Partial Set of ColdFire Instuctions

from

http://www.freescale.com/files/dsp/doc/ref_manual/CFPRM.pdf

First appeared in ISA_A

Operation: Source + Destination → Destination

Assembler Syntax: ADD.L <ea>y,Dx
ADD.L Dy,<ea>x

Attributes: Size = longword

Description: Adds the source operand to the destination operand using binary addition and stores the result in the destination location. The size of the operation may only be specified as a longword. The mode of the instruction indicates which operand is the source and which is the destination as well as the operand size.

The Dx mode is used when the destination is a data register; the destination <ea>x mode is invalid for a data register.

In addition, ADDA is used when the destination is an address register. ADDI and ADDQ are used when the source is immediate data.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X Set the same as the carry bit
N Set if the result is negative; cleared otherwise
Z Set if the result is zero; cleared otherwise
V Set if an overflow is generated; cleared otherwise
C Set if a carry is generated; cleared otherwise

Instruction Format:

15	14	13	12	11 10 9	8 7 6	5 4 3	2 1 0
1	1	0	1	Register	Opmode	Effective Address	
						Mode	Register

Instruction Fields:

- Register field—Specifies the data register.
- Opmode field:

Byte	Word	Longword	Operation
—	—	010	<ea>y + Dx → Dx
—	—	110	Dy + <ea>x → <ea>x

Instruction Fields (continued):

- Effective Address field—Determines addressing mode
  - For the source operand <ea>y, use addressing modes listed in the following table:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dy	000	reg. number:Dy	(xxx).W	111	000
Ay	001	reg. number:Ay	(xxx).L	111	001
(Ay)	010	reg. number:Ay	#<data>	111	100
(Ay)+	011	reg. number:Ay			
–(Ay)	100	reg. number:Ay			
(d_{16},Ay)	101	reg. number:Ay	(d_{16},PC)	111	010
(d_8,Ay,Xi)	110	reg. number:Ay	(d_8,PC,Xi)	111	011

  - For the destination operand <ea>x, use addressing modes listed in the following table:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dx	—	—	(xxx).W	111	000
Ax	—	—	(xxx).L	111	001
(Ax)	010	reg. number:Ax	#<data>	—	—
(Ax)+	011	reg. number:Ax			
–(Ax)	100	reg. number:Ax			
(d_{16},Ax)	101	reg. number:Ax	(d_{16},PC)	—	—
(d_8,Ax,Xi)	110	reg. number:Ax	(d_8,PC,Xi)	—	—

ADDA

Add Address
First appeared in ISA_A

ADDA

Operation: Source + Destination → Destination

Assembler Syntax: ADDA.L <ea>y,Ax

Attributes: Size = longword

Description: Operates similarly to ADD, but is used when the destination register is an address register rather than a data register. Adds the source operand to the destination address register and stores the result in the address register. The size of the operation is specified as a longword.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Destination Register, Ax			1	1	1	Source Effective Address					
										Mode			Register		

Instruction Fields:

- Destination Register field—Specifies the destination register, Ax.
- Source Effective Address field— Specifies the source operand; use addressing modes listed in the following table:

Addressing Mode	Mode	Register
Dy	000	reg. number:Dy
Ay	001	reg. number:Ay
(Ay)	010	reg. number:Ay
(Ay)+	011	reg. number:Ay
–(Ay)	100	reg. number:Ay
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	110	reg. number:Ay

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d_{16},PC)	111	010
(d_8,PC,Xi)	111	011

Bcc

Branch Conditionally

Bcc

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	Condition				8-bit displacement							
16-bit displacement if 8-bit displacement = 0x00															
32-bit displacement if 8-bit displacement = 0xFF															

Instruction Fields:

- Condition field—Binary encoding for one of the conditions listed in the table.
- 8-bit displacement field—Two's complement integer specifying the number of bytes between the branch and the next instruction to be executed if the condition is met.
- 16-bit displacement field—Used when the 8-bit displacement contains 0x00.
- 32-bit displacement field—Used when the 8-bit displacement contains 0xFF.

