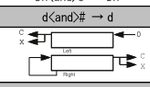
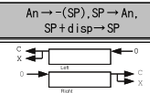
Mnemonic	Size	Address Mode	Dn	An	(An)	(An) +	-(An)	d_{16}(An)	d_8(An,Xn)	Abs.W (Abs).W	Abs.L (Abs).L	d_{16}(PC)	d_8(PC,Xn)	s = immed d = SR/CC	Opcode Bit Pattern	Boolean	Condition Codes
			#	#	#	#	#	#	#	#	#	#	#	#	5432 1098 7654 3210		X N Z V C
ADD	L	s= Dn d=			2	2	2	4	4	4	6				1101 DDD1 10EE EEEE	d + Dn → d	* * * * *
		d= Dn s=	2	2	2	2	2	4	4	4	6	4	4	6	1101 DDD0 10ee eeee	Dn + s → d	
ADDA	L	d= An s=	2	2	2	2	2	4	4	4	6	4	4	6	1101 AAA1 11ee eeee	An + s → An	- - - - -
ADDI	L	s= Imm d=	6												0000 0110 10EE EEEE	Dn + # → Dn	* * * * *
ADDQ	L	s= Imm3 d=	2		2	2	2	2	4	4	6	6			0101 QQQ0 10EE EEEE	d + # → d	* * * * *
ADDX	L	s= Dn d=	2												1101 RRR1 1000 0rrr	d + s + X → d	* * * * *
AND	L	s= Dn d=			2	2	2	4	4	4	6				1100 DDD1 10EE EEEE	d<and>Dn → d	- * * 0 0
		d= Dn s=	2		2	2	2	4	4	4	6	4	4	6	1100 DDD0 10ee eeee	Dn<and>s → Dn	
ANDI	L	s= Imm d=	6												0000 0010 10EE EEEE	d<and># → d	- * * 0 0
ASL, ASR	L	count= Dn d=	2												1110 rrrf 1010 0DDD		* * * 0 *
		count= #1-8 d=	2												1110 QQQf 1000 0DDD		
Bcc	B	d_8=											branch taken	2	0110 CCCC PPPP PPPP	if cc true, then PC + disp → PC	- - - - -
													branch not taken	2			
	W	d_{16}=											branch taken	4			
													branch not taken	4			
BCHG	B	bit# = Dn d=			2	2	2	4	4	4	6				0000 rrr1 01EE EEEE	$\overline{(bit\#)}$ of d→ Z, $\overline{(bit\#)}$ of d→ (bit#) of d	- - * - -
		bit# = Imm d=			4	4	4	6							0000 1000 01EE EEEE		
	L	bit# = Dn d=	2												0000 rrr1 01EE EEEE		
		bit# = Imm d=	4												0000 1000 01EE EEEE		
BCLR	B	bit# = Dn d=			2	2	2	4	4	4	6				0000 rrr1 10EE EEEE	$\overline{(bit\#)}$ of d→ Z, 0→ (bit#)of d	- - * - -
		bit# = Imm d=			4	4	4	6							0000 1000 10EE EEEE		
	L	bit# = Dn d=	2												0000 rrr1 10EE EEEE		
		bit# = Imm d=	4												0000 1000 10EE EEEE		
BRA	B	d_8=												2	0110 0000 PPPP PPPP	PC + disp → PC	- - - - -
	W	d_{16}=												4			
BSET	B	bit# = Dn d=			2	2	4	4	4	4	6				0000 rrr1 11EE EEEE	$\overline{(bit\#)}$ of d→ Z, 1→ (bit#) of d	- - * - -
