Codierte ALU

ITT

Hex Nr.	$C_3\ C_2\ C_1\ C_0$	Funktion
0	0 0 0 0	A
1	0 0 0 1	1
2	0 0 1 0	$\bar{A}$
3	0 0 1 1	B
4	0 1 0 0	0
5	0 1 0 1	A + 1
6	0 1 1 0	A – 1
7	0 1 1 1	A + B
8	1 0 0 0	A – B
9	1 0 0 1	A ∧ B
A	1 0 1 0	A ∨ B
B	1 0 1 1	A ⊻ B
C	1 1 0 0	–1
D	1 1 0 1	nicht verwendet
E	1 1 1 0	
F	1 1 1 1	

A = Daten der Schalter A_7 bis A_0

B = Daten der Schalter B_7 bis B_0

2 Akkumulator

ITT

Hex Nr.	$C_3\,C_2\,C_1\,C_0$	Funktion	Inhalt des Akkus nach Takt
0	0 0 0 0	NOP	A
1	0 0 0 1	SP1	+1
2	0 0 1 0	CMA	$\bar{A}$
3	0 0 1 1	LDA	B
4	0 1 0 0	CLA	0
5	0 1 0 1	INC	A + 1
6	0 1 1 0	DEC	A − 1
7	0 1 1 1	ADD	A + B
8	1 0 0 0	SUB	A − B
9	1 0 0 1	AND	A ∧ B
A	1 0 1 0	IOR	A v B
B	1 0 1 1	XOR	A ⊻ B
C	1 1 0 0	SM1	−1
D	1 1 0 1	nicht verwendet	
E	1 1 1 0		
F	1 1 1 1		

A = Inhalt des Akkus

B = Daten der Schalter B_7 bis B_0

3

Akku mit Speicher

ITT

Hex Nr.	Op-Code	Adresse	Funktion	Akku-Inh.
0	0 0 0 0	x x x x	NOP	A
1	0 0 0 1	x x x x	SP1	+1
2	0 0 1 0	x x x x	CMA	$\bar{A}$
3	0 0 1 1	a a a a	LDA	(a) → A
4	0 1 0 0	x x x x	CLA	O
5	0 1 0 1	x x x x	INC	A + 1
6	0 1 1 0	x x x x	DEC	A – 1
7	0 1 1 1	a a a a	ADD	A + (a) → A
8	1 0 0 0	a a a a	SUB	A – (a) → A
9	1 0 0 1	a a a a	AND	A ∧ (a) → A
A	1 0 1 0	a a a a	IOR	A v (a) → A
B	1 0 1 1	a a a a	XOR	A ⊻ (a) → A
C	1 1 0 0	x x x x	SM1	–1
D	1 1 0 1	x x x x	INP	B → A
E	1 1 1 0	a a a a	STA	A → (a)
F	1 1 1 1	–	–	–

xxxx = Adresse beliebig
aaaa = Adresse 0000 bis 1111

4

Vereinfachter Rechner

ITT

Nr.	Op-Code	Adresse	Befehl	Funktion
0	0 0 0 0	x x x x	NOP	$A \rightarrow A$
1	0 0 0 1	x x x x	SP1	$+1 \rightarrow A$
2	0 0 1 0	x x x x	CMA	$\bar{A} \rightarrow A$
3	0 0 1 1	a a a a	LDA	$(a) \rightarrow A$
4	0 1 0 0	x x x x	CLA	$0 \rightarrow A$
5	0 1 0 1	x x x x	INC	$A + 1 \rightarrow A$
6	0 1 1 0	x x x x	DEC	$A - 1 \rightarrow A$
7	0 1 1 1	a a a a	ADD	$A + (a) \rightarrow A$
8	1 0 0 0	a a a a	SUB	$A - (a) \rightarrow A$
9	1 0 0 1	a a a a	AND	$A \wedge (a) \rightarrow A$
A	1 0 1 0	a a a a	IOR	$A \vee (a) \rightarrow A$
B	1 0 1 1	a a a a	XOR	$A \veebar (a) \rightarrow A$
C	1 1 0 0	x x x x	SM1	$-1 \rightarrow A$
D	1 1 0 1	x x x x	INP	$B \rightarrow A$
E	1 1 1 0	a a a a	STA	$A \rightarrow (a)$
F	1 1 1 1	x x x x	HLT	$A = A$

xxxx = Adresse beliebig

aaaa = Adresse 0000 bis 1111

5

Hypothetischer Rechner

ITT

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

Display-Codes

Hex Nr.	B_5 B_4 B_3 oder B_2 B_1 B_0	Anzeige des Inhaltes von
0	0 0 0	R0
1	0 0 1	R1
2	0 1 0	R2
3	0 1 1	R3
4	1 0 0	PC
5	1 0 1	Flags
6	1 1 0	<PC>
7	1 1 1	<A_7 ... A_0>

