MOVE

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
dX	4/4/4	--/4/4	8/8/12	8/8/12	8/8/12	12/12/16	14/14/18	12/12/16	16/16/20	--/12/--	--/12/--	--
aX	--/4/4	--/4/4	--/8/12	--/8/12	--/8/12	--/12/16	--/14/18	--/12/16	--/16/20	--	--	--/--/4
(aX)	8/8/12	--/8/12	12/12/20	12/12/20	12/12/20	16/16/24	18/18/26	16/16/24	20/20/28	--/16/--	--/16/--	--
(aX)+	8/8/12	--/8/12	12/12/20	12/12/20	12/12/20	16/16/24	18/18/26	16/16/24	20/20/28	--/16/--	--/16/--	--
-(aX)	10/10/14	--/10/14	14/14/22	14/14/22	14/14/22	18/18/26	20/20/28	18/18/26	22/22/30	--/18/--	--/18/--	--
$(aX)	12/12/16	--/12/16	16/16/24	16/16/24	16/16/24	20/20/28	22/22/30	20/20/28	24/24/32	--/20/--	--/20/--	--
$(aX,X)	14/14/18	--/14/18	18/18/26	18/18/26	18/18/26	22/22/30	24/24/32	22/22/30	26/26/34	--/22/--	--/22/--	--
(add).w	12/12/16	--/12/16	16/16/24	16/16/24	16/16/24	20/20/28	22/22/30	20/20/28	24/24/32	--/20/--	--/20/--	--
(add).l	16/16/20	--/16/20	20/20/28	20/20/28	20/20/28	24/24/32	26/26/34	24/24/32	28/28/36	--/24/--	--/24/--	--
$(pc)	12/12/16	--/12/16	16/16/24	16/16/24	16/16/24	20/20/28	22/2/30	20/20/28	24/24/32	--/20/--	--/20/--	--
$(pc,X)	14/14/18	--/14/18	18/18/26	18/18/26	18/18/26	22/22/30	24/24/32	22/22/30	26/26/28	--/22/--	--/22/--	--
#$	8/8/12	--/8/12	12/12/20	12/12/20	12/12/20	16/16/24	18/18/26	16/16/24	20/20/28	--/16/--	--/16/--	--
#$(-128..+127)	--/--/4	--	--	--	--	--	--	--	--	--	--	--
sr	--/6/--	--	--/12/--	--/12/--	--/14/--	--/16/--	--/18/--	--/16/--	--/20/--	--	--	--
usp	--	--/--/4	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
MOVE	–	*	*	0	0	[destination] ⟵ [source]	
MOVEA	–	–	–	–	–	[aX] ⟵ [source]	If .W specified, source sign-extended to destination
MOVEQ	–	*	*	0	0	[destination] ⟵ <literal>	8-bit literal sign-extended to 32 bits
MOVE to CCR	*	*	*	*	*	[CCR] ⟵ [source]	
MOVE from SR	–	–	–	–	–	[destination] ⟵ [SR]	
MOVE to SR	*	*	*	*	*	IF [S] == 1 THEN [SR] ⟵ [source] ELSE TRAP	
MOVE to/from USP	–	–	–	–	–	1IF [S] == 1, [USP] ⟵ [aX] / [aX] ⟵ [USP] ELSE TRAP	This is a privileged instruction

MOVEM

SOURCE \ DESTINATION	dX / aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
dX / aX	--	--/8/8	--	--/8/8	--/12/12	--/14/14	--/12/12	--/16/16	--	--	--
(aX)	--/12/12	--	--	--	--	--	--	--	--	--	--
(aX)+	--/12/12	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--/16/16	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--/18/18	--	--	--	--	--	--	--	--	--	--
(add).w	--/16/16	--	--	--	--	--	--	--	--	--	--
(add).l	--/20/20	--	--	--	--	--	--	--	--	--	--
$(pc)	--/16/16	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--/18/18	--	--	--	--	--	--	--	--	--	--
#$	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
MOVEM	–	–	–	–	–	1. [destination_registers] ⟵ [source] 2. [destination] ⟵ [source_registers]	1. Memory access occurs at one address higher than the last register accessed 2. If .W specified, source sign-extended when moved to dX/aX

cycles for / #/regs involved																
	1	--/4/8	2	--/8/16	3	--/12/24	4	--/16/32	5	--/20/40	6	--/24/48	7	--/28/56	8	--/32/64
	9	--/36/72	10	--/40/80	11	--/44/88	12	--/48/96	13	--/52/104	14	--/56/112	15	--/60/120	16	--/64/128