		bit# = Imm d=			4	4	4	6							0000 1000 11EE EEEE		
	L	bit# = Dn d=	2												0000 rrr1 11EE EEEE		
		bit# = Imm d=	4												0000 1000 11EE EEEE		
BSR	B	d_8=												2	0110 0001 PPPP PPPP	PC →(SP), PC + disp → PC	- - - - -
	W	d_{16}=												4			
BTST	B	bit# = Dn d=			2	2	2	4	4	4	6	4	4		0000 rrr1 00EE EEEE	$\overline{(bit\#)}$ of d → Z	- - * - -
		bit# = Imm d=			4	4	4	6							0000 1000 00EE EEEE		
	L	bit# = Dn d=	2												0000 rrr1 00EE EEEE		
		bit# = Imm d=	4												0000 1000 00EE EEEE		
CLR	B/W	d=	2		2	2	2	4	4	4	6				0100 0010 SSEE EEEE	0 → d	- 0 1 0 0
	L	d=	2		2	2	2	4	4	4	6						
CMP	B/W	d= Dn s=	2	2*	2	2	2	4	4	4	6	4	4	4	1011 DDD0 Ssee eeee	Dn-s	- * * * *
	L	d= Dn s=	2	2	2	2	2	4	4	4	6	4	4	6			
CMPA	W	d= An s=	2	2	2	2	2	4	4	4	6	4	4	4	1011 AAA0 11ee eeee	An-s	- * * * *
	L	d= An s=	2	2	2	2	2	4	4	4	6	4	4	6	1011 AAA1 11ee eee		
CMPI	B/W	s= Imm d=	4												0000 1100 SSEE EEEE	d-#	- * * * *
	L	s= Imm d=	6														
DIVS	W	d= Dn s=	2		2	2	2	4	4	4	6	4	4	4	1000 DDD1 11ee eeee	Dn_{32}/S_{16}→$Dn(R_{16}:Q_{16})$	- * * * 0
	L	d= Dn s=	4		4	4	4	6							0100 1100 01ee eeee 0DDD 1000 0000 0DDD	Dn_{32}/S_{32}→$Dn(Q_{32})$	
DIVU	W	d= Dn s=	2		2	2	2	4	4	4	6	4	4	4	1000 DDD0 11ee eeee	Dn_{32}/S_{16}→$(R_{16}:Q_{16})$	- * * * 0
	L	d= Dn s=	4		4	4	4	6							0100 1100 01ee eeee 0DDD 0000 0000 0DDD	Dn_{32}/S_{32}→$Dn(Q_{32})$	
EOR	L	s= Dn d=	2		2	2	2	4	4	4	6				1011 rrr1 10EE EEEE	d ⊕ Dn → d	- * * 0 0
EORI	L	s= Imm d=	6												0000 1010 1000 0DDD	d ⊕ #→ d	- * * 0 0
EXT	W	d=	2												0100 1000 1000 0DDD	bit 7 → bits 15:8	- * * 0 0
	L	d=	2												0100 1000 1100 0DDD	bit 15 → bits 31:16	
ILLEGAL															0100 1010 1111 1100	PC → -(SSP)	- - - - -
																SR → -(SSP)	
																Vector offset → -(SSP)	
																(illegal vector) → PC	
JMP		d=			2			4	4	4	6	4	4		0100 1110 11EE EEEE	d → PC	- - - - -
JSR		d=			2			4	4	4	6	4	4		0100 1110 10EE EEEE	PC →-(SP),d→ PC	- - - - -
LEA	L	d= An s=			2			4	4	4	6	4	4		0100 AAA1 11ee eeee	s → An	- - - - -
LINK	W	d_{16}= Imm s=		4											0100 1110 0101 0AAA	An→-(SP),SP→An, SP+disp→SP	- - - - -