Bcc

Branch Conditionally
First appeared in ISA_A
.L First appeared in ISA_B

Bcc

Operation: If Condition True
Then PC + d_n → PC

Assembler Syntax: Bcc.sz <label>

Attributes: Size = byte, word, longword (longword supported starting with ISA_B)

Description: If the condition is true, execution continues at (PC) + displacement. Branches can be forward, with a positive displacement, or backward, with a negative displacement. PC holds the address of the instruction word for the Bcc instruction, plus two. The displacement is a two's-complement integer that represents the relative distance in bytes from the current PC to the destination PC. If the 8-bit displacement field is 0, a 16-bit displacement (the word after the instruction) is used. If the 8-bit displacement field is 0xFF, the 32-bit displacement (longword after the instruction) is used. A branch to the next immediate instruction uses 16-bit displacement because the 8-bit displacement field is 0x00.

Condition code specifies one of the following tests, where C, N, V, and Z stand for the condition code bits CCR[C], CCR[N], CCR[V] and CCR[Z], respectively:

Code	Condition	Encoding	Test	Code	Condition	Encoding	Test
CC(HS)	Carry clear	0100	$\overline{C}$	LS	Lower or same	0011	C | Z
CS(LO)	Carry set	0101	C	LT	Less than	1101	N&$\overline{V}$ | $\overline{N}$&V
EQ	Equal	0111	Z	MI	Minus	1011	N
GE	Greater or equal	1100	N&V | $\overline{N}$&$\overline{V}$	NE	Not equal	0110	$\overline{Z}$
GT	Greater than	1110	N&V&Z | $\overline{N}$&$\overline{V}$&Z	PL	Plus	1010	$\overline{N}$
HI	High	0010	$\overline{C}$&$\overline{Z}$	VC	Overflow clear	1000	$\overline{V}$
LE	Less or equal	1111	Z | N&$\overline{V}$ | $\overline{N}$&V	VS	Overflow set	1001	V

Condition Codes: Not affected

JSR

Jump to Subroutine
First appeared in ISA_A

JSR

Operation: SP – 4 → SP; nextPC → (SP); Destination Address → PC

Assembler Syntax: JSR <ea>y

Attributes: Unsized

Description: Pushes the longword address of the instruction immediately following the JSR instruction onto the system stack. Program execution then continues at the address specified in the instruction.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	0	Source Effective Address					
										Mode				Register	

Instruction Field:

- Source Effective Address field—Specifies the address of the next instruction, <ea>y; use the control addressing modes in the following table:

Addressing Mode	Mode	Register
Dy	—	—
Ay	—	—
(Ay)	010	reg. number:Ay
(Ay) +	—	—
– (Ay)	—	—
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	110	reg. number:Ay

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d_{16},PC)	111	010
(d_8,PC,Xi)	111	011

LINK

Link and Allocate
First appeared in ISA_A

LINK

Operation: SP – 4 → SP; Ay → (SP); SP → Ay; SP + d_n → SP

Assembler Syntax: LINK.W Ay,#<displacement>

Attributes: Size = Word

Description: Pushes the contents of the specified address register onto the stack. Then loads the updated stack pointer into the address register. Finally, adds the displacement value to the stack pointer. The displacement is the sign-extended word following the operation word. Note that although LINK is a word-sized instruction, most assemblers also support an unsized LINK.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	Register, Ay		
Word Displacement															

Instruction Fields:

- Register field—Specifies the address register, Ay, for the link.
- Displacement field—Specifies the two's complement integer to be added to the stack pointer.

First appeared in ISA_A

Operation: Source → Destination

Assembler Syntax: MOVE.sz <ea>y,<ea>x

Attributes: Size = byte, word, longword

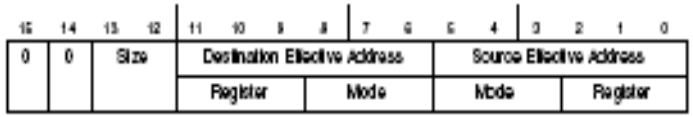
Description: Moves the data at the source to the destination location and sets the condition codes according to the data. The size of the operation may be specified as byte, word, or longword. MOVEA is used when the destination is an address register. MOVEQ is used to move an immediate 8-bit value to a data register. MOV3Q (supported starting with ISA_B) is used to move a 3-bit immediate value to any effective destination address.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

X Not affected
N Set if result is negative; cleared otherwise
Z Set if the result is zero; cleared otherwise
V Always cleared
C Always cleared

Instruction Format:



Instruction fields:

- Size field—Specifies the size of the operand to be moved:
  - — 01 byte operation
  - — 11 word operation
  - — 10 longword operation
- Destination Effective Address field—Specifies destination location, <ea>x; the table below lists possible data alterable addressing modes. The restrictions on combinations of source and destination addressing modes are listed in the table at the bottom of the next page.