MOVEP

OURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	Sr	Usp
dX	--	--	--	--	--	--/16/24	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--/16/24	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--
#$	--	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C	READ	WRITE
MOVEP	–	–	–	–	–	-- / [dX(8:15)] / [dX(24:31) ⟵ [[aX] + $ + 0]	[[aX] + $ + 0] ⟵ -- / [dX(8:15)] / [dX(24:31)
						-- / [dX(0:7)] / [dX(16:23) ⟵ [[aX] + $ + 2]	[[aX] + $ + 2] ⟵ -- / [dX(0:7)] / [dX(16:23)]
						-- / -- / [dX(8: 15) ⟵ [[aX] + $ + 4]	[[aX] + $ + 4] ⟵ -- / -- / [dX(8:15)]
						-- / -- / [dX(0:7) ⟵ [[aX] + $ + 6]	[[aX] + $ + 6] ⟵ -- / -- / [dX(0:7)]

ADD / SUB

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	4/4/8	--/8/8	12/12/20	12/12/20	14/14/22	16/16/24	18/18/26	16/16/24	20/20/28	--	--	--
aX	--/4/8	--/8/8	--	--	--	--	--	--	--	--	--	--
(aX)	8/8/12	--/12/14	--	--	--	--	--	--	--	--	--	--
(aX)+	8/8/12	--/12/14	--	--	--	--	--	--	--	--	--	--
-(aX)	10/10/14	--/14/16	--	--	18/18/30	--	--	--	--	--	--	--
$(aX)	12/12/16	--/16/18	--	--	--	--	--	--	--	--	--	--
$(aX,X)	14/14/18	--/18/20	--	--	--	--	--	--	--	--	--	--
(add).w	12/12/16	--/16/18	--	--	--	--	--	--	--	--	--	--
(add).l	16/16/20	--/20/22	--	--	--	--	--	--	--	--	--	--
$(pc)	12/12/16	--/16/18	--	--	--	--	--	--	--	--	--	--
$(pc,X)	14/14/18	--/18/20	--	--	--	--	--	--	--	--	--	--
#$	8/8/16	--/12/16	16/16/28	16/16/28	18/18/30	20/20/32	22/22/34	20/20/32	24/24/36	--	--	--
#$(1..8)	4/4/8	--/8/8	12/12/20	12/12/20	14/14/22	16/16/24	18/18/26	16/16/24	20/20/28	--	--	--

	X	N	Z	V	C		
ADD / SUB	*	*	*	*	*	[destination] ⟵ [destination] +/- [source]	
ADDA / SUBA	–	–	–	–	–	[aX] ⟵ [aX] +/- [source]	If .W specified, source sign-extended before add to/sub from aX
ADDI / SUBI	*	*	*	*	*	[destination] ⟵ [destination] +/- <literal>	
ADDQ / SUBQ	*	*	*	*	*	[destination] ⟵ [destination] +/- <literal>	A word operation on aX affects all bits of the register. The CCR is
							not updated if destination is aX
ADDX / SUBX	*	*	*	*	*	[destination] ⟵ [destination] +/- [source] + [X]	Z-bit cleared if result is non-zero, left unchanged otherwise

CMP

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	4/4/6	--/6/6	--	--	--	--	--	--	--	--	--	--
aX	--/4/6	--/6/6	--	--	--	--	--	--	--	--	--	--
(aX)	8/8/14	--/10/14	--	--	--	--	--	--	--	--	--	--
(aX)+	8/8/14	--/10/14	--	12/12/20	--	--	--	--	--	--	--	--
-(aX)	10/10/16	--/12/16	--	--	--	--	--	--	--	--	--	--
$(aX)	12/12/18	--/14/18	--	--	--	--	--	--	--	--	--	--
$(aX,X)	14/14/20	--/16/20	--	--	--	--	--	--	--	--	--	--
(add).w	12/12/18	--/14/18	--	--	--	--	--	--	--	--	--	--
(add).l	16/16/22	--/18/22	--	--	--	--	--	--	--	--	--	--
$(pc)	12/12/18	--/14/18	--	--	--	--	--	--	--	--	--	--
$(pc,X)	14/14/20	--/16/20	--	--	--	--	--	--	--	--	--	--
#$	8/8/14	--/10/14	12/12/20	12/12/20	14/14/22	16/16/24	18/18/26	16/16/24	20/20/28	--	--	--

