

68000er-Befehle

Mnemonic	Funktion	Flags	effektive Adresse/Zahl der Taktzyklen													
			XNZVC	Dn	An	(An)	(An)+	-(An)	d(An)	d(An,Rx)	xx.W	xxxx.L	d(PC)	d(PC,Rx)	#xxxx	CCR/SR
A																
ABCD Dx,Dy	(Quelle)10 + (Ziel)10 + X -> Ziel Addition von bcd-Zahlen Ergebnis <= 0: Z=0; else Z=Z	*U*U*	B W L	06 - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
ABCD -(Ax), -(Ay)		*U*U*	B W L	- - -	- - -	- - -	18 - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
ADD ea,Dy	(Quelle) + (Ziel) -> Ziel Addition von Binärzahlen ohne Übertrag	****	B W L	04 04 06	- 08 08	08 08 14	08 10 14	10 12 16	12 14 18	12 14 20	12 14 18	16 12 22	16 14 22	18 14 22	20 18 22	08 08 14
ADD Dx,ea		****	B W L	04 04 06	- 12 12	12 12 14	14 16 18	16 18 20	18 16 20	16 20 20	20 20 28	- - -	- - -	- - -	- - -	- - -
ADDA ea,Ay		-----	B W L	- 08 08	- 08 08	- 12 12	- 14 14	- 16 16	- 18 18	- 20 20	- 22 24	- 24 26	- 26 24	- 28 28	- 20 16	- 18 12
ADDI #xxxx,ea		****	B W L	08 08 16	- 16 16	16 16 18	18 20 20	20 22 20	24 20 24	24 20 32	24 20 32	24 20 36	- - -	- - -	- - -	- - -
ADDQ #x,ea	x: 1..8	****	B W L	04 04 08	- 12 12	12 12 14	14 16 18	16 18 20	18 16 20	20 20 28	- - -	- - -	- - -	- - -	- - -	- - -
ADDX Dx,Dy	(Quelle) + (Ziel) + X -> Ziel mit Übertrag in X Z wie bei ABCD	****	B W L	04 04 08	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
ADDX -(Ax), -(Ay)		****	B W L	- - -	- - -	- - -	18 18 30	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
AND ea,Dy	(Quelle) & (Ziel) -> Ziel	-**00	B W L	04 04 06	- 08 08	08 08 14	10 12 16	12 14 18	12 16 20	14 18 22	14 18 22	16 20 28	18 22 28	20 24 28	08 08 14	- - -
AND Dx,ea		-**00	B W L	04 04 06	- 12 12	12 12 14	14 16 18	16 18 20	18 16 20	20 20 28	- - -	- - -	- - -	- - -	- - -	- - -
ANDI #xxxx,ea		-**00	B W L	08 08 16	- 16 16	16 18 20	18 20 22	20 22 20	24 20 24	24 20 32	24 20 36	- - -	- - -	- - -	- - -	20* 20 -
ASL ea,Dy	(Ziel) leftshift um (Quelle) Bits C/X=MSB; LSB=0 bei #: 1..8	****	B W L	6+2x 6+2x 8+2x	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	6+2x 6+2x 8+2x
ASL ea	Verschiebung um ein Bit	****	B W L	- - -	- 12 12	- 12 14	- 16 18	- 16 20	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
ASR ea,Dy	(Ziel) rightshift um (Quelle) Bits C/X=LSB; MSB=MSB bei #: 1..8	****	B W L	6+2x 6+2x 8+2x	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	6+2x 6+2x 8+2x
ASR ea	Verschiebung um ein Bit	****	B W L	- - -	- 12 12	- 12 14	- 16 18	- 16 20	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
B																
Bcc Label	cc erfüllt: PC + D -> PC Dn: erfüllt; An: nicht erfüllt	-----	B W L	10 10 -	08 12 -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -

Mnemonic	Funktion	Flags	effektive Adresse/Zahl der Taktzyklen													
			XNZVC	Dn	An	(An)	(An)+	-(An)	d(An)	d(An,Rx)	xx.W	xxxx.L	d(PC)	d(PC,Rx)	#xxxx	CCR/SR
BCHG Dx,ea	invert bit(Quelle)(Ziel) Z=bit(Quelle)(Ziel)	---*	B W L	- - 0	- - 8	12 - -	12 - -	14 - -	16 - -	18 - -	16 - -	20 - -	- - -	- - -	- - -	- - -
BCHG #x,ea	Dy: x:0..31; else 0..7	---*	B W L	- - 12	- - -	16 - -	16 - -	18 - -	20 - -	22 - -	20 - -	24 - -	- - -	- - -	- - -	- - -
BCLR Dx,ea	0 -> bit(Quelle)(Ziel) Z=bit(Quelle)(Ziel)	---*	B W L	- - 12	- - -	12 - -	12 - -	14 - -	16 - -	18 - -	16 - -	20 - -	- - -	- - -	- - -	- - -
BCLR #x,ea	Dy: x:0..31; else 0..7	---*	B W L	- - 14	- - -	16 - -	16 - -	18 - -	20 - -	22 - -	20 - -	24 - -	- - -	- - -	- - -	- - -
BSET Dx,ea	1 -> bit(Quelle)(Ziel) Z=bit(Quelle)(Ziel)	---*	B W L	- - 0	- - 8	12 - -	12 - -	14 - -	16 - -	18 - -	16 - -	20 - -	- - -	- - -	- - -	- - -
BSET #x,ea	Dy: x:0..31; else 0..7	---*	B W L	- - 12	- - -	16 - -	16 - -	18 - -	20 - -	22 - -	20 - -	24 - -	- - -	- - -	- - -	- - -
BSR Label	PC -> -(SP); PC + D -> PC	-----														
BTST Dx,ea	Z=bit(Quelle)(Ziel)	---*	B W L	- - 0	- - 6	08 - -	08 - -	10 - -	12 - -	14 - -	12 - -	16 - -	12 - -	14 - -	- - -	- - -
BTST #x,ea	x: 0..31 (Dy); 0..7 (sonst)	---*	B W L	- - 10	- - -	12 - -	12 - -	14 - -	16 - -	18 - -	16 - -	20 - -	16 - -	18 - -	- - -	- - -
C																
CHK ea,Dy	(Quelle)<0 (Quelle)>Dy: Trap #6	-*UUU	T N	44 10	- -	48 14	48 14	50 16	52 18	54 20	52 18	56 22	52 18	54 20	48 14	- -
CLR ea	0 -> Ziel	-0100	B W L	04 04 06	- 12 12	12 14 16	14 16 18	16 18 20	18 20 22	16 18 24	16 20 24	20 24 28	- - -	- - -	- - -	- - -
CMP ea,Dy	Wie SUB aber ohne Ergebnis	****	B W L	04 04 06	- 04 06	08 08 14	08 10 14	10 12 16	12 14 18	12 16 20	12 16 22	14 18 24	14 18 22	16 20 24	08 08 14	- - -
CMPA ea,Ay		****	B W L	- 06 06	- 10 10	- 10 14	- 12 16	- 14 18	- 16 20	- 18 22	- 14 18	- 16 20	- 18 22	- 14 18	- 16 20	08 18 28
CMPI #xxxx,ea		****	B W L	08 08 14	- 12 12	12 14 16	14 16 18	16 18 20	16 20 22	20 24 26	20 24 28	- - -	- - -	- - -	- - -	- - -
CMPM (Ax)+, (Ay)+		****	B W L	- - -	- 12 12	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
D																
DBcc Dx,Label	cc erfüllt: PC+2->PC; else Dx - 1->Dx Dx=1: PC+2->PC; else PC+D->PC	-----														
DIVS ea,Dy	(Ziel) / (Quelle) -> Ziel 32-bit-Ziel durch 16-bit-Quelle Quot -> LSW; Rest -> MSW	****0	B W L	- 158 -	- 162 -	- 162 164	- 164 166	- 166 168	- 166 170	- 166 168	- 170 166	- 166 168	- 168 162	- 162 162	- - -	- - -
DIVU ea,Dy	wie DIVS aber unsigned	****0	B W L	- 140 -	- 144 -	- 144 146	- 146 148	- 148 150	- 148 148	- 152 148	- 148 150	- 148 144	- 144 144	- 144 144	- - -	- - -