Addressing Mode	Mode	Register
Dx	000	reg. number:Dx
Ax	—	—
(Ax)	010	reg. number:Ax
(Ax) +	011	reg. number:Ax
– (Ax)	100	reg. number:Ax
(d_{16},Ax)	101	reg. number:Ax
(d_8,Ax,Xi)	110	reg. number:Ax

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d_{16},PC)	—	—
(d_8,PC,Xi)	—	—

Instruction fields (continued):

- Source Effective Address field—Specifies source operand, <ea>y; the table below lists possible addressing modes. The restrictions on combinations of source and destination addressing modes are listed in the table at the bottom of the next page.

Addressing Mode	Mode	Register
Dy	000	reg. number:Dy
Ay	001	reg. number:Ay
(Ay)	010	reg. number:Ay
(Ay) +	011	reg. number:Ay
– (Ay)	100	reg. number:Ay
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	110	reg. number:Ay

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d_{16},PC)	111	010
(d_8,PC,Xi)	111	011

NOTE

Not all combinations of source/destination addressing modes are possible. The table below shows the possible combinations. Starting with ISA_B, the combination of #<xxx>,d_{16}(Ax) can be used with MOVE.B and MOVE.W opcodes.

Source Addressing Mode	Destination Addressing Mode
Dy, Ay, (Ay), (Ay)+, -(Ay)	All possible
(d_{16}, Ay), (d16, PC)	All possible except (d_8, Ax, Xi), (xxx).W, (xxx).L
(d8, Ay, Xi), (d8, PC, Xi), (xxx).W, (xxx).L, #<xxx>	All possible except (d_{16}, Ax), (d_8, Ax, Xi), (xxx).W, (xxx).L

MOVEA Move Address from Source to Destination MOVEA

First appeared in ISA_A

Operation: Source → Destination

Assembler Syntax: MOVEA.sz <ea>y,Ax

Attributes: Size = word, longword

Description: Moves the address at the source to the destination address register. The size of the operation may be specified as word or longword. Word size source operands are sign extended to 32-bit quantities before the operation is done.

Condition Codes: Not affected

Instruction Format:

15	14	13 12	11 10 9	8	7	6	5 4	3 2 1 0
0	0	Size	Destination Register, Ax	0	0	1	Source Effective Address	
							Mode	Register

Instruction fields:

- Size field—Specifies the size of the operand to be moved:
  - — 0x reserved
  - — 11 word operation
  - — 10 longword operation
- Destination Register field — Specifies the destination address register, Ax.
- Source Effective Address field—Specifies the source operand, <ea>y; the table below lists possible modes.

Addressing Mode	Mode	Register
Dy	000	reg. number:Dy
Ay	001	reg. number:Ay
(Ay)	010	reg. number:Ay
(Ay) +	011	reg. number:Ay
– (Ay)	100	reg. number:Ay
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	110	reg. number:Ay

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d_{16},PC)	111	010
(d_8,PC,Xi)	111	011

First appeared in ISA_A

Operation: Registers → Destination;
Source → Registers

Assembler Syntax: MOVEM.L #list,<ea>x
MOVEM.L <ea>y,#list

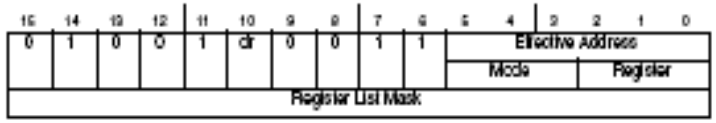
Attributes: Size = longword

Description: Moves the contents of selected registers to or from consecutive memory locations starting at the location specified by the effective address. A register is selected if the bit in the mask field corresponding to that register is set.

The registers are transferred starting at the specified address, and the address is incremented by the operand length (4) following each transfer. The order of the registers is from D0 to D7, then from A0 to A7.

Condition Codes: Not affected

Instruction Format:



Instruction Fields:

- dr field—Specifies the direction of the transfer:
  - — 0 register to memory
  - — 1 memory to register
- Effective Address field—Specifies the memory address for the data transfer. For register-to-memory transfers, use the following table for <ea>x.