Click okay	X	N	Z	V	C		
CMP	–	*	*	*	*	[destination] - [source]	
CMPA	–	*	*	*	*	[destination] - [source]	If .W specified, source sign-extended before compare to aX
CMPI	–	*	*	*	*	[destination] - <literal>	
CMPM	–	*	*	*	*	[destination] - [source]	

ABCD / SBCD

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	6/--/--	--	--	--	--	--	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	18/--/--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
ABCD	*	U	*	U	*	$[destination]_{10}$ ⟵ $[destination]_{10}$ + $[source]_{10}$ + [X]	Z-bit cleared if non-zero, left unchanged otherwise
SBCD	*	U	*	U	*	$[destination]_{10}$ ⟵ $[destination]_{10}$ - $[source]_{10}$ - [X]	Z-bit cleared if non-zero, left unchanged otherwise

AND

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	4/4/8	--	12/12/20	12/12/20	14/14/22	16/16/24	18/18/26	16/16/24	20/20/28	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	8/8/14	--	--	--	--	--	--	--	--	--	--	--
(aX)+	8/8/14	--	--	--	--	--	--	--	--	--	--	--
-(aX)	10/10/16	--	--	--	--	--	--	--	--	--	--	--
$(aX)	12/12/18	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	14/14/20	--	--	--	--	--	--	--	--	--	--	--
(add).w	12/12/18	--	--	--	--	--	--	--	--	--	--	--
(add).l	16/16/22	--	--	--	--	--	--	--	--	--	--	--
$(pc)	12/12/18	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	14/14/20	--	--	--	--	--	--	--	--	--	--	--
#$	8/8/16	--	16/16/28	16/16/28	18/18/30	20/20/32	22/22/34	20/20/32	24/24/36	20/--/--	--/20/--	--

	X	N	Z	V	C		
AND	–	*	*	0	0	[destination] ⟵ [destination] & [source]	
ANDI	–	*	*	0	0	[destination] ⟵ [destination] & <literal>	
ANDI to CCR	*	*	*	*	*	[CCR] ⟵ [CCR] & <literal>	
ANDI to SR	*	*	*	*	*	IF [S] == 1 THEN [SR] ⟵ [SR] & <literal> ELSE TRAP	Used to clear the interrupt mask, S-bit, and T-bit of the SR

EOR

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	4/4/8	--	12/12/20	12/12/20	14/14/22	16/16/24	18/18/26	16/16/24	20/20/28	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--
#$	8/8/16	--	16/16/28	16/16/28	18/18/30	20/20/32	22/22/34	20/20/32	24/24/36	20/--/--	--/20/--	--

	X	N	Z	V	C		
EOR	–	*	*	0	0	[destination] ⟵ [destination] ^ [source]	
EORI	–	*	*	0	0	[destination] ⟵ [destination] ^ <literal>	
EORI to CCR	*	*	*	*	*	[CCR] ⟵ [CCR] ^ <literal>	
EORI to SR	*	*	*	*	*	IF [S] == 1 THEN [SR] ⟵ [SR] ^ <literal> ELSE TRAP	Used to toggle interrupt mask, S-bit, and T-bit of the SR

OR

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	4/4/8	--	--	--	--	--	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	8/8/14	--	--	--	--	--	--	--	--	--	--	--
(aX)+	8/8/14	--	--	--	--	--	--	--	--	--	--	--
-(aX)	10/10/16	--	--	--	--	--	--	--	--	--	--	--
$(aX)	12/12/18	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	14/14/20	--	--	--	--	--	--	--	--	--	--	--
(add).w	12/12/18	--	--	--	--	--	--	--	--	--	--	--
(add).l	16/16/22	--	--	--	--	--	--	--	--	--	--	--
$(pc)	12/12/18	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	14/14/20	--	--	--	--	--	--	--	--	--	--	--
#$	8/8/16	--	16/16/28	16/16/28	18/18/30	20/20/32	22/22/34	20/20/32	24/24/36	20/--/--	--/20/--	--

