MOTOROLA

MC68000

Powers of 16, Powers of 2

16^m m=	2^n n=	Value
0	0	1
	1	2
	2	4
	3	8
1	4	16
	5	32
	6	64
	7	128
2	8	256
	9	512
	10	1,024
	11	2,048
3	12	4,096
	13	8,192
	14	16,384
	15	32,768

16^m m=	2^n n=	Value
4	16	65,536
	17	131,072
	18	262,144
	19	524,288
5	20	1,048,576
	21	2,097,152
	22	4,194,304
	23	8,388,608
6	24	16,777,216
	25	33,554,432
	26	67,108,864
	27	134,217,728
7	28	268,435,456
	29	536,870,912
	30	1,073,741,824
	31	2,147,483,648
8	32	4,294,967,296

ASCII Character Set (7-Bit Code)

LS Dig. \ MS Dig.	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Hexadecimal and Decimal Conversion

How to use:

Conversion to Decimal: Find the decimal weights for corresponding hexadecimal characters beginning with the least significant character. The sum of the decimal weights is the decimal value of the hexadecimal number.

Conversion to Hexadecimal: Find the highest decimal value in the table which is lower than or equal to the decimal number to be converted. The corresponding hexadecimal character is the most significant. Subtract the decimal value found from the decimal number to be converted. With the difference repeat the process to find subsequent hexadecimal characters.

23 Byte 16				15 Byte 8				7 Byte 0			
23 Char 20		19 Char 16		15 Char 12		11 Char 8		7 Char 4		3 Char 0	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,158	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3840	F	240	F	15

1ATX13305-18 PRINTED IN USA 10/93 IMPERIAL LITHO 94700 10,000 MPU YGABAA

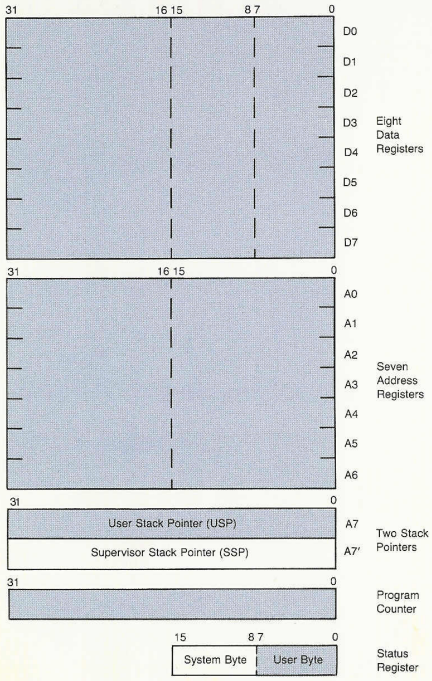
MOTOROLA

MC68000/AC
REV 3

MC68000

16-/32-BIT MICROPROCESSOR PROGRAMMING REFERENCE CARD

Programming Model



Conditional Tests

Mnemonic	Condition	Encoding	Test
T	true	0000	1
F	false	0001	0
HI	high	0010	$\overline{C} \cdot \overline{Z}$
LS	low or same	0011	$C + Z$
CC(HS)	carry clear	0100	$\overline{C}$
CS(LO)	carry set	0101	C
NE	not equal	0110	$\overline{Z}$
EQ	equal	0111	Z
VC	overflow clear	1000	$\overline{V}$
VS	overflow set	1001	V
PL	plus	1010	$\overline{N}$
MI	minus	1011	N
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$
LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$

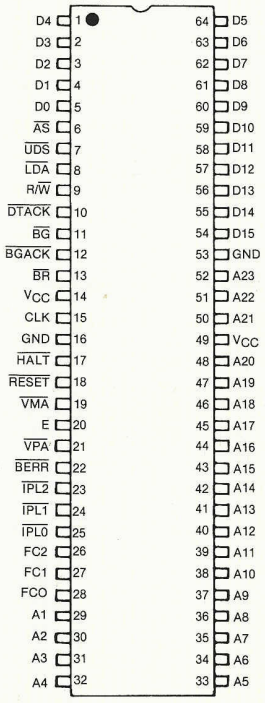
Operation Code Map

Bits 15 through 12	Operation
0000	Bit Manipulation/MOVEP/ Immediate
0001	Move Byte
0010	Move Long
0011	Move Word
0100	Miscellaneous
0101	ADDQ/SUBQ/Scc/DBcc
0110	Bcc/BSR
0111	MOVEQ

Bits 15 through 12	Operation
1000	OR/DIV/SBCD
1001	SUB/SUBX
1010	(Unassigned)
1011	CMP/EOR
1100	AND/MUL/ABCD/EXG
1101	ADD/ADDX
1110	Shift/Rotate
1111	(Unassigned)

