TABLE C.4
68000 instruction set

Mnemonic (Name)	Size	Addressing mode		Addressing mode												
				Dn	An	(An)	(An)+	−(An)	d(An)	d(An,Xi)	Abs.W	Abs.L	d(PC)	d(PC,Xi)	Immed	SR or CCR
ABCD (Add BCD)	B	s = Dn	d =	x												
		s = −(An)	d =					x								
ADD (Add)	B,W,L	s = Dn	d =	x		x	x	x	x	x	x	x				
		d = Dn	s =	x	x	x	x	x	x	x	x	x	x	x	x	
ADD A (Add address)	W	d = An	s =	x	x	x	x	x	x	x	x	x	x	x	x	
	L	d = An	s =	x	x	x	x	x	x	x	x	x	x	x	x	
ADDI (Add immediate)	B,W,L	s = Immed	d =	x		x	x	x	x	x	x	x				
ADDQ (Add quick)	B,W,L	s = Immed3	d =	x	x	x	x	x	x	x	x	x				
ADD X (Add extended)	B,W,L	s = Dn	d =	x												
		s = −(An)	d =					x								
AND (Logical AND)	B,W,L	s = Dn	d =	x		x	x	x	x	x	x	x				
		d = Dn	s =	x		x	x	x	x	x	x	x	x	x	x	
ANDI (AND immediate)	B,W,L	s = Immed	d =	x		x	x	x	x	x	x	x				x
ASL (Arithmetic shift left)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
		count = 1	d =			x	x	x	x	x	x	x				
ASR (Arithmetic shift right)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
		count = 1	d =			x	x	x	x	x	x	x				
BCHG* (Test a bit and change it)	B	bit# = [Dn]	d =			x	x	x	x	x	x	x				
		bit# = Immed	d =			x	x	x	x	x	x	x				
	L	bit# = [Dn]	d =	x												
		bit# = Immed	d =	x												
BCLR* (Test a bit and clear it)	B	bit# = [Dn]	d =			x	x	x	x	x	x	x				
		bit# = Immed	d =			x	x	x	x	x	x	x				
	L	bit# = [Dn]	d =	x												
		bit# = Immed	d =	x												

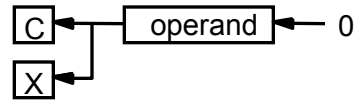
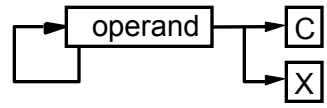
OP code $b_{15} \ldots b_0$	Operation performed	Condition flags X	N	Z	V	C
1100 RRR1 0000 0rrr 1100 RRR1 0000 1rrr	d ← [s]+[d]+[X] Binary-coded decimal addition	x	u	x	u	x
1101 DDD1 SSEE EEEE 1101 DDD0 SSeeeeee	d ← [Dn]+[d] Dn ← [s]+[Dn]	x x	x x	x x	x x	x x
1101 AAA0 11eeeeee 1101 AAA1 11eeeeee	An ← [s]+[An]					
0000 0110 SSEE EEEE	d ← s+[d]	x	x	x	x	x
0101 QQQ0 SSEE EEEE	d ← QQQ+[d]	x	x	x	x	x
1101 RRR1 SS00 0rrr 1101 RRR1 SS00 1rrr	d ← [s]+[d]+[X] Multiprecision addition	x	x	x	x	x
1100 DDD1 SSEE EEEE 1100 DDD0 SSeeeeee	d ← [Dn] ^ [d]		x	x	0	0
0000 0010 SSEE EEEE	d ← s ^ [d]		x	x	0	0
1110 rrr1 SS10 0DDD 1110 QQQ1 SS000DDD 1110 0001 11EE EEEE		x	x	x	x	x
1110 rrr0 SS10 0DDD 1110 QQQ0 SS000DDD 1110 0000 11EE EEEE		x	x	x	x	x
0000 rrr1 01EE EEEE 0000 1000 01EE EEEE 0000 rrr1 01EE EEEE 0000 1000 01EE EEEE	Z ← $\overline{\text{(bit\# of d)}}$; then complement the tested bit in d.			x		
0000 rrr1 10EE EEEE 0000 1000 10EE EEEE 0000 rrr1 10EE EEEE 0000 1000 10EE EEEE	Z ← $\overline{\text{(bit\# of d)}}$; then clear the tested bit in d.			x		