LSL,LSR	L	count=Dn d=	2												11110 rrrf 1010 1DDD		* * * 0 *
		count=#1-8 d=	2												1110 QQQf 1000 1DDD		
MOVE	B/W	d=Dn s=	2	2	2	2	2	4	4	4	6	4	4	4	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	2	2	2	2	4	4	4	6	4	4	4			
		d=(An)+ s=	2	2	2	2	2	4	4	4	6	4	4	4			
		d=-(An) s=	2	2	2	2	2	4	4	4	6	4	4	4			
		d=d_{16}(An) s=	4	4	4	4	4	6				6					
		d=d_8(An,Xn*SF)s=	4	4	4	4	4										
		d=Abs.W s=	4	4	4	4	4										
		d= Abs.L s=	6	6	6	6	6										
	L	d=Dn s=	2	2	2	2	2	4	4	4	6	4	4	6	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	2	2	2	2	4	4	4	6	4	4	6			
		d= (An)+ s=	2	2	2	2	2	4	4	4	6	4	4	6			
		d=-(An) s=	2	2	2	2	2	4	4	4	6	4	4	6			
		d=d_{16}(An) s=	4	4	4	4	4	6				6					
		d=d_8(An,Xn*SF) s=	4	4	4	4	4										
		d=Abs.W s=	4	4	4	4	4										
		d=Abs.L s=	6	6	6	6	6										
MOVE CCR	W	d=CCR s=	2											4	0100 0100 11ee eeee	lower 8 bits of s → CCR	* * * * *
	W	s=CCR d=	2												0100 0010 1100 0DDD	CCR → Dn	
MOVEA	W	d= An s=	2	2	2	2	2	4	4	4	6	4	4	4	0011 AAA0 01ee eeee	s → An	- - - - -
	L	d= An s=	2	2	2	2		4	4	4	6	4	4	6	0010 AAA0 01ee eeee		
MOVEM	L	s= Xn... d=			4			6							0100 1000 11EE EEEE	Xn... → d	- - - - -
		d= Xn... d=			4			6							0100 1100 11ee eeee	s → Xn...	
MOVEQ	L	s= Imm8 d=	2												0111 DDD0 QQQQ QQQQ	# → Dn (sign ext)	- * * 0 0
MULS	W	d= Dn s=	2		2	2	2	4	4	4	6	4	4	4	1100 DDD1 11ee eeee	Dn_{16}*s_{16} → Dn_{32}	- * * 0 0
	L	d= Dn s=	4		4	4	4	6							0100 1100 00ee eeee 0DDD 1000 0000 0000	s_{32}*s_{32} → s_{32}	
MULU	W	d= Dn s=	2		2	2	2	4	4	4	6	4	4	4	1100 DDD0 11ee eeee	Dn_{16}*s_{16} → Dn_{32}	- * * 0 0
	L	d= Dn s=	4		4	4	4	6							0100 1100 00ee eeee 0DDD 0000 0000 0000	Dn_{32}*s_{32} → Dn_{32}	
NEG	L	d=	2												0100 0100 1000 0DDD	0-Dn → Dn	* * * * *
NEGX	L	d=	2												0100 0000 1000 0DDD	0-Dn-X → Dn	* * * * *
NOP															0100 1110 0111 0001	none	- - - - -
Mnemonic	Size	Address Mode	Dn	An	(An)	(An) +	(An) -	d_{16}(An)	d_8(An,Xn)	Abs.W (Abs).W	Abs.L (Abs).L	d_{16}(PC)	d_8(PC,Xn)	s = immed d = SR/CC	Opcode Bit Pattern	Boolean	Condition Codes
			#	#	#	#	#	#	#	#	#	#	#	#	5432 1098 7654 3210		X N Z V C