Pin Assignments

64-Pin Dual-in-Line Package

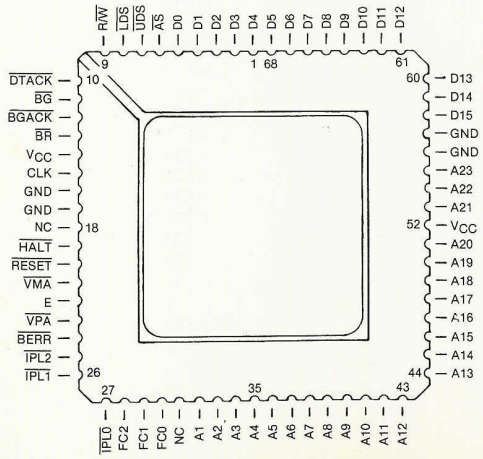


Pin Assignments

68-Pin Grid Array

	1	2	3	4	5	6	7	8	9	10
K	NC	FC2	FC0	A1	A3	A4	A6	A7	A9	NC
J	BERR	IPL0	FC1	NC	A2	A5	A8	A10	A11	A14
H	E	IPL2	IPL1					A13	A12	A16
G	VMA	VPA							A15	A17
F	HALT	RESET				BOTTOM			A18	A19
E	CLK	GND				VIEW			VCC	A20
D	BR	VCC							GND	A21
C	BGACK	BG	R/W					D13	A23	A22
B	DTACK	LDS	UDS	D0	D3	D6	D9	D11	D14	D15
A	NC	AS	D1	D2	D4	D5	D7	D8	D10	D12

68-Terminal Chip Carrier

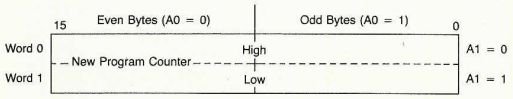


Reference Classification

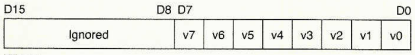
Function Code Output			
FC2	FC1	FC0	Reference Class
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)

Function Code Output			
FC2	FC1	FC0	Reference Class
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

Exception Vector Format

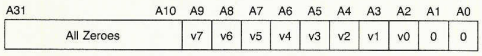


Peripheral Vector Number Format

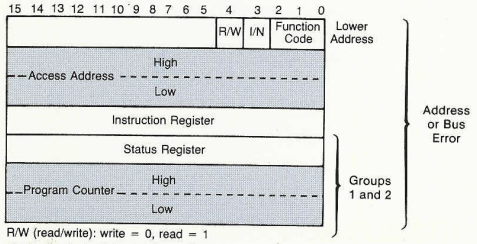


Where
v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Address Translated from 8-Bit Vector Number



Supervisor Stack Format



R/W (read/write): write = 0, read = 1
I/N (instruction/not): instruction = 0, not = 1

Exception Grouping and Priority

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception processing begins within two clock cycles. (Highest priority)
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction.
2	TRAP, TRAPV, CHK Zero Divide	Exception processing is started by normal instruction execution. (Lowest priority)

Exception Vector Assignment

Vector Number(s)	Address			Assignment
	Dec	Hex	Space[6]	
0	0	000	SP	Reset: Initial SSP[2]
1	4	004	SP	Reset: Initial PC[2]
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12[1]	48	030	SD	(Unassigned, Reserved)
13[1]	52	034	SD	(Unassigned, Reserved)
14	56	038	SD	Format Error[5]
15	60	03C	SD	Uninitialized Interrupt Vector
16-23[1]	64	040	SD	(Unassigned, Reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt[3]
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors[4]
	191	0BF		
48-63[1]	192	0C0	SD	(Unassigned, Reserved)
	255	0FF		—
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

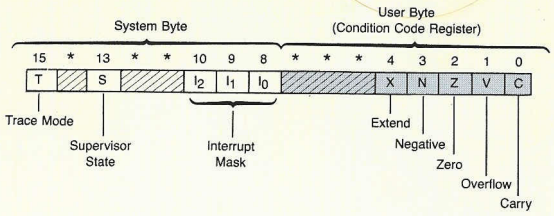
NOTES:
1. Vector numbers 12, 13, 16 through 23, and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.
2. Reset vector (0) requires four words, unlike the other vectors which only require two words, and is located in the supervisor program space.
3. The spurious interrupt vector is taken when there is a bus error indication during interrupt processing.
4. Trap #n uses vector number 32+n.
5. MC68010/MC68012 only.
   This vector is unassigned, reserved on the MC68000, and MC68008.
6. SP denotes supervisor program space, and SD denotes supervisor data space.

Exception Processing Execution Times

Exception	Periods
Address Error	50(4/7)
Bus Error	50(4/7)
CHK Instruction	40(4/3)+
Divide by Zero	38(4/3)+
Illegal Instruction	34(4/3)
Interrupt	44(5/3)*
Privilege Violation	34(4/3)
RESET**	40(6/0)
Trace	34(4/3)
TRAP Instruction	34(4/3)
TRAPV Instruction	34(5/3)

+Add effective address calculation time.
*The interrupt acknowledge cycle is assumed to take four clock periods.
**Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

Status Register

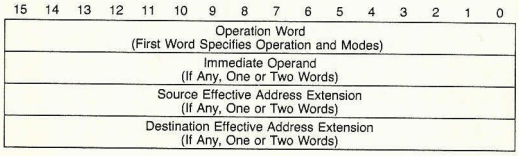


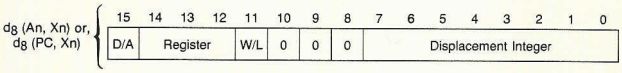
*Denotes reserved bits, read only as "0".