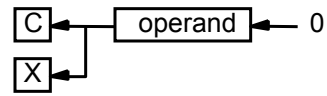
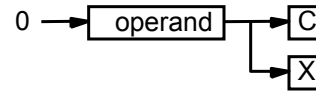
TABLE C.4
(*Continued*)

Mnemonic (Name)	Size	Addressing mode	Addressingmode												
			Dn	An	(An)	(An) +	−(An)	d(An)	d(An,Xi)	Abs.W	Abs.L	d(PC)	d(PC,Xi)	Immed	SR or CCR
BSET* (Test a bit and set it)	B	bit# = [Dn] d =			x	x	x	x	x	x	x				
		bit# = Immed d =			x	x	x	x	x	x	x				
	L	bit# = [Dn] d =	x												
		bit# = Immed d =	x												
BTST* (Test a bit)	B	bit# = [Dn] d =			x	x	x	x	x	x	x				
		bit# = Immed d =			x	x	x	x	x	x	x				
	L	bit# = [Dn] d =	x												
		bit# = Immed d =	x												
CHK (Checkregister)	W	d = Dn s =	x		x	x	x	x	x	x	x	x	x	x	
CLR (Clear)	B,W,L	d =	x		x	x	x	x	x	x	x				
CMP (Compare)	B,W,L	d = Dn s =	x	x	x	x	x	x	x	x	x	x	x	x	
CMPA (Compareaddress)	W	d = An s =	x	x	x	x	x	x	x	x	x	x	x	x	
	L	d = An s =	x	x	x	x	x	x	x	x	x	x	x	x	
CMPI (Compareimmed)	B,W,L	s = Immed d =	x		x	x	x	x	x	x	x				
CMPM (Comparememory)	B,W,L	s = (An)+ d =				x									
DIVS (Divide signed)	W	d = Dn s =	x		x	x	x	x	x	x	x	x	x	x	
DIVU (Divide unsigned)	W	d = Dn s =	x		x	x	x	x	x	x	x	x	x	x	
EOR (Logical XOR)	B,W,L	s = Dn d =	x		x	x	x	x	x	x	x				
EORI (XOR immediate)	B,W,L	s = Immed d =	x		x	x	x	x	x	x	x				x
EXG (Exchange	L	s = Dn d =	x	x											
		s = An d =	x	x											
EXT (Sign extend)	W	d =	x												
	L	d =	x												

OP code $b_{15} \cdots b_0$	Operation performed	Condition flags X	N	Z	V	C
0000rrr1 11EE EEEE 00001000 11EE EEEE 0000rrr1 11EE EEEE 00001000 11EE EEEE	Z ← $\overline{\text{(bit\# of d)}}$; then set to 1 the tested bit in d.			x		
0000rrr1 00EE EEEE 00001000 00EE EEEE 0000rrr1 00EE EEEE 00001000 00EE EEEE	Z ← $\overline{\text{(bit\# of d)}}$;			x		
0100DDD1 10eeeeee	If [Dn] < 0 or [Dn] > [s], then raise an interrupt.		x	u	u	u
01000010 SSEEEEEE	d ← 0		0	1	0	0
1011DDD0 SSeeeeee	[d] – [s]		x	x	x	x
1011AAA0 11eeeeee 1011AAA1 11eeeeee	[An] – [s]		x	x	x	x
00001100 SSEEEEEE	[d] – [s]		x	x	x	x
1011RRR1 SS001rrr	[d] – [s]		x	x	x	x
1000DDD1 11eeeeee	d ← [d] ÷ [s], using 32 bits of d and 16 bits of s.		x	x	x	0
1000DDD0 11eeeeee	d ← [d] ÷ [s], using 32 bits of d and 16 bits of s.		x	x	x	0
1011rrr1 SSEEEEEE	d ← [Dn] ⊕ [d]		x	x	0	0
00001010 SSEEEEEE	d ← s ⊕ [d]		x	x	0	0
1100DDD1 01000DDD 1100AAA1 01001AAA 1100DDD1 10001AAA	[s] ↔ [d]					
01001000 10000DDD 01001000 11000DDD	(bits 15–8 of *d*) ← (bit 7 of *d*) (bits 31–16 of *d*) ← (bit 15 of *d*)		x x	x x	0 0	0 0

