A set of extended instructions provides multiprecision and mixed-size arithmetic: ADDX, SUBX, EXT, and NEGX. Refer to Table 3-3 for a summary of the integer arithmetic operations. In Table 3-3, X refers to the X-bit in the CCR.

Table 3-3. Integer Arithmetic Operation Format

Instruction	Operand Syntax	Operand Size	Operation
ADD ADDA	Dn,<ea> <ea>,Dn <ea>,An	8, 16, 32 8, 16, 32 16, 32	Source + Destination → Destination
ADDI ADDQ	#<data>,<ea> #<data>,<ea>	8, 16, 32 8, 16, 32	Immediate Data + Destination → Destination
ADDX	Dn,Dn –(An), –(An)	8, 16, 32 8, 16, 32	Source + Destination + X → Destination
CLR	<ea>	8, 16, 32	0 → Destination
CMP CMPA	<ea>,Dn <ea>,An	8, 16, 32 16, 32	Destination – Source
CMPI	#<data>,<ea>	8, 16, 32	Destination – Immediate Data
CMPM	(An)+,(An)+	8, 16, 32	Destination – Source
CMP2	<ea>,Rn	8, 16, 32	Lower Bound → Rn → Upper Bound
DIVS/DIVU DIVSL/DIVUL	<ea>,Dn <ea>,Dr–Dq <ea>,Dq <ea>,Dr–Dq	32 ÷16 → 16,16 64 ÷32 → 32,32 32 ÷32 → 32 32 ÷32 → 32,32	Destination ÷Source → Destination (Signed or Unsigned Quotient, Remainder)
EXT EXTB	Dn Dn Dn	8 → 16 16 → 32 8 → 32	Sign-Extended Destination → Destination
MULS/MULU	<ea>,Dn <ea>,Dl <ea>,Dh–Dl	16 x 16 → 32 32 x 32 → 32 32 x 32 → 64	Source x Destination → Destination (Signed or Unsigned)
NEG	<ea>	8, 16, 32	0 – Destination → Destination
NEGX	<ea>	8, 16, 32	0 – Destination – X → Destination
SUB SUBA	<ea>,Dn Dn,<ea> <ea>,An	8, 16, 32 8, 16, 32 16, 32	Destination = Source → Destination
SUBI SUBQ	#<data>,<ea> #<data>,<ea>	8, 16, 32 8, 16, 32	Destination – Immediate Data → Destination
SUBX	Dn,Dn –(An), –(An)	8, 16, 32 8, 16, 32	Destination – Source – X → Destination

3.1.3 Logical Instructions

The logical operation instructions (AND, OR, EOR, and NOT) perform logical operations with all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provides these logical operations with all sizes of immediate data. Table 3-4 summarizes the logical operations.

Table 3-4. Logical Operation Format

Instruction	Operand Syntax	Operand Size	Operation
AND	<ea>,Dn Dn,<ea>	8, 16, 32 8, 16, 32	Source Λ Destination $\rightarrow$ Destination
ANDI	#<data>,<ea>	8, 16, 32	Immediate Data Λ Destination $\rightarrow$ Destination
EOR	Dn,<ea>	8, 16, 32	Source $\oplus$ Destination $\rightarrow$ Destination
EORI	#<data>,<ea>	8, 16, 32	Immediate Data $\oplus$ Destination $\rightarrow$ Destination
NOT	<ea>	8, 16, 32	~ Destination $\rightarrow$ Destination
OR	<ea>,Dn Dn,<ea>	8, 16, 32	Source V Destination $\rightarrow$ Destination
ORI	#<data>,<ea>	8, 16, 32	Immediate Data V Destination $\rightarrow$ Destination

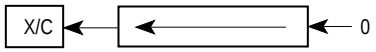
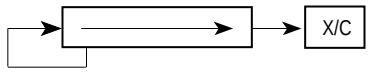
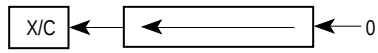
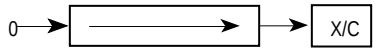
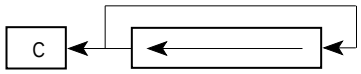
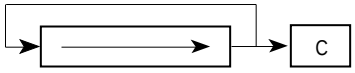
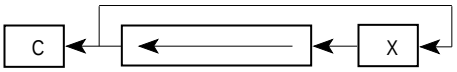
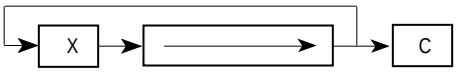
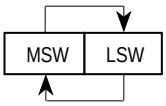
3.1.4 Shift and Rotate Instructions

The ASR, ASL, LSR, and LSL instructions provide shift operations in both directions. The ROR, ROL, ROXR, and ROXL instructions perform rotate (circular shift) operations, with and without the CCR extend bit (X-bit). All shift and rotate operations can be performed on either registers or memory.