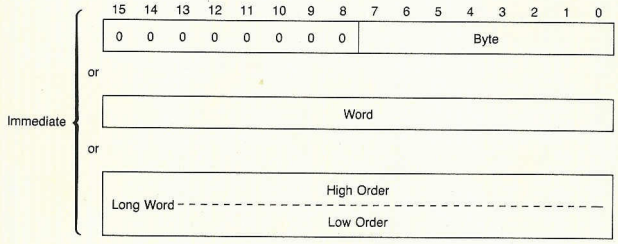
Interrupt Encoding

Priority	$\overline{IPL2}/\overline{I1}/\overline{I0}$ Control Lines	Requested Interrupt Level	Status Reg. Int. Mask I2/I1/I0	Recognized Interrupt Level
Highest	LLL	7	111	7
•	LLH	6	110	7
•	LHL	5	101	6,7
•	LHH	4	100	5-7
•	HLL	3	011	4-7
•	HLH	2	010	3-7
•	HHL	1	001	2-7
Lowest	HHH	None	000	1-7

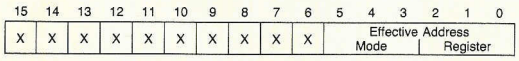
Instruction Operation Word General Format



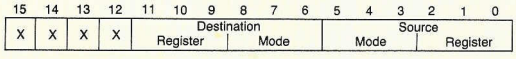




Single-Effective-Address Instruction Operation Word



Double-Effective-Address Instruction Operation Word



Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
ADD, ADDI, ADDQ	*	*	*	?	?	V = Sm • Dm • $\overline{Rm}$ + $\overline{Sm}$ • $\overline{Dm}$ • Rm C = Sm • Dm + $\overline{Rm}$ • Dm + Sm • $\overline{Rm}$
ADDX	*	*	?	?	?	V = Sm • Dm • $\overline{Rm}$ + $\overline{Sm}$ • $\overline{Dm}$ • Rm C = Sm • Dm + Rm • Dm + Sm • $\overline{Rm}$ Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	—	*	*	0	0	
CHK	—	*	U	U	U	
SUB, SUBI, SUBQ	*	*	*	?	?	V = $\overline{Sm}$ • Dm • $\overline{Rm}$ + Sm • $\overline{Dm}$ • Rm C = Sm • $\overline{Dm}$ + Rm • $\overline{Dm}$ + Sm • Rm
SUBX	*	*	?	?	?	V = $\overline{Sm}$ • Dm • $\overline{Rm}$ + Sm • $\overline{Dm}$ • Rm C = Sm • $\overline{Dm}$ + Rm • $\overline{Dm}$ + Sm • Rm Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
CMP, CMPI, CMPM	—	*	*	?	?	V = $\overline{Sm}$ • Dm • $\overline{Rm}$ + Sm • $\overline{Dm}$ • Rm C = Sm • $\overline{Dm}$ + Rm • $\overline{Dm}$ + Sm • Rm
DIVS, DIVU	—	*	*	?	0	V = Division Overflow
MULS, MULU	—	*	*	?	0	V = Multiply Overflow
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$

Operations	X	N	Z	V	C	Special Definition
NEG	*	*	*	?	?	V = Dm • Rm, C = Dm + Rm
NEGX	*	*	?	?	?	V = Dm • Rm, C = Dm + Rm Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
BTST, BCHG, BSET, BCLR	—	—	?	—	—	Z = $\overline{Dn}$
ASL	*	*	*	?	?	V = Dm • ($\overline{D_{m-1}}$ + ... + $\overline{D_{m-r}}$) + $\overline{Dm}$ • (D_{m-1} + ... + D_{m-r}) C = D_{m-r+1}
ASL (r=0)	—	*	*	0	0	
LSL, ROXL	*	*	*	0	?	C = D_{m-r+1}
LSR (r=0)	—	*	*	0	0	
ROXL (r=0)	—	*	*	0	?	C=X
ROL	—	*	*	0	?	C=D_{m-r+1}
ROL (r=0)	—	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	C=D_{r-1}
ASR, LSR (r=0)	—	*	*	0	0	
ROXR (r=0)	—	*	*	0	?	C=X
ROR	—	*	*	0	?	C=D_{r-1}
ROR (r=0)	—	*	*	0	0	

NOTES:
- Sm Source Operated — most significant bit
- Dm Destination operand — most significant bit
- ? See Special Definition
- Rm Result operand — most significant bit
- n bit number
- r shift count
- • Boolean AND
- + Boolean OR
- $\overline{Rm}$ Boolean NOT