TABLE C.4
(*Continued*)

Mnemonic (Name)	Size	Addressing mode		Addressingmode												
				Dn	An	(An)	(An) +	−(An)	d(An)	d(An,Xi)	Abs.W	Abs.L	d(PC)	d(PC,Xi)	Immed	SR or CCR
JMP (Jump)			d=			x			x	x	x	x	x	x		
JSR (Jump to subroutine)			d =			x			x	x	x	x	x	x		
LEA (Load effective address)	L	d = An	s =			x			x	x	x	x	x	x		
LINK (Link and allocate)		disp = Immed	s =		x											
LSL (Logical shift left)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
	W	count = I	d =			x	x	x	x	x	x	x				
LSR (Logical shift right)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
	W	count = I	d =			x	x	x	x	x	x	x				
MOVE (Move)	B,W,L	s = Dn	d =	x		x	x	x	x	x	x	x				
		s = An	d =	x		x	x	x	x	x	x	x				
		s = (An)	d =	x		x	x	x	x	x	x	x				
		s = (An)+	d =	x		x	x	x	x	x	x	x				
		s = −(An)	d =	x		x	x	x	x	x	x	x				
		s = d(An)	d =	x		x	x	x	x	x	x	x				
		s = d(An,Xi)	d =	x		x	x	x	x	x	x	x				
		s = Abs.W	d =	x		x	x	x	x	x	x	x				
		s = Abs.L	d =	x		x	x	x	x	x	x	x				
		s = d(PC)	d =	x		x	x	x	x	x	x	x				
		s = d(PC,Xi)	d =	x		x	x	x	x	x	x	x				
		s = Immed	d =	x		x	x	x	x	x	x	x				
	W	d = CCR	s =	x		x	x	x	x	x	x	x	x	x	x	
		d = SR	s =	x		x	x	x	x	x	x	x	x	x	x	
		s = SR	d =	x		x	x		x	x	x	x				
	L	s = SP	d =		x											
		d = SP	s =		x											
MOVEA (Move address)	W,L	d = An	s =	x	x	x	x	x	x	x	x	x	x	x	x	

OP code $b_{15} \cdots b_0$	Operation performed	Condition flags X	N	Z	V	C
0100 1110 11EE EEEE	PC ← effective address of d					
0100 1110 10EE EEEE	SP ← [SP] – 4; [SP] ← [PC]; PC ← effective address of d					
0100 AAA1 11eeeeee	An ← effective address of s					
0100 1110 0101 0AAA	SP ← [SP] – 4; [SP] ← [An]; An ← [SP]; SP ← [SP] + disp					
1110 rrr1 SS10 1DDD 1110 QQQ1 SS001DDD 1110 0011 11EE EEEE		x	x	x	0	x
1110 rrr0 SS10 1DDD 1110 QQQ0 SS001DDD 1110 0010 11EE EEEE		x	x	x	0	x
00SSRRRM MMee eeee	d ← [s]		x	x	0	0
0100 0100 11eeeeee	CCR ← [bits 7-0 of s]	x	x	x	x	x
0100 0110 11eeeeee	SR ← [s]	x	x	x	x	x
0100 0000 11EE EEEE	d ← [SR]					
0100 1110 0110 1AAA	d ← [SP]					
0100 1110 0110 0AAA	SP ← [d]					
00SSAAA0 01eeeeee	An ← [s]					

Right side of page 3

TABLE C.4
(*Continued*)