Register shift and rotate operations shift all operand sizes. The shift count can be specified in the instruction operation word (to shift from 1 – 8 places) or in a register (modulo 64 shift count).

Memory shift and rotate operations shift word operands one bit position only. The SWAP instruction exchanges the 16-bit halves of a register. Fast byte swapping is possible by using the ROR and ROL instructions with a shift count of eight, enhancing the performance of the shift/rotate instructions. Table 3-5 is a summary of the shift and rotate operations. In Table 3-5, C and X refer to the C-bit and X- bit in the CCR.

Table 3-5. Shift and Rotate Operation Format

Instruction	Operand Syntax	Operand Size	Operation
ASL	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
LSL	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
ROL	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn # ⟨data⟩, Dn ea	8, 16, 32 8, 16, 32 16	
SWAP	Dn	32	

NOTE: X indicates the extend bit and C the carry bit in the CCR.

3.1.5 Bit Manipulation Instructions

BTST, BSET, BCLR, and BCHG are bit manipulation instructions. All bit manipulation operations can be performed on either registers or memory. The bit number is specified either as immediate data or in the contents of a data register. Register operands are 32 bits long, and memory operands are 8 bits long. Table 3-6 summarizes bit manipulation operations; Z refers to the zero bit of the CCR.

Table 3-6. Bit Manipulation Operation Format

Instruction	Operand Syntax	Operand Size	Operation
BCHG	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) → Z → Bit of Destination
BCLR	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) → Z; 0 → Bit of Destination
BSET	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) → Z; 1 → Bit of Destination
BTST	Dn,<ea> #<data>,<ea>	8, 32 8, 32	~ (<Bit Number> of Destination) → Z

3.1.6 Bit Field Instructions

The M68000 family architecture supports variable-length bit field operations on fields of up to 32 bits. The BFINS instruction inserts a value into a bit field. BFEXTU and BFEXTS extract a value from the field. BFFFO finds the first set bit in a bit field. Also included are instructions analogous to the bit manipulation operations: BFTST, BFSET, BFCLR, and BFCHG. Table 3-7 summarizes bit field operations.

Table 3-7. Bit Field Operation Format

Instruction	Operand Syntax	Operand Size	Operation
BFCHG	<ea> {offset:width}	1–32	~ Field → Field
BFCLR	<ea> {offset:width}	1–32	0's → Field
BFEXTS	<ea> {offset:width}, Dn	1–32	Field → Dn; Sign-Extended
BFEXTU	<ea> {offset:width}, Dn	1–32	Field → Dn; Zero-Extended
BFFFO	<ea> {offset:width}, Dn	1–32	Scan for First Bit Set in Field; Offset → Dn.
BFINS	Dn,<ea> {offset:width}	1–32	Dn → Field
BFSET	<ea> {offset:width}	1–32	1's → Field
BFTST	<ea> {offset:width}	1–32	Field MSB → N; ~ (OR of All Bits in Field) → Z

NOTE: All bit field instructions set the CCR N and Z bits as shown for BFTST before performing the specified operation.

3.1.7 Binary-Coded Decimal Instructions

Five instructions support operations on binary-coded decimal (BCD) numbers. The arithmetic operations on packed BCD numbers are ABCD, SBCD, and NBCD. PACK and UNPK instructions aid in the conversion of byte-encoded numeric data, such as ASCII or EBCDIC strings to BCD data and vice versa. Table 3-8 summarizes BCD operations. In Table 3- 8 X refers to the X-bit in the CCR.

Table 3-8. Binary-Coded Decimal Operation Format

Instruction	Operand Syntax	Operand Size	Operation
ABCD	Dn,Dn –(An), –(An)	8 8	$Source_{10}$ + $Destination_{10}$ + X → Destination
NBCD	<ea>	8	0 – $Destination_{10}$ – X → Destination
PACK	–(An), –(An) #<data> Dn,Dn,#<data>	16 → 8 16 → 8	Unpackaged Source + Immediate Data → Packed Destination
SBCD	Dn,Dn –(An), –(An)	8 8	$Destination_{10}$ – $Source_{10}$ – X → Destination
UNPK	–(An),–(An) #<data> Dn,Dn,#<data>	8 → 16 8 → 16	Packed Source → Unpacked Source Unpacked Source + Immediate Data → Unpacked Destination

3.1.8 Program Control Instructions

A set of subroutine call and return instructions and conditional and unconditional branch instructions perform program control operations. Also included are test operand instructions (TST and FTST), which set the integer or floating-point condition codes for use by other program and system control instructions. NOP forces synchronization of the internal pipelines. Table 3-9 summarizes these instructions.