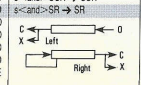
Effective Addressing Mode Categories

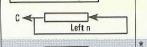
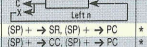
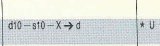
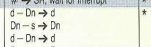
Type	Mode	Register	Generation	Assembler Syntex
Data Register Direct	000	reg. no.	EA=Dn	Dn
Address Register Direct	001	reg. no.	EA=An	An
Register Indirect	010	reg. no.	EA=(An)	(An)
Postincrement Register Indirect	011	reg. no.	EA=(An), An←An+N	(An)+
Predecrement Register Indirect	100	reg. no.	An←An−N, EA=(An)	−(An)
Register Indirect With Offset	101	reg. no.	EA=(An)+d_{16}	d_{16}(An)
Indexed Register Indirect With Offset	110	reg. no.	EA=(An)+(Xn)+d_8	d_8(An, Xn)
Absolute Short	111	000	EA=(Next Word)	xxx
Absolute Long	111	001	EA=(Next Two Words)	xxxxxx
PC Relative With Offset	111	010	EA=(PC)+d_{16}	d_{16}(PC)
PC Relative With Index and Offset	111	011	EA=(PC)+(Xn)+d_8	d_8(PC+Xn)
Immediate	111	100	Data=Next Word(s)	#xxx
Quick Immediate	—	—	Inherent Data	#xxx (1–8)
Implied Register	—	—	EA=SR, USP, SP, PC	—

NOTES:
- EA = Effective Address
- An = Address Register
- Dn = Data Register
- Xn = Address or Data Register used as Index Register
- SR = Status Register
- PC = Program Counter
- d_8 = Eight bit Offset (displacement)
- d_{16} = Sixteen bit Offset (displacement)
- N = 1 for Byte, 2 for Words and 4 for Long Words
- ( ) = Contents of
- ← = Replaces

