	Maschinencode	Hex.-Code	Mnemonische Abkürzung			Flags N	Z	H	P	C	Bytes	Zyklen
Stack-Befehle	1 1 0 0 0 1 0 1	C 5	PUSH B	Push reg. pair BC	(BC) →Stack	–	–	–	–	–	1	11
	1 1 0 1 0 1 0 1	D 5	PUSH D	Push reg. pair DE	(DE) →Stack	–	–	–	–	–	1	11
	1 1 1 0 0 1 0 1	E 5	PUSH H	Push reg. pair HL	(HL) →Stack	–	–	–	–	–	1	11
	1 1 1 1 0 1 0 1	F 5	PUSH PSW	Push progr. status word	(Accu, Flags)→Stack	–	–	–	–	–	1	11
	1 1 0 0 0 0 0 1	C 1	POP B	Pop reg. pair BC	(Stack) →BC	–	–	–	–	–	1	10
	1 1 0 1 0 0 0 1	D 1	POP D	Pop. reg. pair DE	(Stack) →DE	–	–	–	–	–	1	10
	1 1 1 0 0 0 0 1	E 1	POP H	Pop reg. pair HL	(Stack →HL	–	–	–	–	–	1	10
	1 1 1 1 0 0 0 1	F 1	POP PSW	Pop progr. status word	(Stack) →Accu, Flags	x	x	x	x	x	1	10
	0 0 1 1 0 0 0 1	3 1	LXI SP	Load immediate Stack-Pointer	(3. u. 2. Byte)→SP	–	–	–	–	–	3	10
	1 1 1 1 1 0 0 1	F 9	SPHL	Stack-Pointer from HL	(HL) →SP	–	–	–	–	–	1	5
	0 0 1 1 0 0 1 1	3 3	INX SP	Incr. Stack-Pointer	(SP) + 1 →SP	–	–	–	–	–	1	5
	0 0 1 1 1 0 1 1	3 B	DCX SP	Decr. Stack-Pointer	(SP) – 1 →SP	–	–	–	–	–	1	5
	0 0 1 1 1 0 0 1	3 9	DAD SP	Double prec. add SP	(HL) + (SP) →HL	–	–	–	–	x	1	10
	1 1 1 0 0 0 1 1	E 3	XTHL	Exchange HL with top of stack	(HL) mit ((SP)) vertauschen	–	–	–	–	–	1	18
Sonstige Befehle	0 1 1 1 0 1 1 0	7 6	HLT	Halt							1	7
	0 0 0 0 0 0 0 0	0 0	NOP	No operation	**keine** Operation						1	4
	0 0 1 0 1 1 1 1	2 F	CMA	Complement accu	(A)→A						1	4
	0 0 1 1 0 1 1 1	3 7	STC	Set carry	1→C-Flag					1	1	4
	0 0 1 1 1 1 1 1	3 F	CMC	Complement carry	(C-Flag)→C-Flag					x	1	4

Registercode für s s s bzw. d d d:

0 0 0 ≙ Register B
0 0 1 ≙ Register C
0 1 0 ≙ Register D
0 1 1 ≙ Register E
1 0 0 ≙ Register H
1 0 1 ≙ Register L
1 1 0 ≙ Memory (nicht als s s s **und** d d d verwenden!)Set carry
1 1 1 ≙ Accumulator

Abkürzungen:

x = Flag wird beeinflußt
– = Flag wird nicht beeinflußt
(. . .) = Inhalt von . . .
((. . .)) = Inhalt von . . . ist die Adresse, deren Inhalt verarbeitet wird