Table 3-9. Program Control Operation Format

Instruction	Operand Syntax	Operand Size	Operation
Integer and Floating-Point Conditional			
Bcc, FBcc	<label>	8, 16, 32	If Condition True, Then PC + d_n → PC
DBcc, FDBcc	Dn,<label>	16	If Condition False, Then Dn – 1 → Dn If Dn → –1, Then PC + d_n → PC
Scc, FScc	<ea>	8	If Condition True, Then 1's → Destination; Else 0's → Destination
Unconditional			
BRA	<label>	8, 16, 32	PC + d_n → PC
BSR	<label>	8, 16, 32	SP – 4 → SP; PC → (SP); PC + d_n → PC
JMP	<ea>	none	Destination → PC
JSR	<ea>	none	SP – 4 → SP; PC → (SP); Destination → PC
NOP	none	none	PC + 2 → PC (Integer Pipeline Synchronized)
FNOP	none	none	PC + 4 → PC (FPU Pipeline Synchronized)
Returns			
RTD	#<data>	16	(SP) → PC; SP + 4 + d_n → SP
RTR	none	none	(SP) → CCR; SP + 2 → SP; (SP) →PC; SP + 4 → SP
RTS	none	none	(SP) → PC; SP + 4 → SP
Test Operand			
TST	<ea>	8, 16, 32	Set Integer Condition Codes
FTST	<ea> FPn	B, W, L, S, D, X, P X	Set Floating-Point Condition Codes

Letters cc in the integer instruction mnemonics Bcc, DBcc, and Scc specify testing one of the following conditions:

CC—Carry clear	GE—Greater than or equal
LS—Lower or same	PL—Plus
CS—Carry set	GT—Greater than
LT—Less than	T—Always true*
EQ—Equal	HI—Higher
MI—Minus	VC—Overflow clear
F—Never true*	LE—Less than or equal
NE—Not equal	VS—Overflow set

*Not applicable to the Bcc instructions.

3.1.9 System Control Instructions

Privileged and trapping instructions as well as instructions that use or modify the CCR provide system control operations. FSAVE and FRESTORE save and restore the nonuser visible portion of the FPU during context switches in a virtual memory or multitasking system. The conditional trap instructions, which use the same conditional tests as their corresponding program control instructions, allow an optional 16- or 32-bit immediate operand to be included as part of the instruction for passing parameters to the operating system. These instructions cause the processor to flush the instruction pipe. Table 3-10 summarizes these instructions. See 3.2 Integer Unit Condition Code Computation for more details on condition codes.

Table 3-10. System Control Operation Format

Instruction	Operand Syntax	Operand Size	Operation
Privileged			
ANDI to SR	#<data>,SR	16	Immediate Data ∧ SR → SR
EORI to SR	#<data>,SR	16	Immediate Data ⊕ SR → SR
FRESTORE	<ea>	none	State Frame → Internal Floating-Point Registers
FSAVE	<ea>	none	Internal Floating-Point Registers → State Frame
MOVE to SR	<ea>,SR	16	Source → SR
MOVE from SR	SR,<ea>	16	SR → Destination
MOVE USP	USP,An An,USP	32 32	USP → An An → USP
MOVEC	Rc,Rn Rn,Rc	32 32	Rc → Rn Rn → Rc
MOVES	Rn,<ea> <ea>,Rn	8, 16, 32	Rn → Destination Using DFC Source Using SFC → Rn
ORI to SR	#<data>,SR	16	Immediate Data ∨ SR → SR
RESET	none	none	Assert Reset Output
RTE	none	none	(SP) → SR; SP + 2 → SP; (SP) →PC; SP + 4 → SP; Restore Stack According to Format
STOP	#<data>	16	Immediate Data → SR; STOP
Trap Generating			
BKPT	#<data>	none	Run Breakpoint Cycle
CHK	<ea>,Dn	16, 32	If Dn < 0 or Dn > (<ea>), Then CHK Exception
CHK2	<ea>,Rn	8, 16, 32	If Rn< Lower Bound or Rn > Upper Bound, Then CHK Exception
ILLEGAL	none	none	SSP – 2 → SSP; Vector Offset → (SSP); SSP – 4 → SSP; PC → (SSP); SSP – 2 → SSP; SR → (SSP); Illegal Instruction Vector Address → PC
TRAP	#<data>	none	SSP – 2 → SSP; Format and Vector Offset → (SSP) SSP – 4 → SSP; PC → (SSP); SSP – 2 → SSP; SR → (SSP); Vector Address → PC
TRAPcc	none #<data>	none 16, 32	If cc True, Then Trap Exception
FTRAPcc	none #<data>	none 16, 32	If Floating-Point cc True, Then Trap Exception
TRAPV	none	none	If V, Then Take Overflow Trap Exception
Condition Code Register			
ANDI to SR	#<data>,CCR	8	Immediate Data ∧ CCR → CCR
EORI to SR	#<data>,CCR	8	Immediate Data ⊕ CCR → CCR
MOVE to SR	<ea>,CCR	16	Source → CCR
MOVE from SR	CCR,<ea>	16	CCR → Destination
ORI to SR	#<data>,CCR	8	Immediate Data ∨ CCR → CCR