Mnemonic	Size	Address Mode	Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d_{16}(An) #	d_{16}(An) ~	d_8(An,Xn) #	d_8(An,Xn) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~	d_{16}(PC) #	d_{16}(PC) ~	d_8(PC,Xn) #	d_8(PC,Xn) ~	s = Immed d = SR/CC #	s = Immed d = SR/CC ~	Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
DBcc	W	d_{16}=Imm				cc: Counter: Branch																				0101 CCCC 1100 1DDD	If cc true, then NOP else Dn−1→Dn, If Dn≠−1, then PC+disp→PC	- - - - -
		counter=	4			10 false yes 12 true no 14 false expired																						
DIVS	W	d=Dn s=	2	<158			2	<162	2	<162	2	<164	4	<166	4	<168	4	<166	6	<170	4	<166	4	<168	4	<162	1000 DDD1 11ee eeee	Dn32/s16→Dn(r : q)	- * * * 0
DIVU	W	d=Dn s=	2	<140			2	<144	2	<144	2	<146	4	<148	4	<150	4	<148	6	<152	4	<148	4	<150	4	<144	1000 DDD0 11ee eeee	Dn32/s16→Dn(r : q)	- * * * 0
EOR	B/W	s=Dn d=	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							1011 rrr1 SSEE EEEE	d⊕Dn→d	- * * 0 0
	L	s=Dn d=	2	8			2	16	2	16	2	18	4	20	4	22	4	20	6	24									
EORI	B/W	s=Imm d=	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1010 SSEE EEEE	d⊕#→d	- * * 0 0
	L	s=Imm d=	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36									
EORI CCR	B	s=Imm d=																							4	20	0000 1010 0011 1100	s⊕CCR→CCR	* * * * *
EORI SR	W	s=Imm d=																							4	20	0000 1010 0111 1100	s⊕SR→SR	* * * * *
EXG	L	s=Dn d=	2	6																							1100 DDD1 0100 0DDD 1100 AAA1 0100 1AAA 1100 DDD1 1000 1AAA	s↔d	- - - - -
		s=An d=			2	6																						
		s=Dn d=			2	6																						
EXT	W	d=	2	4																							0100 1000 1000 0DDD	bit 7→bits 15 : 8	- * * 0 0
	L	d=	2	4																							0100 1000 1100 0DDD	bit 15→bits 31 : 16	
ILLEGAL			2	34																							0100 1010 1111 1100	PC→−(SSP), SR→−(SSP), (illegal vector)→PC	- - - - -
JMP		d=					2	8					4	10	4	14	4	10	6	12	4	10	4	14			0100 1110 11EE EEEE	d→PC	- - - - -
JSR		d=					2	16					4	18	4	22	4	18	6	20	4	18	4	22			0100 1110 10EE EEEE	PC→−(SP),d→PC	- - - - -
LEA	L	d=An s=					2	4					4	8	4	12	4	8	6	12	4	8	4	12			0100 AAA1 11ee eeee	s→An	- - - - -
LINK		d_{16}=Imm s=			4	16																					0100 1110 0101 0AAA	An→−(SP), SP→An, SP+disp→SP	- - - - -
LSL, LSR	B/W	count=Dn d=	2	6+2n																							1110 rrrf SS10 1DDD	C ← 0 / X ← 0	* * * 0 *
		count=#1–8 d=	2	6+2n																							1110 QQQf SS00 1DDD	Left:	
	L	count=Dn d=	2	8+2n																							1110 rrrf 1010 1DDD	0 → C / 0 → X	
		count=#1–8 d=	2	8+2n																							1110 QQQf 1000 1DDD	Right:	
	Memory	W count=1 d=					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20							1110 001f 11EE EEEE		
MOVE	B/W	d=Dn s=	2	4	2	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	00XX RRRM MMee eeee	s→d	- * * 0 0
		d=An s=	MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA				
		d=(An) s=	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12			
		d=(An)+ s=	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	4	20	4	16	4	18	4	12			
		d=−(An) s=	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12			
		d=d_{16}(An) s=	4	12	4	12	4	16	4	16	4	18	6	20	6	22	6	20	8	24	6	20	6	22	6	16			
		d=d_8(An,Xn) s=	4	14	4	14	4	18	4	18	4	20	6	22	6	24	6	22	8	26	6	22	6	24	6	18			
		d=Abs.W s=	4	12	4	12	4	16	4	16	4	18	6	20	6	22	6	20	8	24	6	20	6	22	6	16			
		d=Abs.L s=	6	16	6	16	6	20	6	20	6	22	8	24	8	26	8	24	10	28	8	24	8	26	8	20			
MOVE	L	d=Dn s=	2	4	2	4	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	6	12	0010 RRRM MMee eeee	s→d	- * * 0 0
		d=An s=	MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA				
		d=(An) s=	2	12	2	12	2	20	2	20	2	22	4	24	4	26	4	24	6	28	4	24	4	26	6	20			
		d=(An)+ s=	2	12	2	12	2	20	2	20	2	22	4	24	4	26	4	24	4	28	4	24	4	26	6	20			
		d=−(An) s=	2	12	2	12	2	20	2	20	2	22	4	24	4	26	4	24	6	28	4	24	4	26	6	20			
		d=d_{16}(An) s=	4	16	4	16	4	24	4	24	4	26	6	28	6	30	6	28	8	32	6	28	6	30	8	24			
		d=d_8(An,Xn) s=	4	18	4	18	4	26	4	26	4	28	6	30	6	32	6	30	8	34	6	30	6	32	8	26			
		d=Abs.W s=	4	16	4	16	4	24	4	24	4	26	6	28	6	30	6	28	8	32	6	28	6	30	8	24			
		d=Abs.L s=	6	20	6	20	6	28	6	28	6	30	8	32	8	34	8	32	10	36	8	32	8	34	10	28			
MOVE CCR	W	d=CCR s=	2	12			2	16	2	16	2	18	4	20	4	22	4	20	6	24	4	20	4	22	4	16	0100 0100 11ee eeee	s→CCR	* * * * *
MOVE SR	W	d=SR s=	2	12			2	16	2	16	2	18	4	20	4	22	4	20	6	24	4	20	4	22	4	16	0100 0110 11ee eeee	s→SR	* * * * *
		s=SR d=	2	6			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0000 11EE EEEE	SR→d	- - - - -
MOVE USP	L	s=USP d=			2	4																					0100 1110 0110 1AAA	USP→An	- - - - -
		d=USP s=			2	4																					0100 1110 0110 0AAA	An→USP	
MOVEA	W	d=An s=	2	4	2	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	0011 AAA0 01ee eeee	s→An	- - - - -
	L	d=An s=	2	4	2	4	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	6	12	0010 AAA0 01ee eeee		
MOVEM	W	s=Xn... d=					4	8+4n			4	8+4n	6	12+4n	6	14+4n	6	12+4n	8	16+4n							0100 1000 10EE EEEE	Xn...→d	- - - - -
		d=Xn... s=					4	12+4n	4	12+4n			6	16+4n	6	18+4n	6	16+4n	8	20+4n	6	16+4n	6	18+4n			0100 1100 10ee eeee	s→Xn...	
	L	s=Xn... d=					4	8+8n			4	8+8n	6	12+8n	6	14+8n	6	12+8n	8	16+8n							0100 1000 11EE EEEE	Xn...→d	
		d=Xn... s=					4	12+8n	4	12+8n			6	16+8n	6	18+8n	6	16+8n	8	20+8n	6	16+8n	6	18+8n			0100 1100 11ee eeee	s→Xn...	
MOVEP	W	s=Dn d=											4	16													0000 DDD1 1000 1AAA	Dn→d by bytes	- - - - -
		s=d_{16}(An) d=	4	16																							0000 DDD1 0000 1AAA	s→Dn by bytes	
	L	s=Dn d=											4	24													0000 DDD1 1100 1AAA	Dn→d by bytes	
		s=d_{16}(An) d=	4	24																							0000 DDD1 0100 1AAA	s→Dn by bytes	