	Maschinencode	Hex.-Code	Mnemonische Abkürzung			Flags N Z H P C	Bytes	Zyklen
Sprungbefehle	1 1 0 0 0 0 1 1	C 3	JMP	Jump unconditionally	(3. und 2. Byte)→PC	keine Beeinflussung der Flags	3	10
	1 1 0 0 0 0 1 0	C 2	JNZ	Jump if not zero	Adresse im 3. u. 2. Byte wird in den Programm-zähler gebracht, wenn: Z = 0		3	10
	1 1 0 0 1 0 1 0	C A	JZ	Jump if zero	Z = 1		3	10
	1 1 0 1 0 0 1 0	D 2	JNC	Jump if carry not set	C = 0		3	10
	1 1 0 1 1 0 1 0	D A	JC	Jump if carry set	C = 1		3	10
	1 1 1 0 0 0 1 0	E 2	JPO	Jump if parity odd	P = 0		3	10
	1 1 1 0 1 0 1 0	E A	JPE	Jump if parity even	P = 1		3	10
	1 1 1 1 0 0 1 0	F 2	JP	Jump if plus	N = 0		3	10
	1 1 1 1 1 0 1 0	F A	JM	Jump if minus	N = 1		3	10
	1 1 1 0 1 0 0 1	E 9	PCHL	Programm-Counter from HL	(HL)→PC		1	5
Unterprogrammanrufe und Rücksprungbefehle	1 1 0 0 1 1 0 1	C D	CALL	Call unconditionally	(3. und 2. Byte)→PC		3	17
	1 1 0 0 0 1 0 0	C 4	CNZ	Call if not zero	(3. u. 2. Byte) in PC Rücksprung-adresse auf den Stack, wenn: Z = 0		3	11/17
	1 1 0 0 1 1 0 0	C C	CZ	Call if zero	Z = 1		3	11/17
	1 1 0 1 0 1 0 0	D 4	CNC	Call if carry not set	C = 0		3	11/17
	1 1 0 1 1 1 0 0	D C	CC	Call if carry set	C = 1		3	11/17
	1 1 1 0 0 1 0 0	E 4	CPO	Call if parity odd	P = 0		3	11/17
	1 1 1 0 1 1 0 0	E C	CPE	Call if parity even	P = 1		3	11/17
	1 1 1 1 0 1 0 0	F 4	CP	Call if plus	N = 0		3	11/17
	1 1 1 1 1 1 0 0	F C	CM	Call if minus	N = 1		3	11/17
	1 1 0 0 1 0 0 1	C 9	RET	Return unconditionally	Rückspr.adr. vom Stack→PC		1	10
	1 1 0 0 0 0 0 0	C 0	RNZ	Return if not zero	Rücksprung-adresse vom Stack in den Programm-zähler, wenn: Z = 0		1	5/11
	1 1 0 0 1 0 0 0	C 8	RZ	Return if zero	Z = 1		1	5/11
	1 1 0 1 0 0 0 0	D 0	RNC	Return if carry not set	C = 0		1	5/11
	1 1 0 1 1 0 0 0	D 8	RC	Return if carry set	C = 1		1	5/11
	1 1 1 0 0 0 0 0	E 0	RPO	Return if parity odd	P = 0		1	5/11
	1 1 1 0 1 0 0 0	E 8	RPE	Return if parity even	P = 1		1	5/11
	1 1 1 1 0 0 0 0	F 0	RP	Return if plus	N = 0		1	5/11
	1 1 1 1 1 0 0 0	F 8	RM	Return if minus	N = 1		1	5/11
Interrupt-befehle	1 1 1 1 1 0 1 1	F B	EI	Enable interrupt	nach EI Interrupt möglich		1	4
	1 1 1 1 0 0 1 1	F 3	DI	Disable interrupt	nach DI Interrupt nicht möglich		1	4
	1 1 a a a 1 1 1		RST	Restart	(PC)→stack; a a a x 8→PC		1	11