Mnemonic	Funktion	Flags	effektive Adresse/Zahl der Taktzyklen															
			XNZVC	Dn	An	(An)	(An)+	-(An)	d(An)	d(An,Rx)	xx.W	xxxx.L	d(PC)	d(PC,Rx)	#xxxx	CCR/SR		
R																		
ROR																		
	Quelle I (Ziel) -> Ziel	-**00	B	04	-	12	12	14	16	18	16	20	-	-	-	-	-	-
			W	04	-	12	12	14	16	18	16	20	-	-	-	-	-	-
			L	08	-	20	20	22	24	26	24	28	-	-	-	-	-	-
RORI #xxxx,ea		-**00	B	08	-	16	16	18	20	22	20	24	-	-	-	-	-	20*
			W	08	-	16	16	18	20	22	20	24	-	-	-	-	-	20
			L	16	-	28	28	30	32	34	32	36	-	-	-	-	-	-
RXG ea,Dy	Quelle <-> Ziel	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	06	06	-	-	-	-	-	-	-	-	-	-	-	-	-
RXG Ax,Ay		-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	-	06	-	-	-	-	-	-	-	-	-	-	-	-	-
RXT Dx		-**00	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	04	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	04	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I																		
ILLEGAL	PC -> -(SSP);SR -> -(SSP);Trap #4	-----		34														
J																		
JMP ea	Ziel->PC	-----	L	-	-	08	-	-	10	14	10	12	10	14	-	-	-	-
JSR ea	PC -> -(SP);Ziel->PC	-----	L	-	-	16	-	-	18	22	18	20	18	22	-	-	-	-
L																		
LEA ea,Ay	Ziel->Ay	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	-	-	04	-	-	08	12	08	12	08	12	-	-	-	-
LINK Ax,#xx	Ax -> -(SP);SP -> Ax;SP + D -> SP	-----	L															
LSL ea,Dy	(Ziel) leftshift um (Quelle) Bits C/X=MSB; LSB=0 bei #: 1..8	***0*	B	6:2x	-	-	-	-	-	-	-	-	-	-	-	-	-	6:2x
			W	6:2x	-	-	-	-	-	-	-	-	-	-	-	-	-	6:2x
			L	8:2x	-	-	-	-	-	-	-	-	-	-	-	-	-	8:2x
LSL ea	Verschiebung um ein Bit	***0*	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	12	12	14	16	18	16	20	-	-	-	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSR ea,Dy	(Ziel) rightshift um (Quelle) Bits C/X=LSB; MSB=0 bei #: 1..8	***0*	B	6:2x	-	-	-	-	-	-	-	-	-	-	-	-	-	6:2x
			W	6:2x	-	-	-	-	-	-	-	-	-	-	-	-	-	6:2x
			L	8:2x	-	-	-	-	-	-	-	-	-	-	-	-	-	8:2x
LSR ea	Verschiebung um ein Bit	***0*	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	12	12	14	16	18	16	20	-	-	-	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
M																		
MOVE ea,Dy	Quelle->Ziel	-**00	B	04	-	08	08	10	12	14	12	16	12	14	08	-	-	-
			W	04	04	08	08	10	12	14	12	16	12	14	08	-	-	-
			L	04	04	12	12	14	16	18	16	20	16	18	12	-	-	-
MOVE ea,(Ay)		-**00	B	08	-	12	12	14	16	18	16	20	16	18	12	-	-	-
			W	08	08	12	12	14	16	18	16	20	16	18	12	-	-	-
			L	12	12	20	20	22	24	26	24	28	24	26	20	-	-	-
MOVE ea,(Ay)+		-**00	B	08	-	12	12	14	16	18	16	20	16	18	12	-	-	-
			W	08	08	12	12	14	16	18	16	20	16	18	12	-	-	-
			L	12	12	20	20	22	24	26	24	28	24	26	20	-	-	-
MOVE ea,-(Ay)		-**00	B	08	-	12	12	14	16	18	16	20	16	18	12	-	-	-
			W	08	08	12	12	14	16	18	16	20	16	18	12	-	-	-
			L	12	12	20	20	22	24	26	24	28	24	26	20	-	-	-
MOVE ea,d(Ay)		-**00	B	12	-	16	16	18	20	22	20	24	20	22	16	-	-	-
			W	12	12	16	16	18	20	22	20	24	20	22	16	-	-	-
			L	16	16	24	24	26	28	30	28	32	28	30	24	-	-	-