<...> =

Inhalt der Adresse die
– PC angibt
– A_7 ... A_0 angibt

Flags:
N = Negativ
Z = Zero
R = Run
C = Carry

5 HYPOTHETISCHER RECHNER

ITT

BEFEHLSLISTE

Befehl	OP-CODE	FUNTION	FLAGS N Z C
HALT	0000 0000 00	hält Rechner an	– – –
NOP	0000 1111 0F	keine Operation	– – –
MOVE	0000 ssdd 0.	(ss) → dd	– – –
ADDR	0001 ssdd 1.	(ss) + (dd) → dd	x x x
SUBR	0010 ssdd 2.	(ss) – (dd) → dd	x x x
IORR	0011 ssdd 3.	(ss) v (dd) → dd	x x 0
XORR	0100 ssdd 4.	(ss) ⊻ (dd) → dd	x x 0
ANDR	0101 ssdd 5.	(ss) ∧ (dd) → dd	x x 0
INCR	0110 00ss 6.	(ss) + 1 → ss	x x –
DECR	0110 01ss 6.	(ss) – 1 → ss	x x –
RACL	0110 10ss 6.	ROT ss u. C links	– – x
RACR	0110 11ss 6.	ROT ss u. C rechts	– – x
STAC	0111 mmss 7.	(ss) → (mm)	– – –
LOAD	1000 mmdd 8.	(mm) → dd	– – –
ADDM	1001 mmdd 9.	(mm) + (dd) → dd	x x x
SUBM	1010 mmdd A.	(mm) – (dd) → dd	x x x
IORM	1011 mmdd B.	(mm) v (dd) → dd	x x 0
XORM	1100 mmdd C.	(mm) ⊻ (dd) → dd	x x 0
ANDM	1101 mmdd D.	(mm) ∧ (dd) → dd	x x 0
JUMP	1110 mm00 E.	SPRUNG (mm) → PC	– – –
JMPZ	1110 mm01 E.	SPRUNG (mm) → PC falls Z = 1	– – –
JMPN	1110 mm10 E.	SPRUNG (mm) → PC falls N = 1	– – –
JMPC	1110 mm11 E.	SPRUNG (mm) → PC falls C = 1	– – –
CALL	1111 mm00 F.	UNTERPROGR.-ANRUF	– – –
CALZ	1111 mm01 F.	UNTERPROGR.-ANRUF falls Z = 1	– – –
CALN	1111 mm10 F.	UNTERPROGR.-ANRUF falls N = 1	– – –
CALC	1111 mm11 F.	UNTERPROGR.-ANRUF falls C = 1	– – –

FLAGS: – = unbeeinflußt, x = 0 oder 1

ADRESS-MODES mm

BEFEHLS-GRUPPE	MODE 00	MODE 01	MODE 10	MODE 11
LOAD ADDM SUBM IORM XORM ANDM	# IMME-DIATE	ABSOLUT oder DIREKT	@ R2 INDEXED über R2	@ R3 ↑ AUTOIN-CREMENT INDEXED über R3
STAC	@ R3 ↓ INDEXED AUTODE-CREMENT über R3	ABSOLUT oder DIREKT	@ R2 INDEXED über R2	@ R3 ↑ AUTOIN-CREMENT INDEXED über R3
JUMP JMPZ JMPN JMPC	ABSOLUT oder DIREKT	@ INDIREKT	@ R2 INDEXED über R2	@ R3 ↑ AUTOIN-CREMENT INDEXED über R3
CALL CALZ CALN CALC	ABSOLUT oder DIREKT	@ INDIREKT	@ R2 INDIREKT über R2	nicht ver-wendet

REGISTER-ADRESSE

REGISTER	ss oder dd
R0	0 0
R1	0 1
R2	1 0
R3	1 1

ss = SRC = Source
dd = DST = Destination
mm = MODE
N = Negativ Flag
Z = Zero Flag
C = Carry Flag

Mikroprozessor 8080

ITT

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

RAM-relat. Adr.	Stack Offset	Register
F4	0	L
F5	1	H
F6	2	E
F7	3	D
F8	4	C
F9	5	B
FA	6	Flags
FB	7	Akku
FC	8	PC low
FD	9	PC high

..00
00..

bei Er-	A	Return	Anwender
weiterung	B	Adr. 1	
	C	Return	
System 7	D	Adr. 2	
	E	Return	
	F	Adr. 3	

