Opcode	Mnemonic	BWL	Params	XNZVC	Ref #s	68000
1100REx10000MREy	ABDC	B--	src, dest	*?*?*	20	6R, 18M
1101REnOPMEAMEAR	ADD	BWL	src, dest	*****		4/6*R, 8/12M +EA
1101REnS11EAMEAR	ADDA	-WL	src, An	-----		8/6 +EA
00000110SIEAMEAR	ADDI	BWL	#imm, dest	*****		8/16R, 12/20M +EA
0101NUM0SIEAMEAR	ADDQ	BWL	#imm, dest	*****	01	4~/8R, 8/12M +EA
1101REx1SI00MREy	ADDX	BWL	src, dest	*****		4/8R, 18/30M
1100REnOPMEAMEAR	AND	BWL	src, dest	-**00		4/6*R, 8/12M +EA
00000010SIEAMEAR	ANDI	BWL	#imm, dest	-**00		8/16R, 12/20M +EA
0000001000111100	ANDI	B--	#imm, CCR	*****		20
0000001001111100	ANDI	-W-	#imm, SR	*****	10	
1110CRe1SIi00REd	ASL	BWL	src, Dn	*****		6+2n/8+2nR, 8M +EA
1110000111EAMEAR	ASL	-W-	dest	*****		8M +EA
1110CRe0SIi00REd	ASR	BWL	src, Dn	***0*		6+2n/8+2nR, 8M +EA
1110000011EAMEAR	ASR	-W-	dest	***0*		8M +EA
0110COND8bitdisp	Bcc	BWL	*label*	-----	02	
0000REn101EAMEAR	BCHG	B-L	Dn, dest	--*--	03	
0000100001EAMEAR	BCHG	B-L	#imm, dest	--*--	03	
0000REn110EAMEAR	BCLR	B-L	Dn, dest	--*--	03	
0000100010EAMEAR	BCLR	B-L	#imm, dest	--*--	03	
1110101011EAMEAR	BFCHG	---	dest{off:wid}	-**00	21	-
1110110011EAMEAR	BFCLR	---	dest{off:wid}	-**00	21	-
1110101111EAMEAR	BFEXTS	---	src{off:wid}, Dn	-**00	21	-
1110100111EAMEAR	BFEXTU	---	src{off:wid}, Dn	-**00	21	-
1110110111EAMEAR	BFFFO	---	src{off:wid}, Dn	-**00	21	-
1110111111EAMEAR	BFINS	---	Dn, dest{off:wid}	-**00	21	-
1110111011EAMEAR	BFSET	---	dest{off:wid}	-**00	21	-
1110100011EAMEAR	BFTST	---	src{off:wid}	-**00	21	-
0100100001001VEC	BKPT	---	#imm	-----		-
011000008bitdisp	BRA	BWL	*label*	-----	02	
0000REn111EAMEAR	BSET	B-L	Dn, dest	--*--	03	
0000100011EAMEAR	BSET	B-L	#imm, dest	--*--	03	
011000018bitdisp	BSR	BWL	*label*	-----	02	
0000REn100EAMEAR	BTST	B-L	Dn, dest	--*--	03	
0000100000EAMEAR	BTST	B-L	#imm, dest	--*--	03	
0000011011EAMEAR	CALLM	---	#imm, src	-----	04	-
00001SI011EAMEAR	CAS	BWL	Du, Dc, dest	-****	05,12	-
00001SI011111100	CAS2	-WL	Dc1:Dc2,Du1:Du2,(Rn1):(Rn2)	-****	05,12	-
0100REnSI0EAMEAR	CHK	-WL	src, Dn	-*???	02,06	10 +EA
00000SI011EAMEAR	CHK2	BWL	src, Rn	-?*?*	05,06	-
01000010SIEAMEAR	CLR	BWL	dest	-0100		
1011REn1SIEAMEAR	CMP	BWL	src, Dn	-****		4/6 +EA
00000SI011EAMEAR	CMP2	BWL	src, Dn	-?*?*	05	-
1011REnOPMEAMEAR	CMPA	-WL	src, An	-****		6/6 +EA
00001100SIEAMEAR	CMPI	BWL	#imm, src	-****		
1011RAx1SI001RAy	CMPM	BWL	(Ay)+, (Ax)+	-****		12/20
1111CID01SCOCOND	cpBcc	-WL	*label*	-----	04,13	-
1111CID001001REn	cpDBcc	-W-	Dn, *label*	-----	04,13	-
1111CID000EAMEAR	cpGEN	---	{coprocessor specific}	*****	04,13	-
1111CID001EAMEAR	cpScc	B--	dest	-----	04,13	-

