Motorola 68000 Instruction Set.

Instruction	Description	Assembler Syntax	Data Size	Condition Codes X N Z V C
ABCD	Add BCD with extend	Dx,Dy -(Ax),-(Ay)	B--	* U * U *
ADD	ADD binary	Dn,<ea> <ea>,Dn	BWL	* * * * *
ADDA	ADD binary to An	<ea>,An	-WL	- - - - -
ADDI	ADD Immediate	#x,<ea>	BWL	* * * * *
ADDQ	ADD 3-bit immediate	#<1-8>,<ea>	BWL	* * * * *
ADDX	ADD eXtended	Dy,Dx -(Ay),-(Ax)	BWL	* * * * *
AND	Bit-wise AND	<ea>,Dn Dn,<ea>	BWL	- * * 0 0
ANDI	Bit-wise AND with Immediate	#<data>,<ea>	BWL	- * * 0 0
ASL	Arithmetic Shift Left	#<1-8>,Dy Dx,Dy <ea>	BWL	* * * * *
ASR	Arithmetic Shift Right	...	BWL	* * * * *
Bcc	Conditional Branch	Bcc.S <label> Bcc.W <label>	BW-	- - - - -
BCHG	Test a Bit and CHanGe	Dn,<ea> #<data>,<ea>	B-L	- - * - -
BCLR	Test a Bit and CLeaR	...	B-L	- - * - -
BSET	Test a Bit and SET	...	B-L	- - * - -
BSR	Branch to SubRoutine	BSR.S <label> BSR.W <label>	BW-	- - - - -
BTST	Bit TeST	Dn,<ea> #<data>,<ea>	B-L	- - * - -
CHK	CHecK Dn Against Bounds	<ea>,Dn	-W-	- * U U U
CLR	CLeaR	<ea>	BWL	- 0 1 0 0
CMP	CoMPare	<ea>,Dn	BWL	- * * * *
CMPA	CoMPare Address	<ea>,An	-WL	- * * * *
CMPI	CoMPare Immediate	#<data>,<ea>	BWL	- * * * *
CMPM	CoMPare Memory	(Ay)+,(Ax)+	BWL	- * * * *
DBcc	Looping Instruction	DBcc Dn,<label>	-W-	- - - - -
DIVS	DIVide Signed	<ea>,Dn	-W-	- * * * 0
DIVU	DIVide Unsigned	<ea>,Dn	-W-	- * * * 0
EOR	Exclusive OR	Dn,<ea>	BWL	- * * 0 0
EORI	Exclusive OR Immediate	#<data>,<ea>	BWL	- * * 0 0
EXG	Exchange any two registers	Rx,Ry	--L	- - - - -
EXT	Sign EXTend	Dn	-WL	- * * 0 0
ILLEGAL	ILLEGAL-Instruction Exception	ILLEGAL		- - - - -
JMP	JuMP to Affective Address	<ea>		- - - - -
JSR	Jump to SubRoutine	<ea>		- - - - -
LEA	Load Effective Address	<ea>,An	--L	- - - - -
LINK	Allocate Stack Frame	An,#<displacement>		- - - - -
LSL	Logical Shift Left	Dx,Dy #<1-8>,Dy <ea>	BWL	* * * 0 *
LSR	Logical Shift Right	...	BWL	* * * 0 *
MOVE	Between Effective Addresses	<ea>,<ea>	BWL	- * * 0 0
MOVE	To CCR	<ea>,CCR	-W-	I I I I I
MOVE	To SR	<ea>,SR	-W-	I I I I I
MOVE	From SR	SR,<ea>	-W-	- - - - -