	Maschinencode	Hex.-Code	Mnemonische Abkürzung			Flags N Z H P C	Bytes	Zyklen
Einzelgenauigkeits-Datentransferbefehle	0 1 d d d s s s		MOV r1, r2	Move register to register	(s s s) →d d d	keine Beeinflussung der Flags	1	5
	0 1 d d d 1 1 0		MOV r, M	Move from memory	(@HL) →d d d		1	7
	0 1 1 1 0 s s s		MOV M, r	Move to memory	(s s s) →@HL		1	7
	0 0 d d d 1 1 0		MVI r	Move immediate to register	(2.Byte)→d d d		2	7
	0 0 1 1 0 1 1 0	3 6	MVI M	Move immediate to memory	(2.Byte)→@HL		2	10
	0 0 1 1 0 0 1 0	3 2	STA adr	Store accumulator	(A) →(3. u. 2. Byte)		3	13
	0 0 1 1 1 0 1 0	3 A	LDA adr	Load accumulator	((3. u. 2. Byte))→A		3	13
	0 0 0 0 0 0 1 0	0 2	STAX B	Store accumulator indexed	(A) →@BC		1	7
	0 0 0 1 0 0 1 0	1 2	STAX D		(A) →@DE		1	7
	0 0 0 0 1 0 1 0	0 A	LDAX B	Load accumulator indexed	(@BC) →A		1	7
	0 0 0 1 1 0 1 0	1 A	LDAX D		(@DE) →A		1	7
Doppelgenauigkeits-Datentransferbefehle	0 0 0 0 0 0 0 1	0 1	LXI B	Load register pair immediate	(2.Byte)→C; (3. Byte)→B		3	10
	0 0 0 1 0 0 0 1	1 1	LXI D		(2.Byte)→E; (3. Byte)→D		3	10
	0 0 1 0 0 0 0 1	2 1	LXI H		(2.Byte)→L; (3. Byte)→H		3	10
	0 0 1 0 0 0 1 0	2 2	SHLD	Store H and L direct	(L) →(3. u. 2. Byte) (H) →(3. u. 2. Byte + 1)		3	16
	0 0 1 0 1 0 1 0	2 A	LHLD	Load H and L direct	((3. u. 2. Byte))→L ((3. u. 2. Byte + 1))→H		3	16
	1 1 1 0 1 0 1 1	E B	XCHG	Exchange HL with DE	(HL)→DE; (DE)→HL		1	4
Ein/Ausgabebefehle	1 1 0 1 1 0 1 1	D B	IN	Input	(Eingabekanal im 2. Byte)→A		2	10
	1 1 0 1 0 0 1 1	D 3	OUT	Output	A→Ausgabekanal im 2. Byte		2	10
Einzelgenauigkeits-arithmetikbefehle	1 0 0 0 0 s s s		ADD r	Add register	(A) + (s s s) →A	alle Flags werden entsprechend den Ergebnissen gesetzt	1	4
	1 0 0 0 0 1 1 0	1 6	ADD M	Add memory	(A) + (@HL) →A		1	7
	1 1 0 0 0 1 1 0	C 6	ADI	Add immediate	(A) + (2. Byte) →A		2	7
	1 0 0 0 1 s s s		ADC r	Add register with carry	(A) + (s s s) + (C) →A		1	4
	1 0 0 0 1 1 1 0	8 E	ADC M	Add memory with carry	(A) + (@HL) + (C) →A		1	7
	1 1 0 0 1 1 1 0	C E	ACI	Add immediate with carry	(A) + (2. Byte) + (C)→A		2	7
	1 0 0 1 0 s s s		SUB r	Subtract register	(A) − (s s s) →A		1	4
	1 0 0 1 0 1 1 0	9 6	SUB M	Subtract memory	(A) − (@HL) →A		1	7
	1 1 0 1 0 1 1 0	D 6	SUI	Subtract immediate	(A) − (2. Byte) →A		2	7
	1 0 0 1 1 s s s		SBB r	Sub. reg. with borrow	(A) − (s s s) − (C) →A		1	4
	1 0 0 1 1 1 1 0	9 E	SBB M	Sub. mem. with borrow	(A) − (@HL) − (C)→A		1	7
	1 1 0 1 1 1 1 0	D E	SBI	Sub. imm. with borrow	(A) − (2.Byte) − (C)→A		2	7
	1 0 1 1 1 s s s		CMP r	Compare register	(A) − (s s s) setzt Flags		1	4
	1 0 1 1 1 1 1 0	B E	CMP M	Compare memory	(A) − (@HL) setzt Flags		1	7
	1 1 1 1 1 1 1 0	F E	CPI	Compare immediate	(A) − (2. Byte) setzt Flags		2	7