Addressing Modes

Mnemonic	Size	Address Mode		Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d_{16}(An) #	d_{16}(An) ~	d_8(An,Xn) #	d_8(An,Xn) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~	d_{16}(PC) #	d_{16}(PC) ~	d_8(PC,Xn) #	d_8(PC,Xn) ~	s = Immed d = SR/CC #	s = Immed d = SR/CC ~	Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
ABCD	B	s = Dn	d =	2	6																							1100 RRR1 0000 0rrr	d10 + s10 + X → d	* U * U *
		s = −(An)	d =									2	18															1100 RRR1 0000 1rrr		
ADD	B/W	s = Dn	d =				ADDA	2	12	2	12	2	14	4	16	4	18	4	16	6	20							1101 DDD1 SSEE EEEE	d + Dn → d	* * * * *
		d = Dn	s =	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1101 DDD0 SSee eeee	Dn + s → Dn	
	L	s = Dn	d =				ADDA	2	20	2	20	2	22	4	24	4	26	4	24	6	28							1101 DDD1 10EE EEEE	d + Dn → d	
		d = Dn	s =	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	16	1101 DDD0 10ee eeee	Dn + s → Dn	
ADDA	W	d = An	s =	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12	1101 AAA0 11ee eeee	An + s → An	− − − − −
	L	d = An	s =	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	16	1101 AAA1 11ee eeee		
ADDI	B/W	s = Imm	d =	4	8		ADDA	4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0110 SSEE EEEE	d + # → d	* * * * *
	L	s = Imm	d =	6	16		ADDA	6	28	6	28	6	30	8	32	8	34	8	32	10	36									
ADDQ	B/W	s = Imm3	d =	2	4	2*	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 QQQ0 SSEE EEEE	d + # → d	* * * * *
	L	s = Imm3	d =	2	8	2	8	2	20	2	20	2	22	4	24	4	26	4	24	6	28									
ADDX	B/W	s = Dn	d =	2	4																							1101 RRR1 SS00 0rrr	d + s + X → d	* * * * *
		s = −(An)	d =									2	18															1101 RRR1 SS00 1rrr		
	L	s = Dn	d =	2	8																							1101 RRR1 1000 0rrr		
		s = −(An)	d =									2	30															1101 RRR1 1000 1rrr		
AND	B/W	s = Dn	d =					2	12	2	12	2	14	4	16	4	18	4	16	6	20							1100 DDD1 SSEE EEEE	d <and> Dn → d	− * * 0 0
		d = Dn	s =	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1100 DDD0 SSee eeee	Dn <and> s → Dn	
	L	s = Dn	d =					2	20	2	20	2	22	4	24	4	26	4	24	6	28							1100 DDD1 10EE EEEE	d <and> Dn → d	
		d = Dn	s =	2	8			2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	16	1100 DDD0 10ee eeee	Dn <and> s → Dn	
ANDI	B/W	s = Imm	d =	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0010 SSEE EEEE	d <and> # → d	− * * 0 0
	L	s = Imm	d =	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36									
ANDI CCR	B	s = Imm	d =																							4	20	0000 0010 0011 1100	s <and> CCR → CCR	* * * * *
ANDI SR	W	s = Imm	d =																							4	20	0000 0010 0111 1100	s <and> SR → SR	* * * * *
ASL, ASR	B/W	count = Dn	d =	2	6 + 2n																							1110 rrrf SS10 0DDD		* * * * *
		count = #1–8	d =	2	6 + 2n																							1110 QQQf SS00 0DDD		
	L	count = Dn	d =	2	8 + 2n																							1110 rrrf 1010 0DDD		
		count = #1–8	d =	2	8 + 2n																							1110 QQQf 1000 0DDD		
Memory	W	count = 1	d =					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20							1110 0001 11EE EEEE		
Bcc	B		d_8 =																					branch taken	2	10		0110 CCCC PPPP PPPP	If cc true, then PC + disp → PC	− − − − −
	W		d_{16} =																					branch not taken	2	8				
																								branch taken	4	10				
																								branch not taken	4	12				
BCHG	B	bit# = Dn	d =					2	12	2	12	2	14	4	16	4	18	4	16	6	20							0000 rrr1 01EE EEEE	~(bit#) of d → Z,	− − * − −
		bit# = Imm	d =					4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1000 01EE EEEE	~(bit#) of d → (bit#) of d	
	L	bit# = Dn	d =	2	<8																							0000 rrr1 01EE EEEE		
		bit# = Imm	d =	4	<12																							0000 1000 01EE EEEE		
BCLR	B	bit# = Dn	d =					2	12	2	12	2	14	4	16	4	18	4	16	6	20							0000 rrr1 10EE EEEE	~(bit#) of d → Z,	− − * − −
		bit# = Imm	d =					4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1000 10EE EEEE	0 → (bit#) of d	
	L	bit# = Dn	d =	2	<10																							0000 rrr1 10EE EEEE		
		bit# = Imm	d =	4	<14																							0000 1000 10EE EEEE		
BRA	B		d_8 =																							2	10	0110 0000 PPPP PPPP	PC + disp → PC	− − − − −
	W		d_{16} =																							4	10			
BSET	B	bit# = Dn	d =					2	12	2	12	2	14	4	16	4	18	4	16	6	20							0000 rrr1 11EE EEEE	~(bit#) of d → Z,	− − * − −
		bit# = Imm	d =					4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 1000 11EE EEEE	1 → (bit#) of d	
	L	bit# = Dn	d =	2	<8																							0000 rrr1 11EE EEEE		
		bit# = Imm	d =	4	<12																							0000 1000 11EE EEEE		
BSR	B		d_8 =																							2	18	0110 0001 PPPP PPPP	PC → −(SP), PC + disp → PC	− − − − −
	W		d_{16} =																							4	18			
BTST	B	bit# = Dn	d =					2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14			0000 rrr1 00EE EEEE	~(bit#) of d → Z	− − * − −
		bit# = Imm	d =					4	12	4	12	4	14	6	16	6	18	6	16	8	20	6	16	6	18			0000 1000 00EE EEEE		
	L	bit# = Dn	d =	2	<8																							0000 rrr1 00EE EEEE		
		bit# = Imm	d =	4	<10																							0000 1000 00EE EEEE		
CHK	W	d = Dn	s(bound) =	2	<<44	trap		2	<<48	2	<<48	2	<<50	4	<<52	4	<<54	4	<<52	6	<<56	4	<<52	4	<<54	4	<<48	0100 DDD1 10ee eeee	If Dn<0 or Dn>(bound), then trap	− * U U U
				2	10	no trap		2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	4	14			
CLR	B/W		d =	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0010 SSEE EEEE	0 → d	− 0 1 0 0
	L		d =	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28									
CMP	B/W	d = Dn	s =	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1011 DDD0 SSee eeee	Dn − s	− * * * *
	L	d = Dn	s =	2	6	2	6	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14			
CMPA	W	d = An	s =	2	6	2	6	2	10	2	10	2	12	4	14	4	16	4	14	6	18	4	14	4	16	4	10	1011 AAA0 11ee eeee	An − s	− * * * *
	L	d = An	s =	2	6	2	6	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1011 AAA1 11ee eeee		
CMPI	B/W	s = Imm	d =	4	8		CMPA	4	12	4	12	4	14	6	16	6	18	6	16	8	20							0000 1100 SSEE EEEE	d − #	− * * * *
	L	s = Imm	d =	6	14		CMPA	6	20	6	20	6	22	8	24	8	26	8	24	10	28									
CMPM	B/W	s = (An)+	d =							2	12																	1011 RRR1 SS00 1rrr	d − s	− * * * *
	L	s = (An)+	d =							2	20																			