	X	N	Z	V	C		
OR	–	*	*	0	0	[destination] ⟵ [destination] | [source]	
ORI	–	*	*	0	0	[destination] ⟵ [destination] | <literal>	
OR I to CCR	*	*	*	*	*	[CCR] ⟵ [CCR] | <literal>	
OR I to SR	*	*	*	*	*	IF [S] == 1 THEN [SR] ⟵ [SR] | <literal> ELSE TRAP	Used to set the interrupt mask, S-bit, and T-bit of the SR

BCHG / BSET

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	--/--/6	--	12/--/--	12/--/--	14/--/--	16/--/--	18/--/--	16/--/--	20/--/--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--
#$	--/--/10	--	16/--/--	16/--/--	18/--/--	20/--/--	22/--/--	20/--/--	24/--/--	--	--	--

	X	N	Z	V	C	
BCHG	–	–	*	–	–	[Z] ⟵ <bit number> OF [destination] == 0
						<bit number> OF [destination] ⟵ <bit number> OF [destination] ^ 1
BSET	–	–	*	–	–	[Z] ⟵ <bit number> OF [destination] == 0
						<bit number> OF [destination] ⟵ 1

BCLR

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	--/--/8	--	12/--/--	12/--/--	14/--/--	16/--/--	18/--/--	16/--/--	20/--/--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--
#$	--/--/12	--	16/--/--	16/--/--	18/--/--	20/--/--	22/--/--	20/--/--	24/--/--	--	--	--

	X	N	Z	V	C	
BCLR	–	–	*	–	–	[Z] ⟵ <bit number> OF [destination] == 0
						<bit number> OF [destination] ⟵ 0

BTST

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	--/--/6	--	8/--/--	8/--/--	10/--/--	12/--/--	14/--/--	12/--/--	16/--/--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--
#$	--/--/10	--	12/--/--	12/--/--	14/--/--	16/--/--	18/--/--	16/--/--	20/--/--	--	--	--

	X	N	Z	V	C	
BTST	–	–	*	–	–	[Z] ⟵ <bit number> OF [destination] == 0

TAS

DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
	4/--/--	--	14/--/--	14/--/--	16/--/--	18/--/--	20/--/--	18/--/--	22/--/--	--	--	--

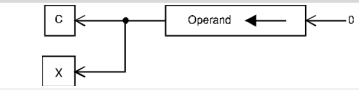
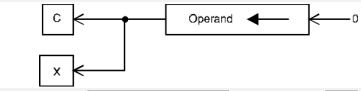
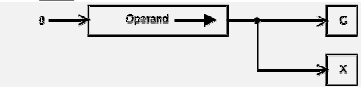
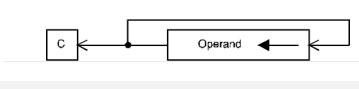
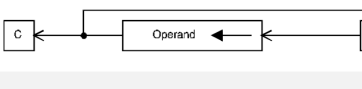
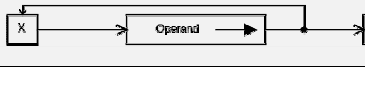
	X	N	Z	V	C		
TAS	–	*	*	0	0	[destination] - 0 (update CCR only);	Indivisible and uses a read-modify-write cycle
						[destination(7)] ⟵ 1	