Mnemonic	Funktion	Flags	effektive Adresse/Zahl der Taktzyklen															
			XNZVC	Dn	An	(An)	(An)+	-(An)	d(An)	d(An,Rx)	xx.W	xxxx.L	d(PC)	d(PC,Rx)	#xxxx	CCR/SR		
R																		
MOVE ea,d(Ay,Ri)		-**00	B	14	-	18	18	20	22	24	22	26	22	24	18	-	-	-
			W	14	14	18	18	20	22	24	22	26	22	24	18	-	-	-
			L	18	18	26	26	28	30	32	30	34	30	32	26	-	-	-
MOVE ea,xx.W		-**00	B	12	-	16	16	18	20	22	20	24	20	22	16	-	-	-
			W	12	12	16	16	18	20	22	20	24	20	22	16	-	-	-
			L	16	16	24	24	26	28	30	28	32	28	30	24	-	-	-
MOVE ea,xxxx.L		-**00	B	16	-	24	26	28	30	32	34	32	36	32	20	-	-	-
			W	16	16	20	20	22	24	26	24	28	24	26	20	-	-	-
			L	20	20	28	28	30	32	34	32	36	32	34	28	-	-	-
MOVE ea,CCR		*****	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	12	-	16	16	18	20	22	20	24	20	22	16	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MOVE ea,SR		*****	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	12*	-	16*	16*	18*	20*	22*	20*	24*	20*	22*	16*	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MOVE SR,ea		-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	06	-	12	12	14	16	18	16	20	-	-	-	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MOVE USP,Ay		-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	04*	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MOVE Ax,USP		-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	04*	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MOVEA ea,Ay	16 Bit: Vorzeichenweiterung	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	04	04	08	08	10	12	14	12	16	12	14	08	-	-	-
			L	04	04	12	12	14	16	18	16	20	16	18	12	-	-	-
MOVEM ea,RL	Quelle -> mehrere Register	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	12+4n	12+4n	-	-	-	-	-	-	-	-	-	-	-
			L	-	-	12+8n	12+8n	-	-	-	-	-	-	-	-	-	-	-
MOVEM RL,ea	mehrere Register -> Quelle	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	8+5n	8+5n	12+5n	14+5n	12+5n	16+5n	-	-	-	-	-	-	-
			L	-	-	8+10n	8+10n	12+10n	14+10n	12+10n	16+10n	-	-	-	-	-	-	-
MOVEP Dx,d(Ay)	Daten <-> 8bit-Peripherie Übertragung zw. folgenden gerade/ungerade Adr. u. Dtag.	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	-	-	-	16	-	-	-	-	-	-	-	-	-
			L	-	-	-	-	-	24	-	-	-	-	-	-	-	-	-
MOVEP d(Ax),Dy	Es werden erst höherwertige Bytes übertragen Ax/y = Ax/y + 2	-----	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	-	-	-	16	-	-	-	-	-	-	-	-	-
			L	-	-	-	-	-	24	-	-	-	-	-	-	-	-	-
MOVEQ #xx,Dy	Daten -> Ziel 32bit Vorzeichenweitert	-**00	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			L	04	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MULS ea,Dy	(Quelle) * (Ziel) -> Ziel 2 16bit -> 32bit	-**00	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	70	-	74	74	76	78	80	78	82	78	80	74	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MULU ea,Dy	wie MULS aber unsigned	-**00	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
			W	70	-	74	74	76	78	80	78	82	78	80	74	-	-	-
			L	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
N																		
NBCD ea	0 - (Ziel)10 - X -> Ziel	*U*U*	B	06	-	12												

CC	Bedeutung	Test
TR	true	1
F	false	0
HI	high	$C \& Z$
LS	low same	$C Z$
CC	carry clear	\overline{C}
CS	carry set	C
NE	not equal	Z
EQ	equal	\overline{Z}
VC	overflow clear	\overline{V}
VS	overflow set	V
PL	plus	\overline{N}
MI	minus	N
GE	greater equal	$(N \& V) (\overline{N} \& \overline{V})$
LT	less	$(N \& \overline{V}) (\overline{N} \& V)$
GT	greater	$(N \& V \& Z) (\overline{N} \& \overline{V} \& \overline{Z})$
LE	less equal	$Z (N \& \overline{V}) (\overline{N} \& V)$

Status-Register															
System-Byte						User-Byte									
T		S			12	11	10				X	N	Z	V	C
Trace		Supervisor			Interrupt- maske						Extended	Negativ	Zero	Overflow	Carry