Condition bits affected: None

RST INSTRUCTION

Format:

[LABEL:] RST EXP

NOTE: 000B ≤ EXP ≤ 111B

CODE	DESCRIPTION
RST	((SP)-1) ← (PCH), ((SP)-2) ← (PCL), (SP) ← (SP) -2 (PC) ← 0000000000EXP000B Call subroutine at address specified by EXP

Condition bits affected: None

INTERRUPT FLIP-FLOP INSTRUCTIONS

Format.

[LABEL:] CODE

CODE	DESCRIPTION	
EI	(INTE) ← 1	Enable the interrupt system
DI	(INTE) ← 0	Disable the interrupt system

Condition bits affected: None

INPUT/OUTPUT INSTRUCTIONS

Format:

[LABEL:] CODE COMMAND

CODE	DESCRIPTION	
IN	(A) — input device	Read a byte from device into the accumulator
OUT	output device—COMMAND, (A)	Send the COMMAND and the accumulator contents to device

Condition bits affected: None

HLT INSTRUCTION

Format:

[LABEL:] HLT

CODE	DESCRIPTION	
HLT	—————————	Instruction execution halts until an interrupt occurs

Condition bits affected: None

PSEUDO – INSTRUCTIONS

ORG PSEUDO – INSTRUCTION

Format:

ORG EXP

CODE	DESCRIPTION	
ORG	LOCATION COUNTER ← EXP	Set Assembler location counter to EXP

EQU PSEUDO – INSTRUCTION

Format:

NAME EQU EXP

CODE	DESCRIPTION	
EQU	NAME ← EXP	Assign the value EXP to the symbol NAME

SET PSEUDO – INSTRUCTION

Format:

NAME SET EXP

CODE	DESCRIPTION	
SET	NAME ← EXP	Assign the value EXP to the symbol NAME, which may have been previously SET.

END PSEUDO – INSTRUCTION

Format:

```
        END
```

CODE	DESCRIPTION
END	End the assembly

CONDITIONAL ASSEMBLY PSEUDO – INSTRUCTIONS

Format:

```
        IF                        EXP
                    –and–
                    ENDIF
```

CODE	DESCRIPTION
IF	If EXP = 0, ignore assembler statements until ENDIF is reached. Otherwise, continue assembling statements
ENDIF	End range of preceding IF

MACRO DEFINITION PSEUDO – INSTRUCTIONS

Format:

```
        NAME            MACRO         LIST
                        –and–
                        ENDM
```

CODE	DESCRIPTION
MACRO	Define a macro named NAME with parameters LIST
ENDM	End Macro definition

TITLE PSEUDO – INSTRUCTION

Format:

```
        TITLE           'STRING'
```

CODE	DESCRIPTION
TITLE	Define 'title' to appear beneath page header.

Rev. C

ANHANG B-2

ANWEISUNG AUSFÜHRUNGSDAUER BIT-STRUKTUREN UND OPERATIONSCODE

This appendix summarizes the bit patterns and number of time states associated with every 8080 CPU instruction.

The instructions are listed in both mnemonic (alphabetical) and operation code (numerical) sequence.

When using this summary, note the following symbology:

1) DDD represents a destination register. SSS represents a source register. Both DDD and SSS are interpreted as follows:

DDD or SSS	Interpretation
000	Register B
001	Register C
010	Register D
011	Register E
100	Register H
101	Register L
110	A memory register
111	The accumulator

2) Instruction execution time equals number of time periods multiplied by the duration of a time period.

A time period may vary from 480 nanosecs to 2 μsec.

Where two numbers of time periods are shown (eq. 5/11), it means that the smaller number of time periods will be required if a condition is not met, and the larger number of time periods will be required if the condition is met.

MNEMONIC	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	NUMBER OF TIME PERIODS
CALL	1	1	0	0	1	1	0	1	17
CC.	1	1	0	1	1	1	0	0	11/17
CNC	1	1	0	1	0	1	0	0	11/17
CZ	1	1	0	0	1	1	0	0	11/17
CNZ	1	1	0	0	0	1	0	0	11/17
CP	1	1	1	1	0	1	0	0	11/17
CM	1	1	1	1	1	1	0	0	11/17
CPE	1	1	1	0	1	1	0	0	11/17
CPO	1	1	1	0	0	1	0	0	11/17
RET	1	1	0	0	1	0	0	1	10
RC	1	1	0	1	1	0	0	0	5/11
RNC	1	1	0	1	0	0	0	0	5/11
RZ	1	1	0	0	1	0	0	0	5/11
RNZ	1	1	0	0	0	0	0	0	5/11
RP	1	1	1	1	0	0	0	0	5/11
RM	1	1	1	1	1	0	0	0	5/11
RPE	1	1	1	0	1	0	0	0	5/11
RPO	1	1	1	0	0	0	0	0	5/11