BEFEHLSLISTE

BEFEHL	OP-CODE	FUNTION	FLAGS NZHPC
MOV	0 1 d d d s s s	Move	– – – – –
MVI	0 0 d d d 1 1 0	Move #	– – – – –
STA	0 0 1 1 0 0 1 0	Store accu	– – – – –
LDA	0 0 1 1 1 0 1 0	Load accu	– – – – –
STAX	0 0 r r 0 0 1 0	Store accu @ rp	– – – – –
LDAX	0 0 r r 1 0 1 0	Load accu @ rp	– – – – –
LXI	0 0 r r 0 0 0 1	Load rp #	– – – – –
SHLD	0 0 1 0 0 0 1 0	Store HL dir.	– – – – –
LHLD	0 0 1 0 1 0 1 0	Load HL dir.	– – – – –
XCHG	1 1 1 0 1 0 1 1	Exchange HL, DE	– – – – –
IN	1 1 0 1 1 0 1 1	Input	– – – – –
OUT	1 1 0 1 0 0 1 1	Output	– – – – –
ADD	1 0 0 0 0 s s s	Add	x x x x x
ADI	1 1 0 0 0 1 1 0	Add #	x x x x x
ADC	1 0 0 0 1 s s s	Add with Carry	x x x x x
ACI	1 1 0 0 1 1 1 0	Add # with Carry	x x x x x
SUB	1 0 0 1 0 s s s	Subtract	x x x x x
SUI	1 1 0 1 0 1 1 0	Sub. #	x x x x x
SBB	1 1 0 1 1 s s s	Sub. with borrow	x x x x x
SBI	1 1 0 1 1 1 1 0	Sub. # with borrow	x x x x x
CMP	1 0 1 1 1 s s s	Compare	x x x x x
CPI	1 1 1 1 1 1 1 0	Compare #	x x x x x
DAD	0 0 r r 1 0 0 1	Double prec. add.	– – – – x
DAA	0 0 1 0 0 1 1 1	Decimal adj. accu	x x x x x
ANA	1 0 1 0 0 s s s	AND accu	x x – x x
ANI	1 1 1 0 0 1 1 0	AND #	x x – x x
ORA	1 0 1 1 0 s s s	OR accu	x x – x x

BEFEHL	OP-CODE	FUNTION	FLAGS NZHPC
ORI	1 1 1 1 0 1 1 0	OR #	x x – x x
XRA	1 0 1 0 1 s s s	EXCL. OR accu	x x – x x
XRI	1 1 1 0 1 1 1 0	EXCL. OR #	x x – x x
RLC	0 0 0 0 0 1 1 1	Rot. left into Carry	– – – – x
RAL	0 0 0 1 0 1 1 1	Rot. accu left	– – – – x
RRC	0 0 0 0 1 1 1 1	Rot. right into Carry	– – – – x
RAR	0 0 0 1 1 1 1 1	Rot. accu right	– – – – x
INR	0 0 d d d 1 0 0	Increment register	x x x x –
INX	0 0 r r 0 0 1 1	Increment rp	– – – – –
DCR	0 0 d d d 1 0 1	Decrement register	x x x x –
DCX	0 0 r r 1 0 1 1	Decrement rp	– – – – –
JMP	1 1 0 0 0 0 1 1	Jump unconditionally	– – – – –
J...	1 1 b b b 0 1 0	Jump if ...	– – – – –
PCHL	1 1 1 0 1 0 0 1	HL to PC	– – – – –
CALL	1 1 0 0 1 1 0 1	Call subroutine	– – – – –
C...	1 1 b b b 1 0 0	Call if ...	– – – – –
RET	1 1 0 0 1 0 0 1	Return from subrout.	– – – – –
R...	1 1 b b b 0 0 0	Return if ...	– – – – –
EI	1 1 1 1 1 0 1 1	Interupt enable	– – – – –
DI	1 1 1 1 0 0 1 1	Interupt disable	– – – – –
RST	1 1 a a a 1 1 1	Restart to aaax8	– – – – –
PUSH	1 1 r r 0 1 0 1	Push rp outo stack	– – – – –
POP	1 1 r r 0 0 0 1	Pop rp into stack	x x x x x
XTHL	1 1 1 0 0 0 1 1	Exchange HL, st.-top	– – – – –
SPHL	1 1 1 1 1 0 0 1	HL to stack pointer	– – – – –
CMA	0 0 1 0 1 1 1 1	Complement accu	– – – – –
CMC	0 0 1 1 1 1 1 1	Complem. Carry flag	– – – – x

BEFEHL	OP-CODE	FUNTION	FLAGS NZHPC
STC	0 0 1 1 0 1 1 1	Set Carry flag	– – – – 1
NOP	0 0 0 0 0 0 0 0	No-operation	– – – – –
HLT	0 1 1 1 0 1 1 0	Halt	– – – – –

REGISTER-CODE

sss, ddd	REGISTER
0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L
1 1 0	MEMORY
1 1 1	A (accu)

REGISTER-PAIR-CODE

r r	REG.-PAIR (rp)
0 0	B – C = B
0 1	D – E = D
1 0	H – L = H
1 1	stack pointer SP

BRANCH-CONDITION-CODE

b b b	BEFEHL	CONDITION
0 0 0	... NZ	... not zero
0 0 1	... Z	... zero
0 1 0	... NC	... carry 0
0 1 1	... C	... carry 1
1 0 0	... PO	... par. odd
1 0 1	... PE	... par. even
1 1 0	... P	... plus
1 1 1	... M	... minus

Abkürzungen:

sss = source register
ddd = destination register
rr = register-pair (rp)
bbb = branch condition
N = negative flag
Z = zero flag
H = half-carry flag
P = parity flag
C = carry flag
# = immediate
@ = indexed
x = flag to 0 or 1
– = operation without flag
aaa = (000 ... 111) x 8 for adress