Mnemonic	Size	Address Mode		Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d_{16}(An) #	d_{16}(An) ~	d_8(An,Xn) #	d_8(An,Xn) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~	d_{16}(PC) #	d_{16}(PC) ~	d_8(PC,Xn) #	d_8(PC,Xn) ~	s = Immed d = SR/CC #	s = Immed d = SR/CC ~	Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
MOVEQ	L	s = Imm8	d =	2	4																							0111 DDD0 QQQQ QQQQ	# → Dn	− * * 0 0
MULS	W	d = Dn	s =	2	<70			2	<74	2	<74	2	<76	4	<78	4	<80	4	<78	6	<82	4	<78	4	<80	4	<74	1100 DDD1 11ee eeee	Dn × s → Dn	− * * 0 0
MULU	W	d = Dn	s =	2	<70			2	<74	2	<74	2	<76	4	<78	4	<80	4	<78	6	<82	4	<78	4	<80	4	<74	1100 DDD0 11ee eeee	Dn × s → Dn	− * * 0 0
NBCD	B		d =	2	6			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 1000 00EE EEEE	0 − d10 − X → d	* U * U *
NEG	B/W		d =	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0100 SSEE EEEE	0 − d → d	* * * * *
	L		d =	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28									
NEGX	B/W		d =	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0000 SSEE EEEE	0 − d − X → d	* * * * *
	L		d =	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28									
NOP				2	4																							0100 1110 0111 0001	none	− − − − −
NOT	B/W		d =	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0110 SSEE EEEE	~d → d	− * * 0 0
	L		d =	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28									
OR	B/W	s = Dn	d =					2	12	2	12	2	14	4	16	4	18	4	16	6	20							1000 DDD1 SSEE EEEE	d <or> Dn → d	− * * 0 0
		d = Dn	s =	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1000 DDD0 SSee eeee	Dn <or> s → Dn	
	L	s = Dn	d =					2	20	2	20	2	22	4	24	4	26	4	24	6	28							1000 DDD1 10EE EEEE	d <or> Dn → d	
		d = Dn	s =	2	8			2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	16	1000 DDD0 10ee eeee	Dn <or> s → Dn	
ORI	B/W	s = Imm	d =	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0000 SSEE EEEE	d <or> # → D	− * * 0 0
	L	s = Imm	d =	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36									
ORI CCR	B	s = Imm8	d =																							4	20	0000 0000 0011 1100	s <or> CCR → CCR	* * * * *
ORI SR	W	s = Imm	d =																							4	20	0000 0000 0111 1100	s <or> SR → SR	* * * * *
PEA	L		d =					2	12					4	16	4	20	4	16	6	20	4	16	4	20			0100 1000 01ee eeee	address of d → −(SP)	− − − − −
RESET				2	132																							0100 1110 0111 0000	assert RESET pin	− − − − −
ROL, ROR	B/W	count = Dn	d =	2	6 + 2n																							1110 rrrf SS11 1DDD		− * * 0 *
		count = #1–8	d =	2	6 + 2n																							1110 QQQf SS01 1DDD		
	L	count = Dn	d =	2	8 + 2n																							1110 rrrf 1011 1DDD		
		count = #1–8	d =	2	8 + 2n																							1110 QQQf 1001 1DDD		
Memory	W	count = 1	d =					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20							1100 0111 11EE EEEE		
ROXL, ROXR	B/W	count = Dn	d =	2	6 + 2n																							1110 rrrf SS11 0DDD		* * * 0 *
		count = #1–8	d =	2	6 + 2n																							1110 QQQf SS01 0DDD		
	L	count = Dn	d =	2	8 + 2n																							1110 rrrf 1011 0DDD		
		count = #1–8	d =	2	8 + 2n																							1110 QQQf 1001 0DDD		
Memory	W	count = 1	d =					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20							1110 010f 11EE EEEE		
RTE				2	20																							0100 1110 0111 0011	(SP) + → SR, (SP) + → PC	* * * * *
RTR				2	20																							0100 1110 0111 0111	(SP) + → CC, (SP) + → PC	* * * * *
RTS				2	16																							0100 1110 0111 0101	(SP) + → PC	− − − − −