Rev. C

MNEMONIC	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	NUMBER OF TIME PERIODS
RST	1	1	A	A	A	1	1	1	11
IN	1	1	0	1	1	0	1	1	10
OUT	1	1	0	1	0	0	1	1	10
LXI B	0	0	0	0	0	0	0	1	10
LXI D	0	0	0	1	0	0	0	1	10
LXI H	0	0	1	0	0	0	0	1	10
LXI SP	0	0	1	1	0	0	0	1	10
PUSH B	1	1	0	0	0	1	0	1	11
PUSH D	1	1	0	1	0	1	0	1	11
PUSH H	1	1	1	0	0	1	0	1	11
PUSH PSW	1	1	1	1	0	0	0	1	11
POP B	1	1	0	0	0	0	0	1	10
POP D	1	1	0	1	0	0	0	1	10
POP H	1	1	1	0	0	0	0	1	10
POP PSW	1	1	1	1	0	0	0	1	10
STA	0	0	1	1	0	0	1	0	13
LDA	0	0	1	1	1	0	1	0	13
XCHG	1	1	1	0	1	0	1	1	4
XTHL	1	1	1	0	0	0	1	1	18
SPHL	1	1	1	1	1	0	0	1	5
PCHL	1	1	1	0	1	0	0	1	5
DAD B	0	0	0	0	1	0	0	1	10
DAD D	0	0	0	1	1	0	0	1	10
DAD H	0	0	1	0	1	0	0	1	10
DAD SP	0	0	1	1	1	0	0	1	10
STAX B	0	0	0	0	0	0	1	0	7
STAX D	0	0	0	1	0	0	1	0	7
LDAX B	0	0	0	0	1	0	1	0	7
LDAX D	0	0	0	1	1	0	1	0	7
INX B	0	0	0	0	0	0	1	1	5
INX D	0	0	0	1	0	0	1	1	5
INX H	0	0	1	0	0	0	1	1	5
INX SP	0	0	1	1	0	0	1	1	5
MOV r_1, r_2	0	1	D	D	D	S	S	S	5
MOV M, r	0	1	1	1	0	S	S	S	7
MOV r, M	0	1	D	D	D	1	1	0	7
HLT	0	1	1	1	0	1	1	0	7
MVI r	0	0	D	D	D	1	1	0	7
MVI M	0	0	1	1	0	1	1	0	10
INR	0	0	D	D	D	1	0	0	5
DCR	0	0	D	D	D	1	0	1	5
INR A	0	0	1	1	1	1	0	0	5
DCR A	0	0	1	1	1	1	0	1	5
INR M	0	0	1	1	0	1	0	0	10
DCR M	0	0	1	1	0	1	0	1	10
ADD r	1	0	0	0	0	S	S	S	4
ADC r	1	0	0	0	1	S	S	S	4
SUB r	1	0	0	1	0	S	S	S	4
SBB r	1	0	0	1	1	S	S	S	4
AND r	1	0	1	0	0	S	S	S	4
XRA r	1	0	1	0	1	S	S	S	4
ORA r	1	0	1	1	0	S	S	S	4
CMP r	1	0	1	1	1	S	S	S	4
ADD M	1	0	0	0	0	1	1	0	7
ADC M	1	0	0	0	1	1	1	0	7

Rev. B

MNEMONIC	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	NUMBER OF TIME PERIODS
SUB M	1	0	0	1	0	1	1	0	7
SBB M	1	0	0	1	1	1	1	0	7
AND M	1	0	1	0	0	1	1	0	7
XRA M	1	0	1	0	1	1	1	0	7
ORA M	1	0	1	1	0	1	1	0	7
CMP M	1	0	1	1	1	1	1	0	7
ADI	1	1	0	0	0	1	1	0	7
ACI	1	1	0	0	1	1	1	0	7
SUI	1	1	0	1	0	1	1	0	7
SBI	1	1	0	1	1	1	1	0	7
ANI	1	1	1	0	0	1	1	0	7
XRI	1	1	1	0	1	1	1	0	7
ORI	1	1	1	1	0	1	1	0	7
CPI	1	1	1	1	1	1	1	0	7
RLC	0	0	0	0	0	1	1	1	4
RRC	0	0	0	0	1	1	1	1	4
RAL	0	0	0	1	0	1	1	1	4
RAR	0	0	0	1	1	1	1	1	4
JMP	1	1	0	0	0	0	1	1	10
JC	1	1	0	1	1	0	1	0	10
JNC	1	1	0	1	0	0	1	0	10
JZ	1	1	0	0	1	0	1	0	10
JNZ	1	1	0	0	0	0	1	0	10
JP	1	1	1	1	0	0	1	0	10
JM	1	1	1	1	1	0	1	0	10
JPE	1	1	1	0	1	0	1	0	10
JPO	1	1	1	0	0	0	1	0	10
DCX B	0	0	0	0	1	0	1	1	5
DCX D	0	0	0	1	1	0	1	1	5
DCX H	0	0	1	0	1	0	1	1	5
DCX SP	0	0	1	1	1	0	1	1	5
CMA	0	0	1	0	1	1	1	1	4
STC	0	0	1	1	0	1	1	1	4
CMC	0	0	1	1	1	1	1	1	4
DAA	0	0	1	0	0	1	1	1	4
SHLD	0	0	1	0	0	0	1	0	16
LHLD	0	0	1	0	1	0	1	0	16
EI	1	1	1	1	1	0	1	1	4
DI	1	1	1	1	0	0	1	1	4
NOP	0	0	0	0	0	0	0	0	4

ANHANG B-3

ANWEISUNGEN AN DIE ZE 8080 IN OPERATIONSCODE-FOLGE

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	- - -
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	- - -	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	- - -
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	- - -	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	- - -	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	- - -	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	- - -
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	- - -	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	- - -	F6	ORI D8
20	- - -	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	- - -
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	- - -	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

Adr = 16-bit address.