General Notes

- * Word Only
- # Number of Bytes in Instruction
- s Source (s10 = base 10 operand)
- d Destination (d10 = base 10 operand)
- < Value is Maximum Number
- $\overline{}$ 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

Mnemonic	Size	Address Mode	Dn	An	(An)	(An) +	-(An)	d_{16}(An)	d_8(An,Xn)	Abs.W (Abs).W	Abs.L (Abs).L	d_{16}(PC)	d_8(PC,Xn)	s = immed d = SR/CC	Opcode Bit Pattern	Boolean	Condition Codes
			#	#	#	#	#	#	#	#	#	#	#	#	5432 1098 7654 3210		X N Z V C
NOT	L	d=	2												0100 0110 1000 0DDD	$\overline{Dn} \rightarrow Dn$	- * * 0 0
OR	L	s= Dn d=			2	2	2	4	4	4	6				1000 DDD1 10EE EEEE	d<or>Dn → d	- * * 0 0
		d= Dn s=	2		2	2	2	4	4	4	6	4	4	6	1000 DDD0 10ee eeee	Dn<or>s → Dn	- * * 0 0
ORI	L	s= Imm d=	6												0000 0000 1000 0DDD	Dn<or># → Dn	- * * 0 0
PEA	L	d=			2			4	4	4	6	4	4		0100 1000 01ee eeee	address of d →(SP)	- - - - -
RTE															0100 1110 0111 0011	(SP+2)→ SP (SP+4)→ SR SP + 8→ PC Adjust stack according to format	* * * * *
RTS			2												0100 1110 0111 0101	(SP)+→ PC	- - - - -
STOP		d=												4	0100 1110 0111 0010	#→SR, wait for interrupt	* * * * *
SUB	L	s= Dn d=		SUBA	2	2	2	4	4	4	6				1001 DDD1 10EE EEEE	d-Dn→ d	* * * * *
		d= Dn s=	2	2	2	2	2	4	4	4	6	4	4	6	1001 DDD0 10ee eeee	Dn-s→ Dn	
SUBA	L	d= An s=	2	2	2	2	2	4	4	4	6	4	4	6	1001 AAA1 11ee eeee	An-s→ An	- - - - -
SUBI	L	s= Imm d=	6	SUBA											0000 0100 1000 0DDD	d-#→ d	* * * * *
SUBQ	L	s= Imm3 d=	2	2	2	2	2	4	4	4	6				0101 QQQ1 10EE EEEE	d-#→d	* * * * *
SUBX	L	s= Dn d=	2												1001 RRR1 1000 0rrr	d-s-X→ d	* * * * *
SWAP	W	d=	2												0100 1000 0100 0DDD	Dn(31:16) ↔ Dn(15:0)	- * * 0 0
TAS	B	d=			2	2	2	4	4	4	6				0100 1010 11EE EEEE	test d→cc, 1→bit 7 of d	- * * 0 0
TRAP		vector=#0-15	2												0100 1110 0100 VVVV	PC → -(SSP) SR → -(SSP) Format/Offset→-(SSP) (VBR+0x80+4*n)→ PC	- - - - -
TST	B	d=	2		2	2	2	4	4	4	6	4	4	4	0100 1010 SSEE EEEE	test d→CC	- * * 0 0
	W	d=	2	2	2	2	2	4	4	4	4	4	4	4			
	L	d=	2	2	2	2	2	4	4	4	4	4	4	6			
UNLK				2											0100 1110 0101 1AAA	An → SP, (SP)+→ An	- - - - -
Mnemonic	Size	Address Mode	Dn	An	(An)	(An) +	(An) -	d_{16}(An)	d_8(An,Xn)	Abs.W (Abs).W	Abs.L (Abs).L	d_{16}(PC)	d_8(PC,Xn)	s = immed d = SR/CC	Opcode Bit Pattern	Boolean	Condition Codes
			#	#	#	#	#	#	#	#	#	#	#	#	5432 1098 7654 3210		X N Z V C

General Notes

- * Word Only
- # Number of Bytes in Instruction
- 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

Powers of 16, Powers of 2

16^n m=	2^n n=	Value	16^n m=	2^n n=	Value
0	0	1	4	16	65 536
	1	2		17	131 072
	2	4		18	262 144
	3	8		19	524 288
1	4	16	5	20	1 048 576
	5	32		21	2 097 152
	6	64		22	4 194 304
	7	128		23	8 388 608
2	8	256	6	24	16 777 216
	9	512		25	33 554 432
	10	1 024		26	67 108 864
	11	2 048		27	134 217 728
3	12	4 096	7	28	268 435 456
	13	8 192		29	536 870 912
	14	16 384		30	1 07 374 1824
	15	32 768		31	2 147 483 648
			8	32	4 294 967 296