1111CID0011110PM	cpTRAPcc	-WL	#imm	-----	04,13	-
0101COND11001REn	DBcc	-W-	Dn, *label*	-----		
1000REn111EAMEAR	DIVS	-W-	src, Dn	-***0	06	???-158# +EA
0100110001EAMEAR	DIVS	--L	src, Dq/Dr:Dq	-***0	05,06	-
0100110001EAMEAR	DIVSL	--L	src, Dr:Dq	-***0	05,06	-
1000REn011EAMEAR	DIVU	-W-	src, Dn	-***0	06	???-140# +EA
0100110001EAMEAR	DIVU	--L	src, Dq/Dr:Dq	-***0	05,06	-
0100110001EAMEAR	DIVUL	--L	src, Dr:Dq	-***0	05,06	-
1011REn1SIEAMEAR	EOR	BWL	Dn, dest	-**00		4/8R, 8/12M +EA
00001010SIEAMEAR	EORI	BWL	#imm, dest	-**00		
0000101000111100	EORI	B--	#imm, CCR	*****		20
1100REx1SPOPMREy	EXG	--L	Rx, Ry	-----		6
01001000SI000REn	EXT	-WL	Rn	-**00		4
0100100111000REn	EXTB	--L	Rn	-**00		-
0100101011111100	ILLEGAL	---		-----		
0100111011EAMEAR	JMP	---	<ea>	-----		
0100111010EAMEAR	JSR	---	<ea>	-----		
0100REn111EAMEAR	LEA	--L	<ea>, An	-----		
0100111001010REn	LINK	-W-	An, #imm	-----		16
0100100000001REn	LINK	--L	An, #imm	-----		-
1110CRe1SIi01REd	LSL	BWL	src, Dn	***0*		
1110001111EAMEAR	LSL	-W-	dest	***0*		
1110CRe0SIi01REd	LSR	BWL	src, Dn	***0*		
1110001011EAMEAR	LSR	-W-	dest	*0*0*		
00SIDREDAMSAMSRE	MOVE	BWL	src, dest	-**00		
0100001011EAMEAR	MOVE	-W-	CCR, dest	-----		-
0100000011EAMEAR	MOVE	-W-	SR, dest	-----	11	
0100010011EAMEAR	MOVE	-W-	src, CCR	*****		
0100011011EAMEAR	MOVE	-W-	src, SR	*****	10	
1111011000100REs	MOVE16	---	Ax, Ay	-----	05	-
111101100000OPREn	MOVE16	---	src, dest	-----		-
00SIREd001EAMEAR	MOVEA	-WL	src, An	-----		
010010001SEAMEAR	MOVEM	-WL	{reglist}, dest	-----		
010011001SEAMEAR	MOVEM	-WL	src, {reglist}	-----		
0000REx10S001REy	MOVEP	-WL	(d16,Ay), Dx	-----		16/24
0000REx11S001REy	MOVEP	-WL	Dx, (d16,Ay)	-----		16/24
0111REn08bitvalu	MOVEQ	--L	#imm8, Dn	-**00		
1100REn111EAMEAR	MULS	-W-	src, Dn	-**00		38-70# +EA
0100110000EAMEAR	MULS	--L	src, Dl/Dh-Dl	-***0	05	-
1100REn011EAMEAR	MULU	-W-	src, Dn	-**00		38-70# +EA
0100110000EAMEAR	MULU	--L	src, Dl/Dh-Dl	-***0	05	-
0100100000EAMEAR	NBCD	B--	dest	*?*?*	20	6R, 8M +EA
01000100SIEAMEAR	NEG	BWL	dest	*****		4/6R, 8/12M +EA
01000000SIEAMEAR	NEGX	BWL	dest	*****		
0100111001110001	NOP	---		-----	14	
01000110SIEAMEAR	NOT	BWL	dest	-**00		
1000REnOPMEAMEAR	OR	BWL	src, dest	-**00		
00000000SIEAMEAR	ORI	BWL	#imm, dest	-**00		
0000000000111100	ORI	B--	#imm, CCR	*****		
1000REy10100MREx	PACK	---	src, dest, #imm	-----		-