	Maschinencode	Hex.-Code	Mnemonische Abkürzung			Flags N Z H P C	Bytes	Zyklen
Doppelgenauigkeits- und Dezimalarithmetikbefehle	0 0 0 0 1 0 0 1	0 9	DAD B	Double precision add	(HL) + (BC) →HL	– – – – x	1	10
	0 0 0 1 1 0 0 1	1 9	DAD D		(HL) + (DE) →HL	– – – – x	1	10
	0 0 1 0 1 0 0 1	2 9	DAD H		(HL) + (HL) →HL	– – – – x	1	10
	0 0 1 1 1 0 0 1	3 9	DAD SP		(HL) + (SP) →HL	– – – – x	1	10
	0 0 1 0 0 1 1 1	2 7	DAA	Decimal adjust Accu		x x x x x	1	4
Logische Befehle	1 0 1 0 0 s s s		ANA r	AND accumulator	(A) ∧ (s s s) →A	x x x x 0	1	4
	1 0 1 1 0 s s s		ORA r	OR accumulator	(A) ∨ (s s s) →A	x x 0 x 0	1	4
	1 0 1 0 1 s s s		XRA r	EXCLUSIVE-OR accu	(A) ∀ (s s s) →A	x x 0 x 0	1	4
	1 0 1 0 0 1 1 0	A 6	ANA M	AND memory to accu	(A) ∧ (@HL) →A	x x x x 0	1	7
	1 0 1 1 0 1 1 0	B C	ORA M	OR memory to accu	(A) ∨ (@HL) →A	x x 0 x 0	1	7
	1 0 1 0 1 1 1 0	A E	XRA M	EXCL.-OR memory to accu	(A) ∀ (@HL) →A	x x 0 x 0	1	7
	1 1 1 0 0 1 1 0	E 6	ANI	AND immediate	(A) ∧ (2. Byte) →A	x x 0 x 0	2	7
	1 1 1 1 0 1 1 0	F 6	ORI	OR immediate	(A) ∨ (2. Byte) →A	x x 0 x 0	2	7
	1 1 1 0 1 1 1 0	E E	XRI	EXCL.-OR immediate	(A) ∀ (2. Byte) →A	x x 0 x 0	2	7
Rotier- und Verschiebebefehle	0 0 0 0 0 1 1 1	0 7	RLC	Rotate left	(bit 7)→bit 0 und C	– – – – x	1	4
	0 0 0 0 1 1 1 1	0 F	RRC	Rotate right	(bit 0)→bit 7 und C	– – – – x	1	4
	0 0 0 1 0 1 1 1	1 7	RAL	Rotate left through C	(bit 7)→C; (C)→bit 0	– – – – x	1	4
	0 0 0 1 1 1 1 1	1 F	RAR	Rotate right through C	(bit 0)→C; (C)→bit 7	– – – – x	1	4
	0 0 1 0 1 0 0 1	2 9	DAD H	Double precision add	≙ linksschieben des Registerpaares H, L	– – – – x	1	10
Increment- und Decrement-Befehle	0 0 d d d 1 0 0		INR r	Increment register	(d d d) + 1→d d d	x x x x –	1	5
	0 0 d d d 1 0 1		DCR r	Decrement register	(d d d) – 1→d d d	x x x x –	1	5
	0 0 1 1 0 1 0 0	3 4	INR M	Increment memory	(@HL) + 1→@HL	x x x x –	1	10
	0 0 1 1 0 1 0 1	3 5	DCR M	Decrement memory	(@HL) – 1→@HL	x x x x –	1	10
	0 0 0 0 0 0 1 1	0 3	INX B	Increment Register pair	(BC) + 1→BC	– – – – –	1	5
	0 0 0 1 0 0 1 1	1 3	INX D		(DE) + 1→DE	– – – – –	1	5
	0 0 1 0 0 0 1 1	2 3	INX H		(HL) + 1→HL	– – – – –	1	5
	0 0 0 0 1 0 1 1	0 B	DCX B	Decrement Register pair	(BC) – 1→BC	– – – – –	1	5
	0 0 0 1 1 0 1 1	1 B	DCX D		(DE) – 1→DE	– – – – –	1	5
	0 0 1 0 1 0 1 1	2 B	DCX H		(HL) – 1 HL	– – – – –	1	5
	0 0 1 1 0 0 1 1	3 3	INX SP	Incr. Stack-Pointer	(SP) + 1→SP	– – – – –	1	5
	0 0 1 1 1 0 1 1	3 B	DCX SP	Decr. Stack-Pointer	(SP) – 1→SP	– – – – –	1	5