MOVE	USP to/from Address Register	USP,An An,USP	--L	- - - - -
MOVEA	MOVE Address	<ea>,An	-WL	- - - - -
MOVEM	MOVE Multiple	<register list>,<ea> <ea>,<register list>	-WL	- - - - -
MOVEP	MOVE Peripheral	Dn,x(An) x(An),Dn	-WL	- - - - -
MOVEQ	MOVE 8-bit immediate	#<-128.+127>,Dn	--L	- * * 0 0
MULS	MULtiply Signed	<ea>,Dn	-W-	- * * 0 0
MULU	MULtiply Unsigned	<ea>,Dn	-W-	- * * 0 0
NBCD	Negate BCD	<ea>	B--	* U * U *
NEG	NEGate	<ea>	BWL	* * * * *
NEGX	NEGate with eXtend	<ea>	BWL	* * * * *
NOP	No OPeration	NOP		- - - - -
NOT	Form one's complement	<ea>	BWL	- * * 0 0
OR	Bit-wise OR	<ea>,Dn Dn,<ea>	BWL	- * * 0 0
ORI	Bit-wise OR with Immediate	#<data>,<ea>	BWL	- * * 0 0
PEA	Push Effective Address	<ea>	--L	- - - - -
RESET	RESET all external devices	RESET		- - - - -
ROL	ROtate Left	#<1-8>,Dy Dx,Dy <ea>	BWL	- * * 0 *
ROR	ROtate Right	...	BWL	- * * 0 *
ROXL	ROtate Left with eXtend	...	BWL	* * * 0 *
ROXR	ROtate Right with eXtend	...	BWL	* * * 0 *
RTE	ReTurn from Exception	RTE		I I I I I
RTR	ReTurn and Restore	RTR		I I I I I
RTS	ReTurn from Subroutine	RTS		- - - - -
SBCD	Subtract BCD with eXtend	Dx,Dy -(Ax),-(Ay)	B--	* U * U *
Scc	Set to -1 if True, 0 if False	<ea>	B--	- - - - -
STOP	Enable & wait for interrupts	#<data>		I I I I I
SUB	SUBtract binary	Dn,<ea> <ea>,Dn	BWL	* * * * *
SUBA	SUBtract binary from An	<ea>,An	-WL	- - - - -
SUBI	SUBtract Immediate	#x,<ea>	BWL	* * * * *
SUBQ	SUBtract 3-bit immediate	#<data>,<ea>	BWL	* * * * *
SUBX	SUBtract eXtended	Dy,Dx -(Ay),-(Ax)	BWL	* * * * *
SWAP	SWAP words of Dn	Dn	-W-	- * * 0 0
TAS	Test & Set MSB & Set N/Z-bits	<ea>	B--	- * * 0 0
TRAP	Execute TRAP Exception	#<vector>		- - - - -
TRAPV	TRAPV Exception if V-bit Set	TRAPV		- - - - -
TST	TeST for negative or zero	<ea>	BWL	- * * 0 0
UNLK	Deallocate Stack Frame	An		- - - - -

Symbol	**Meaning**
*	Set according to result of operation
-	Not affected
0	Cleared
1	Set
U	Outcome (state after operation) undefined
I	Set by immediate data

<ea>	Effective Address Operand
<data>	Immediate data

<label> Assembler label
<vector> TRAP instruction Exception vector (0-15)
<rg.lst> MOVEM instruction register specification list
<displ.> LINK instruction negative displacement
...   Same as previous instruction

Addressing Modes	**Syntax**
Data Register Direct	Dn
Address Register Direct	An
Address Register Indirect	(An)
Address Register Indirect with Post-Increment	(An)+
Address Register Indirect with Pre-Decrement	-(An)
Address Register Indirect with Displacement	w(An)
Address Register Indirect with Index	b(An,Rx)
Absolute Short	w
Absolute Long	l
Program Counter with Displacement	w(PC)
Program Counter with Index	b(PC,Rx)
Immediate	#x
Status Register	SR
Condition Code Register	CCR

Legend

Dn	Data Register	(n is 0-7)
An	Address Register	(n is 0-7)
b	08-bit constant	
w	16-bit constant	
l	32-bit constant	
x	8-, 16-, 32-bit constant	
Rx	Index Register Specification, one of:	
	Dn.W Low 16 bits of Data Register	
	Dn.L All 32 bits of Data Register	
	An.W Low 16 bits of Address Register	
	An.L All 32 bits of Address Register	

Condition Codes for Bcc, DBcc and Scc Instructions.

Condition Codes set after CMP D0,D1 Instruction.