Addressing Mode	Mode	Register
Dx	—	—
Ax	—	—
(Ax)	010	reg. number:Ax
(Ax) +	—	—
– (Ax)	—	—
(d_{16},Ax)	101	reg. number:Ax
(d_8,Ax,Xi)	—	—

Addressing Mode	Mode	Register
(xxx).W	—	—
(xxx).L	—	—
#<data>	—	—
(d_{16},PC)	—	—
(d_8,PC,Xi)	—	—

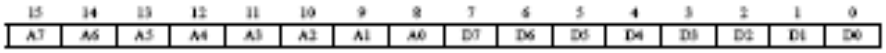
Instruction Fields (continued):

- Effective Address field (continued)—For memory-to-register transfers, use the following table for <ea>y.

Addressing Mode	Mode	Register
Dy	—	—
Ay	—	—
(Ay)	010	reg. number:Ay
(Ay) +	—	—
– (Ay)	—	—
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	—	—

Addressing Mode	Mode	Register
(xxx).W	—	—
(xxx).L	—	—
#<data>	—	—
(d_{16},PC)	—	—
(d_8,PC,Xi)	—	—

- Register List Mask field—Specifies the registers to be transferred. The low-order bit corresponds to the first register to be transferred; the high-order bit corresponds to the last register to be transferred. The mask correspondence is shown below.



MOVEQ

Move Quick
First appeared in ISA_A

MOVEQ

Operation: Immediate Data → Destination

Assembler Syntax: MOVEQ.L #<data>,Dx

Attributes: Size = longword

Description: Moves a byte of immediate data to a 32-bit data register, Dx. The data in an 8-bit field within the operation word is sign-extended to a longword operand in the data register as it is transferred.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

X Not affected
N Set if result is negative; cleared otherwise
Z Set if the result is zero; cleared otherwise
V Always cleared
C Always cleared

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	Register, Dx			0	Immediate Data							

Instruction Fields:

- Register field—Specifies the data register, Dx, to be loaded.
- Data field—8 bits of data, which are sign-extended to a longword operand.

MULU

Unsigned Multiply

First appeared in ISA_A

MULU

Operation: Source * Destination → Destination

Assembler Syntax: MULU.W <ea>y,Dx 16 x 16 → 32
MULU.L <ea>y,Dx 32 x 32 → 32

Attributes: Size = word, longword

Description: Multiplies two unsigned operands yielding an unsigned result. This instruction has a word operand form and a longword operand form.

In the word form, the multiplier and multiplicand are both word operands, and the result is a longword operand. A register operand is the low-order word; the upper word of the register is ignored. All 32 bits of the product are saved in the destination data register.

In the longword form, the multiplier and multiplicand are both longword operands, and the destination data register stores the low order 32 bits of the product. The upper 32 bits of the product are discarded.

Note that CCR[V] is always cleared by MULU, unlike the 68K family processors.

Condition Codes:

X	N	Z	V	C
—	*	*	0	0

X Not affected
N Set if result is negative; cleared otherwise
Z Set if the result is zero; cleared otherwise
V Always cleared
C Always cleared

Instruction Format: (Word)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	Register, Dx			0	1	1	Source Effective Address					
										Mode				Register	

Instruction Fields (Word):

- Register field—Specifies the destination data register, Dx.
- Effective Address field—Specifies the source operand, <ea>y; use only those data addressing modes listed in the following table:

Addressing Mode	Mode	Register
Dy	000	reg. number:Dy
Ay	—	—
(Ay)	010	reg. number:Ay
(Ay) +	011	reg. number:Ay
– (Ay)	100	reg. number:Ay
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	110	reg. number:Ay

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	111	100
(d_{16},PC)	111	010
(d_8,PC,Xi)	111	011

Instruction Format: (Longword)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	Source Effective Address					
										Mode			Register		
0	Register, Dx			0	0	0	0	0	0	0	0	0	0	0	0

Instruction Fields (Longword):

- Source Effective Address field—Specifies the source operand; use only data addressing modes listed in the following table:

Addressing Mode	Mode	Register
Dy	000	reg. number:Dy
Ay	—	—
(Ay)	010	reg. number:Ay
(Ay) +	011	reg. number:Ay
– (Ay)	100	reg. number:Ay
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	—	—

Addressing Mode	Mode	Register
(xxx).W	—	—
(xxx).L	—	—
#<data>	—	—
(d_{16},PC)	—	—
(d_8,PC,Xi)	—	—

- Register field—Specifies a data register, Dx, for the destination operand; the 32-bit multiplicand comes from this register, and the low-order 32 bits of the product are loaded into this register.