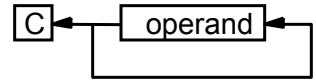
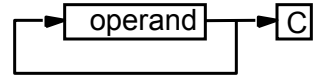
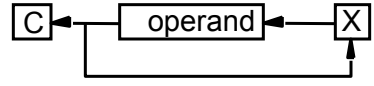
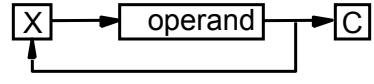
Mnemonic (Name)	Size	Addressing mode	Addressing mode												
			Dn	An	(An)	(An) +	− (An)	d(An)	d(An,Xi)	Abs.W	Abs.L	d(PC)	d(PC,Xi)	Immed	SR or CCR
MOVEM* (Move multiple registers)	W	s = Xn d =			x		x	x	x	x	x				
		d = Xn s =			x	x		x	x	x	x	x	x		
	L	s = Xn d =			x		x	x	x	x	x				
		d = Xn s =			x	x		x	x	x	x	x	x		
MOVEP* (Move peripheral data)	W	s = Dn d =						x							
	L	s = Dn d =						x							
	W	s = d(An) d =	x												
	L	s = d(An) d =	x												
MOVEQ (Move quick)	L	s = Immed8 d =	x												
MULS (Multiply signed)	W	d = Dn s =	x		x	x	x	x	x	x	x	x	x	x	
MULU (Multiply unsigned)	W	d = Dn s =	x		x	x	x	x	x	x	x	x	x	x	
NBCD (Negate BCD)	B	d =	x		x	x	x	x	x	x	x				
NEG (Negate)	B,W,L	d =	x		x	x	x	x	x	x	x				
NEGX (Negate extended)	B,W,L	d =	x		x	x	x	x	x	x	x				
NOP (No operation)															
NOT (Complement)	B,W,L	d =	x		x	x	x	x	x	x	x				
OR (Logical OR)	B,W,L	s = Dn d =			x	x	x	x	x	x	x				
		d = Dn s =	x		x	x	x	x	x	x	x	x	x	x	
ORI (OR immediate)	B,W,L	s = Immed d =	x		x	x	x	x	x	x	x				x
PEA (Push effective address)	L	s =			x			x	x	x	x	x	x		

OP code $b_{15} \cdots b_0$	Operation performed	Condition flags X	N	Z	V	C
0100 1000 10EE EEEE 0100 1100 10eeeeee 0100 1000 11EE EEEE 0100 1100 11eeeeee	d ← [Xn] Xn ← [s] d ← [Xn] Xn ← [s] A second word is used to specify the registers involved:					
0000 DDD1 1000 1AAA 0000 DDD1 1100 1AAA 0000 DDD1 0000 1AAA 0000 DDD1 0100 1AAA	Alternate bytes of d ← [Dn] Dn ← alternate bytes of d					
0111 DDD0 QQQQ QQQQ	Dn ← QQQQQQQQ		x	x	0	0
1100 DDD1 11ee eeee	Dn ← [s] × [Dn]		x	x	0	0
1100 DDD0 11ee eeee	Dn ← [s] × [Dn]		x	x	0	0
0100 1000 00EE EEEE	d ← 0 – [d] – [X] using BCD arithmetic	x	u	x	u	x
0100 0100 SSEE EEEE	d ← 0 – [d]	x	x	x	x	x
0100 0000 SSEE EEEE	d ← 0 – [d] – [X]	x	x	x	x	x
0100 1110 0111 0001	none					
0100 0110 SSEE EEEE	d ← $\overline{[d]}$		x	x	0	0
1000 DDD1 SSEE EEEE 1000 DDD0 SSeeeeee	d ← [s] ∨ [d]		x	x	0	0
0000 0000 SSEE EEEE	d ← s ∨ [d]		x	x	0	0
0100 1000 01eeeeee	SP ← [SP] – 4; [SP] ← effective address of s					

Right side of page 4

TABLE C.4
(*Continued*)

Mnemonic (Name)	Size	Addressingmode		Addressingmode												
				Dn	An	(An)	(An) +	– (An)	d(An)	d(An,Xi)	Abs.W	Abs.L	d(PC)	d(PC,Xi)	Immed	SR or CCR
RESET																
ROL (Rotate left without X)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
	W	count = 1	d =			x	x	x	x	x	x	x				
ROR (Rotate right without X)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
	W	count = 1				x	x	x	x	x	x	x				
ROXL (Rotate left with X)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
	W	count = 1	d =			x	x	x	x	x	x	x				
ROXR (Rotate right with X)	B,W,L	count = [Dn]	d =	x												
		count = QQQ	d =	x												
	W	count = 1	d =			x	x	x	x	x	x	x				
RTE (Return from exception)																
RTR (Return and restore CCR)																
RTS (Return from subroutine)																
SBCD (Subtract BCD)	B	s = Dn	d =	x												
		s = – (An)	d =					x								
Scc (Set on condition)	B		d =	x		x	x	x	x	x	x	x				
STOP (Load SR and stop)			s =												x	