ASL / ASR / LSL / LSR / ROL / ROR / ROXL / ROXR

DESTINATION

SHIFT COUNT	dX dX	#$ dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
0	6/6/8	--	--	--	--	--	--	--	--	--	--	--	--
1	8/8/10	8/8/10	--	--/12/--	--/12/--	--/14/--	--/16/--	--/18/--	--/16/--	--/20/--	--	--	--
2	10/10/12	10/10/12	--	--	--	--	--	--	--	--	--	--	--
3	12/12/14	12/12/14	--	--	--	--	--	--	--	--	--	--	--
4	14/14/16	14/14/16	--	--	--	--	--	--	--	--	--	--	--
5	16/16/18	16/16/18	--	--	--	--	--	--	--	--	--	--	--
6	18/18/20	18/18/20	--	--	--	--	--	--	--	--	--	--	--
7	20/20/22	20/20/22	--	--	--	--	--	--	--	--	--	--	--
8	22/22/24	22/22/24	--	--	--	--	--	--	--	--	--	--	--
9	24/24/26	--	--	--	--	--	--	--	--	--	--	--	--
10	26/26/28	--	--	--	--	--	--	--	--	--	--	--	--
11	28/28/30	--	--	--	--	--	--	--	--	--	--	--	--
12	30/30/32	--	--	--	--	--	--	--	--	--	--	--	--
13	32/32/34	--	--	--	--	--	--	--	--	--	--	--	--
14	34/34/36	--	--	--	--	--	--	--	--	--	--	--	--
15	36/36/38	--	--	--	--	--	--	--	--	--	--	--	--
16	38/38/40	--	--	--	--	--	--	--	--	--	--	--	--
17	40/40/42	--	--	--	--	--	--	--	--	--	--	--	--
18	42/42/44	--	--	--	--	--	--	--	--	--	--	--	--
19	44/44/46	--	--	--	--	--	--	--	--	--	--	--	--
20	46/46/48	--	--	--	--	--	--	--	--	--	--	--	--
21	48/48/50	--	--	--	--	--	--	--	--	--	--	--	--
22	50/50/52	--	--	--	--	--	--	--	--	--	--	--	--
23	52/52/54	--	--	--	--	--	--	--	--	--	--	--	--
24	54/54/56	--	--	--	--	--	--	--	--	--	--	--	--
25	56/56/58	--	--	--	--	--	--	--	--	--	--	--	--
26	58/58/60	--	--	--	--	--	--	--	--	--	--	--	--
27	60/60/62	--	--	--	--	--	--	--	--	--	--	--	--
28	62/62/64	--	--	--	--	--	--	--	--	--	--	--	--
29	64/64/68	--	--	--	--	--	--	--	--	--	--	--	--
30	68/68/70	--	--	--	--	--	--	--	--	--	--	--	--
31	70/70/72	--	--	--	--	--	--	--	--	--	--	--	--
--63	+2 for each	--	--	--	--	--	--	--	--	--	--	--	--

NOTES

	X	N	Z	V	C		
ASL	*	*	*	*	*		Arithmetic Shift Left; MSB shifted into both X-and C-bits; V-bit set if sign change during shifting
ASR	*	*	*	0	*		Arithmetic Shift Right; LSB shifted into both X-and C-bits; MSB replicated to preserve sign of number
LSL	*	*	*	0	*		Logical Shift Left; MSB shifted into both X- and C-bits; zero shift count leaves X-bit unaffected and C-bit clear
LSR	*	*	*	0	*		Logical Shift Right; LSB shifted into both X- and C-bits; zero shift count leaves X-bit unaffected and C-bit clear
ROL	–	*	*	0	*		ROtate Left; circular rotate; MSB shifted into C-bit; zero shift count leaves C-bit clear
ROR	–	*	*	0	*		ROtate Right; circular rotate; LSB shifted into C-bit; zero shift count leaves C-bit clear
ROXL	*	*	*	0	*		ROtate Left with eXtend; MSB shifted into both X- and C-bits; zero count leaves X-bit unaffected and C-bit clear
ROXR	*	*	*	0	*		ROtate Right with eXtend; LSB shifted into both X- and C-bits; zero count leaves X-bit unaffected and C-bit clear

BSR / JMP / JSR / LEA / PEA

SOURCE \ DESTINATION	BSR : pc	JMP : pc	JSR : pc	LEA : aX	PEA : -(sp)
dX	--	--	--	--	--
aX	--	--	--	--	--
(aX)	--	--/--/8	--/--/16	--/--/4	--/--/12
(aX)+	--	--	--	--	--
-(aX)	--	--	--	--	--
$(aX)	--	--/--/10	--/--/18	--/--/8	--/--/16
$(aX,X)	--	--/--/14	--/--/22	--/--/12	--/--/20
(add).w	--	--/--/10	--/--/18	--/--/8	--/--/16
(add).l	--	--/--/12	--/--/20	--/--/12	--/--/20
$(pc)	18/18/--	--/--/10	--/--/18	--/--/8	--/--/16
$(pc,X)	--	--/--/14	--/--/22	--/--/12	--/--/20
#$	--	--	--	--	--