Letters cc in the TRAPcc and FTRAPcc specify testing for a condition.

3.1.10 Cache Control Instructions (MC68040)

The cache instructions provide maintenance functions for managing the instruction and data caches. CINV invalidates cache entries in both caches, and CPUSH pushes dirty data from the data cache to update memory. Both instructions can operate on either or both caches and can select a single cache line, all lines in a page, or the entire cache. Table 3-11 summarizes these instructions.

Table 3-11. Cache Control Operation Format

Instruction	Operand Syntax	Operand Size	Operation
CINVL	caches,(An)	none	Invalidate cache line
CINVP	caches, (An)	none	Invalidate cache page
CINVA	caches	none	Invalidate entire cache
CPUSHL CPUSHP CPUSHA	caches,(An) caches, (An) caches	none none none	Push selected dirty data cache lines, then invalidate selected cache lines

3.1.11 Multiprocessor Instructions

The TAS, CAS, and CAS2 instructions coordinate the operations of processors in multiprocessing systems. These instructions use read- modify-write bus cycles to ensure uninterrupted updating of memory. Coprocessor instructions control the coprocessor operations. Table 3- 12 summarizes these instructions.

Table 3-12. Multiprocessor Operations

Instruction	Operand Syntax	Operand Size	Operation
Read-Write-Modify			
CAS	Dc,Du,<ea>	8, 16, 32	Destination – Dc → CC If Z, Then Du → Destination Else Destination → Dc
CAS2	Dc1–Dc2, Du1–Du2, (Rn)–(Rn)	16, 32	Dual Operand CAS
TAS	<ea>	8	Destination – 0; Set Condition Codes; 1 → Destination [7]
Coprocessor			
cpBcc	<label>	16, 32	If cpcc True, Then PC + d_n → PC
cpDBcc	<label>,Dn	16	If cpcc False, Then Dn – 1 → Dn If Dn ≠ –1, Then PC + d_n → PC
cpGEN	User Defined	User Defined	Operand → Coprocessor
cpRESTORE	<ea>	none	Restore Coprocessor State from <ea>
cpSAVE	<ea>	none	Save Coprocessor State at <ea>
cpScc	<ea>	8	If cpcc True, Then 1's → Destination; Else 0's → Destination
cpTRAPcc	none #<data>	none 16, 32	If cpcc True, Then TRAPcc Exception

3.1.12 Memory Management Unit (MMU) Instructions

The PFLUSH instructions flush the address translation caches (ATCs) and can optionally select only nonglobal entries for flushing. PTEST performs a search of the address translation tables, stores the results in the MMU status register, and loads the entry into the ATC. Table 3-13 summarizes these instructions.

Table 3-13. MMU Operation Format

Instruction	Processor	Operand Syntax	Operand Size	Operation
PBcc	MC68851	<label>	none	Branch on PMMU Condition
PDBcc	MC68851	Dn,<label>	none	Test, Decrement, and Branch
PFLUSHA	MC68030 MC68040 MC68851	none	none	Invalidate All ATC Entries
PFLUSH	MC68040	(An)	none	Invalidate ATC Entries at Effective Address
PFLUSHN	MC68040	(An)	none	Invalidate Nonglobal ATC Entries at Effective Address
PFLUSHAN	MC68040	none	none	Invalidate All Nonglobal ATC Entries
PFLUSHS	MC68851	none	none	Invalidate All Shared/Global ATC Entries
PFLUSHR	MC68851	<ea>	none	Invalidate ATC and RPT Entries
PLOAD	MC68030 MC68851	FC,<ea>	none	Load an Entry into the ATC
PMOVE	MC68030 MC68851	MRn,<ea> <ea>,MRn	8,16,32,64	Move to/from MMU Registers
PRESTORE	MC68851	<ea>	none	PMMU Restore Function
PSAVE	MC68851	<ea>	none	PMMU Save Function
PScc	MC68851	<ea>	8	Set on PMMU Condition
PTEST	MC68030 MC68040 MC68851	(An)	none	Information About Logical Address → MMU Status Register
PTRAPcc	MC68851	#<data>	16,32	Trap on PMMU Condition