Relationship	**Unsigned**	**Signed**
D1 < D0	CS - Carry Bit Set	LT - Less Than
D1 <= D0	LS - Lower or Same	LE - Less than or Equal
D1 = D0	EQ - Equal (Z-bit Set)	EQ - Equal (Z-bit Set)
D1 != D0	NE - Not Equal (Z-bit Clear)	NE - Not Equal (Z-bit Clear)
D1 > D0	HI - HIgher than	GT - Greater Than
D1 >= D0	CC - Carry Bit Clear	GE - Greater than or Equal
	PL - PLus (N-bit Clear)	MI - Minus (N-bit Set)
	VC - V-bit Clear (No Overflow)	VS - V-bit Set (Overflow)
	RA - BRanch Always	

DBcc Only - F - Never Terminate (DBRA is an alternate to DBF)
T - Always Terminate

Scc Only - SF - Never Set
ST - Always Set

Berkeley risc 1

Opcode	Operands	Register Transfer	Description
Data manipulation instructions			
ADD	Rs,S2,Rd	$Rd \leftarrow Rs + S2$	Integer add
ADDC	Rs,S2,Rd	$Rd \leftarrow Rs + S2 +$ carry	Add with carry
SUB	Rs,S2,Rd	$Rd \leftarrow Rs - S2$	Integer subtract
SUBC	Rs,S2,Rd	$Rd \leftarrow Rs - S2 -$ carry	Subtract with carry
SUBR	Rs,S2,Rd	$Rd \leftarrow S2 - Rs$	Subtract reverse
SUBCR	Rs,S2,Rd	$Rd \leftarrow S2 - Rs -$ carry	Subtract with carry
AND	Rs,S2,Rd	$Rd \leftarrow Rs \wedge S2$	AND
OR	Rs,S2,Rd	$Rd \leftarrow Rs \vee S2$	OR
XOR	Rs,S2,Rd	$Rd \leftarrow Rs \oplus S2$	Exclusive-OR
SLL	Rs,S2,Rd	$Rd \leftarrow Rs$ shifted by $S2$	Shift-left
SRL	Rs,S2,Rd	$Rd \leftarrow Rs$ shifted by $S2$	Shift-right logical
SRA	Rs,S2,Rd	$Rd \leftarrow Rs$ shifted by $S2$	Shift-right arithmetic
Data transfer instructions			
LDL	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Load long
LDSU	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Load short unsigned
LDSS	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Load short signed
LDBU	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Load byte unsigned
LDBS	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Load byte signed
LDHI	Rd,Y	$Rd \leftarrow Y$	Load immediate high
STL	Rd,(Rs)S2	$M[Rs + S2] \leftarrow Rd$	Store long
STS	Rd,(Rs)S2	$M[Rs + S2] \leftarrow Rd$	Store short
STB	Rd,(Rs)S2	$M[Rs + S2] \leftarrow Rd$	Store byte
GETPSW	Rd	$Rd \leftarrow PSW$	Load status word
PUTPSW	Rd	$PSW \leftarrow Rd$	Set status word
Program control instructions			
JMP	COND, S2(Rs)	$PC \leftarrow Rs + S2$	Conditional jump
JMPR	COND,Y	$PC \leftarrow PC + Y$	Jump relative
CALL	Rd,S2(Rs)	$Rd \leftarrow PC$ $PC \leftarrow Rs + S2$ $CWP \leftarrow CWP - 1$	Call subroutine and change window
CALLR	Rd,Y	$Rd \leftarrow PC$ $PC \leftarrow PC + Y$ $CWP \leftarrow$ CWP $- 1$	Call relative and change window
RET	Rd,S2	$PC \leftarrow Rd + S2$ $CWP - CWP + 1$	Return and change window
CALLINT	Rd	$Rd \leftarrow PC$ $CWP \leftarrow CWP - 1$	Disable interrupts
RETINT	Rd,S2	$PC \leftarrow Rd + S2$ $CWP \leftarrow CWP + 1$	Enable interrupts
GTLPC	Rd	$Rd \leftarrow PC$	Get last PC