	X	N	Z	V	C		
BSR	–	–	–	–	–	[--[SP]] ← [PC]; [PC] ← [PC] + $	Branch to SubRoutine
JMP	–	–	–	–	–	[PC] ← <effective address>	JuMP (unconditionally)
JSR	–	–	–	–	–	[--[SP]] ← [PC]; [PC] ← <effective address>	Jump to SubRoutine
LEA	–	–	–	–	–	[aX] ← <effective address>	Load Effective Address
PEA	–	–	–	–	–	[--[SP]] ← <effective address>	Push Effective Address

LINK

16 (some state 18)

UNLK

12

	X	N	Z	V	C		
LINK aX, #$.w	–	–	–	–	–	[--[SP]] ← [aX]; [aX] ← [SP]; [SP] ← [SP] + $	Use negative $ to allocate stack area; a6 used by convention
UNLK aX	–	–	–	–	–	[SP] ← [aX]; [aX] ← [[SP]++]	Collapses the stack frame created by LINK

RTE / RTR

20

RTS

16

	X	N	Z	V	C		
RTE	*	*	*	*	*	IF [S] == 1 THEN [SR] ← [[SP]++]; [PC] ← [[SP]++]	Used to terminate exception handler; CCR restored to its
						ELSE TRAP	pre-exception state
RTR	*	*	*	*	*	[CCR] ← [[SP]++]; [PC] ← [[SP]++]	CCR restored to pre-exception state
RTS	–	–	–	–	–	[PC] ← [[SP]++]	Use to terminate subroutine

Bcc

Taken	10/10/--
Not Taken	8/12/--

DBcc

Taken	--/10/--
Not Taken	--/12/--
Expired	--/14/--

Scc

DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
Taken	6/--/--	--	12/--/--	12/--/--	14/--/--	16/--/--	18/--/--	16/--/--	20/--/--	--	--	--
Not Taken	4/--/--	--	12/--/--	12/--/--	14/--/--	16/--/--	18/--/--	16/--/--	20/--/--	--	--	--

	X	N	Z	V	C		
Bcc	–	–	–	–	–	IF cc == 1 THEN [PC] ← [PC] + d	If TRUE, branch; BT and BRA same
DBcc	–	–	–	–	–	IF cc == 0 AND --[dX] != -1 THEN [PC] ← [PC] + d	If FALSE, decrement then branch; DBF and DBRA same
Scc	–	–	–	–	–	IF cc == 1 THEN [destination] ← $FF ELSE $00	If condition TRUE, set destination else reset destination

Encoding	Type	Mnemonic	Condition	Test
0	--	T	True (i.e., always)	1
1	--	F	False (i.e., never)	0
2	Unsigned	HI	HIgher than	C | Z == 0
3	Unsigned	LS	Lower than or Same	C | Z == 1
4	Unsigned	CC / HS	Carry Clear / Higher than or Same	C == 0
5	Unsigned	CS / LO	Carry Set / LOwer than	C == 1
6	--	NE	Not Equal	Z == 0
7	--	EQ	EQual	Z == 1
8	--	VC	oVerflow Clear	V == 0
9	--	VS	oVerflow Set	V == 1
10	--	PL	PLus	N == 0
11	--	MI	MInus	N == 1
12	Signed	GE	Greater than or Equal	N ^ V == 0
13	Signed	LT	Less Than	N ^ V == 1
14	Signed	GT	Greater Than	(N ^ V) | Z == 0
15	Signed	LE	Less than or Equal	(N ^ V) | Z == 1

DIVS

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	120-156	--	--	--	--	--	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	124-160	--	--	--	--	--	--	--	--	--	--	--
(aX)+	124-160	--	--	--	--	--	--	--	--	--	--	--
-(aX)	126-162	--	--	--	--	--	--	--	--	--	--	--
$(aX)	128-164	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	130-166	--	--	--	--	--	--	--	--	--	--	--
(add).w	128-164	--	--	--	--	--	--	--	--	--	--	--
(add).l	132-168	--	--	--	--	--	--	--	--	--	--	--
$(pc)	128-164	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	130-166	--	--	--	--	--	--	--	--	--	--	--
#$	124-160	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
DIVS	–	*	*	*	0	[destination(0:15)] ⟵ [destination(0:31)] / [source(0:15)] [destination(16:31)] ⟵ remainder	Signed divide; division by 0 causes exception; sign of remainder always same as sign of dividend (unless remainder 0); if upper word of dividend is greater than or equal to divisor, V-bit is set and operands unaffected (16-18 cycles).