Exception Vector Assignment

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
0	0x000	-	Initial Stack Pointer
1	0x004	-	Initial Program Counter
2	0x008	Fault	Access Error
3	0x00C	Fault	Address Error
4	0x010	Fault	Illegal Instruction
5	0x014	Fault	Divide by Zero
6-7	0x018-0x01C	-	Reserved
8	0x020	Fault	Privilege Violation
9	0x024	Next	Trace
10	0x028	Fault	Unimplemented line-a opcode
11	0x02C	Fault	Unimplemented line-f opcode
12	0x030	Next	Debug Interrupt
13	0x034	-	Reserved
14	0x038	Fault	Format Error
15-23	0x03C-0x05C	-	Reserved
24	0x060	Next	Spurious Interrupt
25-31	0x064-0x07C	-	Reserved
32-47	0x080-0x0BC	Next	Trap #0-15 Instructions
48-63	0x0C0-0x0FC	-	Reserved
64-255	0x100-0x3FC	Next	User-defined Interrupts

"Fault" refers to the PC of the instruction that caused the exception;
"Next" refers to the PC of the next instruction that follows the instruction that caused the fault

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 Word (One Word, Specifies Operation and Modes)
Extension Word (If Any)
Extension Word (If Any)

ASCII Character Set (7-Bit Code)

MSD, Dig

LSD, 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

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 Zero Divide	Exception processing is started by normal instruction execution. (Lowest Priority)

NOTES:

- Sm Source Operand 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

Condition Code Computation

Operations	X	N	Z	V	C	Special Definition
ADD, ADDI, ADDQ	*	*	*	$V = Sm \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$		
ADDX	*	*	?	$V = Sm \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$		
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	-	*	*	0	0	
SUB, SUBI, SUBQ	*	*	*	$V = Sm \cdot \overline{Dm} \cdot \overline{Rm} + \overline{Sm} \cdot Dm \cdot Rm$ $C = Sm \cdot \overline{Dm} + \overline{Rm} \cdot \overline{Dm} + Sm \cdot Rm$		
SUBX	*	*	?	$V = Sm \cdot \overline{Dm} \cdot \overline{Rm} + \overline{Sm} \cdot Dm \cdot Rm$ $C = Sm \cdot \overline{Dm} + \overline{Rm} \cdot \overline{Dm} + Sm \cdot Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$		
CMP, CMPI	-	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + \overline{Rm} \cdot \overline{Dm} + Sm \cdot Rm$
DIVS, DIVU	-	*	*	?	?	V = Division Overflow
MULS, MULU	*	*	?	0	0	
NEG	*	*	?	?	?	$V = Dm \cdot Rm$ $C = Dm + Rm$
NEGX	*	*	?	?	?	$V = Dm \cdot Rm$ $C = Dm + Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
BTST, BCHG, BSET, BCLR	-	-	?	-	-	$Z = \overline{Dn}$
ASL	*	*	?	?	?	$V = (Dm \cdot (D_{m-1} + \ldots + D_0) + \overline{Dm} \cdot (D_{m-1} + \ldots + D_0))$ $C = D_{m-(r-1)}$
LSL	*	*	0	?	0	$C = D_{r-1}$
ASR, LSR	*	*	?	0	?	$C = D_{r-1}$

Effective Addressing Mode Categories

Type	Mode	Register	Generation	Assembler Syntax
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*SF) + d_8	d_8(An,Xn*SF)
Absolute Short	111	000	EA = (Next Word)	xxxx.W (xxxx).W
Absolute Long	111	001	EA = (Next Two Words)	xxxxxxxx.L (xxxxxxxx).L
PC Relative with Offset	111	010	EA = (PC) + d_{16}	d_{16}(PC)
PC Relative with Index and Offset	111	011	EA = (PC) + (Xn*SF) + d_8	d_8(PC,Xn*SF)
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
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
Dn = Data Registers
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

Status Register