OP code $b_{15} \cdots b_0$	Operation performed	Condition flags				
		X	N	Z	V	C
01001110 0111 0000	Assert RESET output line.					
1110 rrr1 SS11 1DDD 1110 QQQ1 SS011DDD 1110 0111 11EE EEEE			x	x	0	x
1100 rrr1 SS11 1DDD 1110 QQQ0 SS011DDD 1110 0111 11EE EEEE			x	x	0	x
1110 rrr1 SS11 0DDD 1110 QQQ1 SS010DDD 1110 0101 11EE EEEE		x	x	x	0	x
1110 rrr0 SS11 0DDD 1110 QQQ0 SS010DDD 1110 0100 11EE EEEE		x	x	x	0	x
01001110 0111 0011	SR ← [[SP]]; SP ← [SP] + 2; PC ← [[SP]]; SP ← [SP] + 4;	x	x	x	x	x
01001110 0111 0111	CCR ← [[SP]]; SP ← [SP] + 2; PC ← [[SP]]; SP ← [SP] + 4;	x	x	x	x	x
01001110 0111 0101	PC ← [[SP]]; SP ← [SP] + 4					
1000RRR1 0000 0rrr 1000RRR1 0000 1rrr	d ← [d] – [s] – [X] Binary-coded decimal subtraction	x	u	x	u	x
0101CCCC 11EE EEEE	Set all 8 bits of d to 1 if cc is true; otherwise, clear them to 0.					
01001110 0111 0010	SR ← s; wait for interrupt.	x	x	x	x	x

Right side of page 5

TABLE C.4
(*Continued*)

Mnemonic (Name)	Size	Addressingmode		Addressingmode												
				Dn	An	(An)	(An) +	– (An)	d(An)	d(An,Xi)	Abs.W	Abs.L	d(PC)	d(PC,Xi)	Immed	SR or CCR
SUB (Subtract)	B,W,L	s = Dn	d =	x		x	x	x	x	x	x	x				
		d = Dn	s =	x	x	x	x	x	x	x	x	x	x	x	x	
SUBA (Subtractaddress)	W	d = An	s =	x	x	x	x	x	x	x	x	x	x	x	x	
	L	d = An	s =	x	x	x	x	x	x	x	x	x	x	x	x	
SUBI (Subtractimmed)	B,W,L	s = Immed	d =	x		x	x	x	x	x	x	x				
SUBQ (Subtractquick)	B,W,L	s = Immed3	d =	x	x	x	x	x	x	x	x	x				
SUBX (Subtract extended)	B,W,L	s = Dn	d =	x												
		s = – (An)	d =					x								
SWAP (Swap register halves)	W		d =	x												
TAS (Test and set)	B		d =	x		x	x	x	x	x	x	x				
TRAP (Trap)		s = Immed4														
TRAPV (Trap on overflow)																
TST (Test)	B,W,L		d =	x		x	x	x	x	x	x	x				
UNLK (Unlink)					x											

OP code $b_{15} \cdots b_0$	Operation performed	Condition flags				
		X	N	Z	V	C
1001 DDD1 SSEE EEEE 1001 DDD0 SSeeeeee	d ← [d] – [s]	x	x	x	x	x
1001 AAA0 11ee eeee 1001 AAA1 11ee eeee	An ← [An] – [s]					
0000 0100 SSEE EEEE	d ← [d] – s	x	x	x	x	x
0101 QQQ1 SSEE EEEE	d ← [d] – QQQ	x	x	x	x	x
1001 RRR1 SS00 0rrr 1001 RRR1 SS00 1rrr	d ← [d] – [s] – [X]	x	x	x	x	x
0100 1000 0100 0DDD	$[Dn]_{31-16} \leftarrow [Dn]_{15-0}$		x	x	0	0
0100 1010 11EE EEEE	Test d and set N and Z flags; set bit 7 of d to 1.		x	x	0	0
0100 1110 0100 VVVV	SP ← [SP] – 4; [SP] ← [PC]; SP ← [SP] – 2; [SP] ← [SR]; PC ← vector					
0100 1110 0111 0110	If V = 1, then SP ← [SP] – 4; [SP] ← [PC]; SP ← [SP] – 2; [SP] ← [SR]; PC ← TRAPV vector					
0100 1010 SSEE EEEE	Test d and set N and Z flags.		x	x	0	0
0100 1110 0101 1AAA	SP ← [An]; An ← [[SP]]; SP ← [SP] + 4					