DIVU

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	76-136	--	--	--	--	--	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	80-140	--	--	--	--	--	--	--	--	--	--	--
(aX)+	80-140	--	--	--	--	--	--	--	--	--	--	--
-(aX)	82-142	--	--	--	--	--	--	--	--	--	--	--
$(aX)	84-144	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	86-146	--	--	--	--	--	--	--	--	--	--	--
(add).w	84-144	--	--	--	--	--	--	--	--	--	--	--
(add).l	88-148	--	--	--	--	--	--	--	--	--	--	--
$(pc)	84-144	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	86-146	--	--	--	--	--	--	--	--	--	--	--
#$	80-140	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
DIVU	–	*	*	*	0	[destination(0:15)] ⟵ [destination(0:31)] / [source(0:15)] [destination(16:31)] ⟵ remainder	Unsigned divide; division by 0 causes exception; if upper word of dividend is greater than or equal to divisor, V-bit is set and operands unaffected (10 cycles).

MULS / MULU

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
dX	38-70	--	--	--	--	--	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	42-74	--	--	--	--	--	--	--	--	--	--	--
(aX)+	42-74	--	--	--	--	--	--	--	--	--	--	--
-(aX)	44-76	--	--	--	--	--	--	--	--	--	--	--
$(aX)	46-78	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	48-80	--	--	--	--	--	--	--	--	--	--	--
(add).w	46-78	--	--	--	--	--	--	--	--	--	--	--
(add).l	50-82	--	--	--	--	--	--	--	--	--	--	--
$(pc)	46-78	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	48-80	--	--	--	--	--	--	--	--	--	--	--
#$	42-74	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
MULS	–	*	*	0	0	[destination(0:31)] ⟵ [destination(0:15)] * [source(0:15)]	Signed multiply; Result is a 32-bit longword; Base cycle count is 38 + 2 * n; concatenate a zero in the LSB position to make a 17-bit source, n is the number of bit changes (occurrences of 0 ⟶ 1, or 1 ⟶ 0) (e.g., $5555 ⟶ %0 10101010 10101010 = 16 changes, 38 + 2 * 16 = 70 cycles)
MULU	–	*	*	0	0	[destination(0:31)] ⟵ [destination(0:15)] * [source(0:15)]	Unsigned multiply; Result is a 32-bit longword; Base cycle count is 38 + 2 * n; n = number of ones in source (e.g., $FFFF ⟶ %11111111 11111111 = 16 ones, 38 + 2 * 16 = 70 cycles)

CHK

SOURCE \ DESTINATION	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
dX	--/44/--	--	--	--	--	--	--	--	--	--	--	--
aX	--	--	--	--	--	--	--	--	--	--	--	--
(aX)	--/48/--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--/48/--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--/50/--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--/52/--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--/54/--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--/52/--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--/56/--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--/52/--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--/54/--	--	--	--	--	--	--	--	--	--	--	--
#$	--/48/--	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
CHK	–	*	U	U	U	IF [dX] < 0 OR [dX] > [source] THEN TRAP	Upper bound is a two's complement integer; Exception handling address uses vector 6, address $018

ILLEGAL / TRAP

	34

TRAPV

Trap	34
Continue	4

	X	N	Z	V	C		
ILLEGAL	–	–	–	–	–	[--[SSP]] ← [PC]; [--[SSP]] ← [SR]; [PC] ← Illegal instruction vector (16)	Any unknown pattern of bits read as an instruction causes illegal instruction trap
TRAP #<vector>	–	–	–	–	–	S ← 1; [--[SSP]] ← [PC]; [--[SSP]] ← [SR]; [PC] ← vector (33-47)	Use to perform system independent operating system calls
TRAPV	–	–	–	–	–	IF V == 1 THEN [--[SSP]] ← [PC]; [--[SSP]] ← [SR]; [PC] ← [$01C] (vector 7) ELSE continue	Used to call operating system if overflow occurs; exception vector is located at address $01C