PEA

Push Effective Address
First appeared in ISA_A

PEA

Operation: SP – 4 → SP; <ea>y → (SP)

Assembler Syntax: PEA.L <ea>y

Attributes: Size = longword

Description: Computes the effective address and pushes it onto the stack. The effective address is a longword address.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	Source Effective Address					
										Mode		Register			

Instruction Field:

- Effective Address field—Specifies the address, <ea>y, to be pushed onto the stack; use only those control addressing modes listed in the following table:

Addressing Mode	Mode	Register
Dy	—	—
Ay	—	—
(Ay)	010	reg. number:Ay
(Ay) +	—	—
– (Ay)	—	—
(d_{16},Ay)	101	reg. number:Ay
(d_8,Ay,Xi)	110	reg. number:Ay

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
#<data>	—	—
(d_{16},PC)	111	010
(d_8,PC,Xi)	111	011

RTS

Return from Subroutine
First appeared in ISA_A

RTS

Operation: (SP) → PC; SP + 4 → SP

Assembler Syntax: RTS

Attributes: Unsized

Description: Pulls the program counter value from the stack. The previous program counter value is lost. The opcode for RTS is 0x4E75.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

SUBQ — Subtract Quick — SUBQ

First appeared in ISA_A

Operation: Destination - Immediate Data → Destination

Assembler Syntax: SUBQ.L #<data>,<ea>x

Attributes: Size = longword

Description: Operates similarly to SUB, but is used when the source operand is immediate data ranging in value from 1 to 8. Subtracts the immediate value from the operand at the destination location. The size of the operation is specified as longword. The immediate data is zero-filled to a longword before being subtracted from the destination. When adding to address registers, the condition codes are not altered.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

- X Set the same as the carry bit
- N Set if the result is negative; cleared otherwise
- Z Set if the result is zero; cleared otherwise
- V Set if an overflow is generated; cleared otherwise
- C Set if an carry is generated; cleared otherwise

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	Data			1	1	0	Destination Effective Address					
										Mode			Register		

Instruction Fields:

- Data field—3 bits of immediate data representing 8 values (0 – 7), with the immediate values 1-7 representing values of 1-7 respectively and 0 representing a value of 8.
- Destination Effective Address field—specifies the destination location; use only those alterable addressing modes listed in the following table:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dx	000	reg. number:Dx	(xxx).W	111	000
Ax	001	reg. number:Ax	(xxx).L	111	001
(Ax)	010	reg. number:Ax	#<data>	—	—
(Ax) +	011	reg. number:Ax			
– (Ax)	100	reg. number:Ax			
(d_{16},Ax)	101	reg. number:Ax	(d_{16},PC)	—	—
(d_8,Ax,Xi)	110	reg. number:Ax	(d_8,PC,Xi)	—	—

TRAP — Trap — TRAP

First appeared in ISA_A

Operation: 1 → S-Bit of SR
SP – 4 → SP; nextPC → (SP); SP – 2 → SP;
SR → (SP); SP – 2 → SP; Format/Offset → (SP);
(VBR + 0x80 + 4*n) → PC
where n is the TRAP vector number

Assembler Syntax: TRAP #<vector>

Attributes: Unsized

Description: Causes a TRAP #<vector> exception. The TRAP vector field is multiplied by 4 and then added to 0x80 to form the exception address. The exception address is then added to the VBR to index into the exception vector table. The vector field value can be 0 – 15, providing 16 vectors.

Note when SR is copied onto the exception stack frame, it represents the value at the beginning of the TRAP instruction's execution. At the conclusion of the exception processing, the SR is updated to clear the T bit and set the S bit.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	0	Vector			

Instruction Fields:

- Vector field—Specifies the trap vector to be taken.

UNLK

Unlink

First appeared in ISA_A

UNLK

Operation: Ax → SP; (SP) → Ax; SP + 4 → SP

Assembler Syntax: UNLK Ax

Attributes: Unsized

Description: Loads the stack pointer from the specified address register, then loads the address register with the longword pulled from the top of the stack.

Condition Codes: Not affected

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	Register, Ax		

Instruction Field:

- Register field—Specifies the address register, Ax, for the instruction.