0100100001EAMEAR	PEA	--L	src	-----		
1110CRe1SIi11REn	ROL	BWL	src, Dn	-**0*		
1110011111EAMEAR	ROL	-W-	dest	-**0*		
1110CRe0SIi11REn	ROR	BWL	src, Dn	-**0*		
1110011011EAMEAR	ROR	-W-	dest	-**0*		
1110CRe1SIi10REn	ROXL	BWL	src, Dn	***0*		
1110010111EAMEAR	ROXL	-W-	dest	***0*		
1110CRe0SIi10REn	ROXR	BWL	src, Dn	***0*		
1110010011EAMEAR	ROXR	-W-	dest	***0*		
0100111001110100	RTD	---	#imm16	-----		-
0000011011100REn	RTM	---	Rn	*****	04	-
0100111001110111	RTR	---		*****		
0100111001110101	RTS	---		-----		
1000REy10000MREx	SBCD	B--	src, dest	*?*?*	20	
0101COND11EAMEAR	Scc	B--	dest	-----		
1001REnOPMEAMEAR	SUB	BWL	src, dest	*****		
1001REnS11EAMEAR	SUBA	-WL	src, dest	-----		
00000100SIEAMEAR	SUBI	BWL	#imm, dest	*****		
1010NUM1SIEAMEAR	SUBQ	BWL	#imm, dest	*****		
1001REy1SI00MREx	SUBX	BWL	src, dest	*****		
0100100001000REn	SWAP	-W-	Dn	-**00		
0100101011EAMEAR	TAS	B--	dest	-**00	12	
0100111001000VECT	TRAP	---	#imm	-----		

Ref #	Description
01	If an address register is used as a destination, then a long operation occurs and the condition codes are not affected.
02	Not all sizes are supported by all processors. See the specific opcode definition for more details.
03	Not all sizes are supported by all addressing modes. See the specific opcode definition for more details.
04	This opcode is not upwardly compatible (some or all future processors do not support it) and use of it is not recommended
05	This opcode has a custom extension word which may contain extra data that distinguishes it from other instructions with the same encoding
06	This opcode may throw an exception. See opcode details for information.
10	This is a supervisor-only operation. Using it in User Mode will cause an invalid instruction exception.
11	This is a supervisor-only operation except on the 68000 (not including the 68EC000) and 68008 processors.
12	This opcode guarantees that it will not be interrupted by other processors by locking memory accessed during it's operation
13	This instruction is designed specifically to work with a co-processor.
14	On processors that support multiple pipelines, this instruction causes the processor to wait for all pipelines/bus cycles to complete before executing
20	This is a BCD operation. See documentation for the instruction itself for details.
21	This is a bit field operation. See documentation for the instruction itself for details.

When two clock values are listed, separated by a dash (-), the amount of time required for this opcode is variable and falls between the two values specified.

When two values are separated by a /, the left side will be for Byte and Word operations and the right side will be for Long operations

A timing value with an R means it is using a Register as a destination. A value with an M means it is using a Memory value as a destination. A value with a T means a condition was True. A value with an F means a condition was false

Specials

68000

EA

Description	Clocks (r/w)	
	Byte/Word	Long
Dn	0 (0/0)	0 (0/0)
An	0 (0/0)	0 (0/0)
(An)	4 (1/0)	8 (2/0)
(An)+	4 (1/0)	8 (2/0)
-(An)	6 (1/0)	10 (2/0)
d(An)	8 (2/0)	12 (3/0)
d(An,ix)	10 (2/0)	14 (3/0)
xxx.W	8 (2/0)	12 (3/0)
xxx.L	12 (3/0)	16 (4/0)
d(PC)	8 (2/0)	12 (3/0)
d(PC,ix)	10 (2/0)	14 (3/0)
#xxx	4 (1/0)	8 (2/0)

*****

The base time for this opcode is increased by 2 if the source operand is Dn or #imm

~

The base time for this opcode is increased by 4 if the destination operand is An

#

DIVS/DIVU: The internal algorithm used by the 68000 has less than a 10% difference between the best case and worst case timing for these instructions

MULS: The internal algorithm requires 38+2n clocks. To calculate n, take the 16bit <ea> value, append a 0 in the least significant byte so that it is a 17bit value. Then, for every 10 and 01 combination in the resulting value, increase n by 1. This means that if the 16bit <ea> value is $5555, n is 16.

MULU: The internal algorithm requires 38+2n clocks, with n as the number of bits set in the <ea> value