3.1.13 Floating-Point Arithmetic Instructions

The following paragraphs describe the floating-point instructions, organized into two categories of operation: dyadic (requiring two operands) and monadic (requiring one operand).

The dyadic floating-point instructions provide several arithmetic functions that require two input operands, such as add and subtract. For these operations, the first operand can be located in memory, an integer data register, or a floating-point data register. The second operand is always located in a floating-point data register. The results of the operation store in the register specified as the second operand. All FPU operations support all data formats. Results are rounded to either extended-, single-, or double-precision format. Table 3-14 gives the general format of dyadic instructions, and Table 3-15 lists the available operations.

Table 3-14. Dyadic Floating-Point Operation Format

Instruction	Operand Syntax	Operand Format	Operation
F<dop>	<ea>,FPn FPm,FPn	B, W, L, S, D, X, P X	FPn <Function> Source → FPn

NOTE: < dop > is any one of the dyadic operation specifiers.

Table 3-15. Dyadic Floating-Point Operations

Instruction	Operation
FADD, FSADD, FDADD	Add
FCMP	Compare
FDIV, FSDIV, FDDIV	Divide
FMOD	Modulo Remainder
FMUL, FSMUL, FDMUL	Multiply
FREM	IEEE Remainder
FSCALE	Scale Exponent
FSUB, FSSUB, FDSUB	Subtract
FSGLDIV, FSGLMUL	Single-Precision Divide, Multiply

The monadic floating-point instructions provide several arithmetic functions requiring only one input operand. Unlike the integer counterparts to these functions (e.g., NEG < ea >), a source and a destination can be specified. The operation is performed on the source operand and the result is stored in the destination, which is always a floating-point data register. When the source is not a floating-point data register, all data formats are supported. The data format is always extended precision for register-to-register operations. Table 3-16 lists the general format of these instructions, and Table 3-17 lists the available operations.

Table 3-16. Monadic Floating-Point Operation Format

Instruction	Operand Syntax	Operand Format	Operation
F<mop>	<ea>,FPn FPm,FPn FPn	B, W, L, S, D, X, P X X	Source → Function → FPn FPn → Function → FPn

NOTE: < mop > is any one of the monadic operation specifiers.

Table 3-17. Monadic Floating-Point Operations

Instruction	Operation
FABS	Absolute Value
FACOS	Arc Cosine
FASIN	Arc Sine
FATAN	Hyperbolic Art Tangent
FCOS	Cosine
FCOSH	Hyperbolic Cosine
FETOX	e^x
FETOXM1	$e^x - 1$
FGETEXP	Extract Exponent
FGETMAN	Extract Mantissa
FINT	Extract Integer Part
FINTRZ	Extract Integer Part, Rounded-to-Zero

Instruction	Operation
FLOGN	In(x)
FLOGNP1	In(x + 1)
FLOG10	$Log_{10}(x)$
FLOG2	$Log_2(x)$
FNEG	Negate
FSIN	Sine
FSINH	Hyperbolic Sine
FSQRT	Square Root
FTAN	Tangent
FTANH	Hyperbolic Tangent
FTENTOX	10^x
FTWOTOX	2^x

3.2 INTEGER UNIT CONDITION CODE COMPUTATION

Many integer instructions affect the CCR to indicate the instruction,s results. Program and system control instructions also use certain combinations of these bits to control program and system flow. The condition codes meet consistency criteria across instructions, uses, and instances. They also meet the criteria of meaningful results, where no change occurs unless it provides useful information. Refer to **Section 1 Introduction** for details concerning the CCR.

Table 3-18 lists the integer condition code computations for instructions and Table 3-19 lists the condition names, encodings, and tests for the conditional branch and set instructions. The test associated with each condition is a logical formula using the current states of the condition codes. If this formula evaluates to one, the condition is true. If the formula evaluates to zero, the condition is false. For example, the T condition is always true, and the EQ condition is true only if the Z-bit condition code is currently true.