RESET

132

	X	N	Z	V	C		
RESET	–	–	–	–	–	IF [S] == 1 THEN Assert RESET* line ELSE TRAP	Used to reset all peripherals connected to 68000's RESET* pin; Privileged instruction; does not affect 68000's operation

STOP

--/4/--

	X	N	Z	V	C		
STOP #<data>	*	*	*	*	*	IF [S] == 1 THEN [SR] ← <data>; STOP ELSE TRAP	Status register updated and processor halted; Resumes when trace, interrupt, or reset exception occurs; Trace exception occurs if trace bit set when STOP encountered; Interrupt exception occurs if interrupt requested with higher priority than current; A privilege violation occurs if S-bit is cleared by STOP; Reset exception occurs with external reset

CLR / NEG / NEGX / NOT

DESTINATION

	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
	4/4/6	--	12/12/20	12/12/20	14/14/22	16/16/24	18/18/26	16/16/24	20/20/28	--	--	--

	X	N	Z	V	C		
CLR	–	0	1	0	0	[destination] ⟵ 0	Read from memory occurs before write
NEG	*	*	*	*	*	[destination] ⟵ 0 - [destination]	2's complement; X-bit set to value of C-bit
NEGX	*	*	*	*	*	[destination] ⟵ 0 - [destination] - [X]	2's complement; X-bit set to value of C-bit;
							Z-bit cleared if non-zero, unchanged otherwise
NOT	–	*	*	0	0	[destination] ⟵ 0 - [destination] - 1	1's complement

NBCD

DESTINATION

	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
	6/--/--	--	12/--/--	12/--/--	14/--/--	16/--/--	18/--/--	16/--/--	20/--/--	--	--	--

	X	N	Z	V	C		
NBCD	*	U	*	U	*	[destination]$_{10}$ ⟵ 0 - [destination]$_{10}$ - [X]	Z-bit cleared if non-zero, left unchanged otherwise

EXG

DESTINATION

SOURCE	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
dX	--/--/6	--/--/6	--	--	--	--	--	--	--	--	--	--
aX	--/--/6	--/--/6	--	--	--	--	--	--	--	--	--	--
(aX)	--	--	--	--	--	--	--	--	--	--	--	--
(aX)+	--	--	--	--	--	--	--	--	--	--	--	--
-(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX)	--	--	--	--	--	--	--	--	--	--	--	--
$(aX,X)	--	--	--	--	--	--	--	--	--	--	--	--
(add).w	--	--	--	--	--	--	--	--	--	--	--	--
(add).l	--	--	--	--	--	--	--	--	--	--	--	--
$(pc)	--	--	--	--	--	--	--	--	--	--	--	--
$(pc,X)	--	--	--	--	--	--	--	--	--	--	--	--
#$	--	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C	
EXG	–	–	–	–	–	[Rx] ⟵ [Ry]; [Ry] ⟵ [Rx]

EXT

DESTINATION

	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	sp
	--/4/4	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C	WORD	LONG
EXT	–	*	*	0	0	[destination(8:15)] ⟵ [destination(7)]	[destination(16:31)] ⟵ [destination(15)]

SWAP

DESTINATION

	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	usp
	--/4/--	--	--	--	--	--	--	--	--	--	--	--

	X	N	Z	V	C		
SWAP	–	*	*	0	0	[register (16:31)] ⟵ [register(0:15)];	N-bit and Z-bit set to 32 bit result
						[register (0:15)] ⟵ [register (16:31)]	

TST

DESTINATION

	dX	aX	(aX)	(aX)+	-(aX)	$(aX)	$(aX,X)	(add).w	(add).l	ccr	sr	Usp
	4/4/4	--	8/8/12	8/8/12	10/10/14	12/12/16	14/14/18	12/12/16	16/16/20	--	--	--

	X	N	Z	V	C		
TST	–	*	*	0	0	[destination] - 0 (update CCR only)	Identical to CMPI #0

NOP

4

	X	N	Z	V	C	
NOP	–	–	–	–	–	No Operation