SBCD	B	s = Dn	d =	2	6																							1000 RRR1 0000 0rrr	d10 − s10 − X → d	* U * U *
		s = −(An)	d =									2	18															1000 RRR1 0000 1rrr		
Scc	B	cc = True	d =	2	6			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 CCCC 11EE EEEE	If cc true, then 1s → d	− − − − −
		cc = False	d =	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20								else 0s → d	
STOP	W		d =																							4	4	0100 1110 0111 0010	# → SR, wait for interrupt	* * * * *
SUB	B/W	s = Dn	d =				SUBA	2	12	2	12	2	14	4	16	4	18	4	16	6	20							1001 DDD1 SSEE EEEE	d − Dn → d	* * * * *
		d = Dn	s =	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1001 DDD0 SSee eeee	Dn − s → Dn	
	L	s = Dn	d =				SUBA	2	20	2	20	2	22	4	24	4	26	4	24	6	28							1001 DDD1 10EE EEEE	d − Dn → d	
		d = Dn	s =	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	16	1001 DDD0 10ee eeee	Dn − s → Dn	
SUBA	W	d = An	s =	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12	1001 AAA0 11ee eeee	An − s → An	− − − − −
	L	d = An	s =	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	16	1001 AAA1 11ee eeee		
SUBI	B/W	s = Imm	d =	4	8		SUBA	4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0100 SSEE EEEE	d − # → d	* * * * *
	L	s = Imm	d =	6	16		SUBA	6	28	6	28	6	30	8	32	8	34	8	32	10	36									
SUBQ	B/W	s = Imm3	d =	2	4	2*	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 QQQ1 SSEE EEEE	d − # → d	* * * * *
	L	s = Imm3	d =	2	8	2	8	2	20	2	20	2	22	4	24	4	26	4	24	6	28									
SUBX	B/W	s = Dn	d =	2	4																							1001 RRR1 SS00 0rrr	d − s − X → d	* * * * *
		s = −(An)	d =									2	18															1001 RRR1 SS00 1rrr		
	L	s = Dn	d =	2	8																							1001 RRR1 1000 0rrr		
		s = −(An)	d =									2	30															1001 RRR1 1000 1rrr		
SWAP	W		d =	2	4																							0100 1000 0100 0DDD	Dn(31:16) ⟷ Dn(15:0)	− * * 0 0
TAS	B		d =	2	4			2	14	2	14	2	16	4	18	4	20	4	18	6	22							0100 1010 11EE EEEE	test d → CC, 1 → bit 7 of d	− * * 0 0
TRAP		vector = #0–15		2	38																							0100 1110 0100 VVVV	PC → −(SSP), SR → −(SSP), (1 of 16 trap vectors) → PC	− − − − −
TRAPV				2	34	trap taken																						0100 1110 0111 0110	If V = 1, then PC → −(SSP), SR → −(SSP), (TRAPV vector) → PC, else NOP	− − − − −
				2	4	trap not taken																								
TST	B/W		d =	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16							0100 1010 SSEE EEEE	test d → CC	− * * 0 0
	L		d =	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20									
UNLK						2	12																					0100 1110 0101 1AAA	An → SP, (SP) + → An	− − − − −

General Notes:

* Word Only
# Number of Bytes in Instruction
~ Execution Time in Clock Periods
s Source (s10 = base 10 operand)
d Destination (d10 = base 10 operand)
< Value is Maximum Number
' Complement (invert)
d_8 8-Bit Displacement
d_{16} 16-Bit Displacement
Imm Immediate Data
Imm3 Immediate Data, 3 Bits
Imm8 Immediate Data, 8 Bits

Opcode Bit Pattern Codes:

A Address Register Number
C Test Condition
D Data Register Number
E Destination Effective Address
e Source Effective Address
f Direction: 0 = Right 1 = Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
R Destination Register
r Source Register
S Size: 00 = Byte 01 = Word 10 = Long
V Vector Number
XX Move size: 01 = Byte 11 = Word

Condition Code Notation:

* Set according to result of operation.
— Not affected by operation.
0 Cleared
1 Set
U Undefined after operation.