Parte I: Apéndice C
Juego de instrucciones del M68000

Este apéndice proporciona información detallada sobre el uso de la mayor parte de las instrucciones disponibles en el M68000. Para más información, el lector deberá referirse al *M68000 User's Manual* (Manual de usuario del M68000), editado por Motorola. La información suministrada aquí sobre cada instrucción es la siguiente:

- su sintaxis, incluyendo los modos posibles de direccionamiento y el tamaño de los operandos
- una descripción de la operación que ejecuta
- la forma en que se ven afectados los códigos de condición.

El siguiente ejemplo muestra la información disponible sobre cada instrucción.

ADD **Suma**

Sintaxis: ADD.S a_1, D_i o ADD.S D_i, a_3
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_1) + (D_i) \rightarrow D_i$ o $(D_i) + (a_3) \rightarrow a_3$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

La sintaxis especifica el nombre de la instrucción y los modos de direccionamiento admisibles. Dichos modos pueden definirse mediante el nombre de un registro (D_i o A_i por ejemplo), o mediante un conjunto de modos de direccionamiento (por ejemplo a_1) que se aplican a los operandos origen o destino de la instrucción. La operación se describe verbalmente o utilizando

notación simbólica. Finalmente se indica, mediante el empleo de símbolos especiales, la forma en que la instrucción afecta a los códigos de condición. A continuación se muestran las abreviaturas utilizadas en la descripción de las instrucciones.

La Tabla C.1 define abreviaturas para todos los modos de direccionamiento empleados en este texto. Dado que a un operando específico de una instrucción sólo puede aplicársele un subconjunto de todos los modos de direccionamiento disponibles, se definen nueve subconjuntos de direccionamiento (a_1, a_2, ...,a_9) en la Tabla C.2, utilizando las abreviaturas definidas en la Tabla C.1 . La colocación de la "X" en una cierta posición indica que el modo de direccionamiento correspondiente puede utilizarse. Así por ejemplo, a_1 significa que pueden utilizarse todos los modos de direccionamiento para especificar un operando, mientras que a_4 significa que todos los modos de direccionamiento están disponibles con la excepción de ARD (direccionamiento mediante registro de direcciones).

Abrev.	Modo de direccionamiento	Ejemplo
IMM	Inmediato	MOVE.B #1,D0
ABS	Absoluto	MOVE.B 1,D0
DRD	Mediante registro	MOVE.B D1,D0
ARD	Mediante registro de direcciones	MOVE.B A1,D0
IND	Relativo a registro	MOVE.B (A1),D0
IDI	Relativo a registro con posincremento	MOVE.B (A1)+,D0
IDD	Relativo a registro con predecremento	MOVE.B -(A1),D0
AID	Relativo a registro con desplazamiento	MOVE.B 5(A0),D0
AII	Relativo a registro con índice	MOVE.B 5(A0,D1),D0
PID	Relativo a contador de programa con desplazamiento	MOVE.B 5(PC),D0
PII	Relativo a contador de programa con índice	MOVE.B 5(PC,D1),D0

Tabla C.1 Modos de direccionamiento del M68000.

	IMM	ABS	DRD	ARD	IND	IDI	IDD	AID	AII	PID	PII
a_1	X	X	X	X	X	X	X	X	X	X	X
a_2		X	X		X	X	X	X	X		
a_3		X			X	X	X	X	X		
a_4	X	X	X		X	X	X	X	X	X	X
a_5		X			X		X	X	X		
a_6		X			X	X		X	X	X	X
a_7		X			X			X	X	X	X
a_8		X	X	X	X	X	X	X	X		
a_9		X	X		X	X	X	X	X	X	X

Tabla C.2 Subconjuntos de modos de direccionamiento utilizados en las instrucciones del M68000.

Los símbolos mostrados en la Tabla C.3 sirven para especificar cómo afectan las distintas instrucciones a los códigos de condición. Se entiende por comportamiento normal aquél en el que los indicadores toman valores acordes con su significado, y se representa con un asterisco "*". La Tabla C.4 precisa exactamente el significado de dicho comportamiento normal. Existen algunas instrucciones en las que algún indicador puede tomar sus valores de acuerdo con ciertas reglas especiales. Este caso se representa mediante una exclamación "!", y la regla especial se incluye en la descripción de la instrucción.

Descripción	Notación
Caso normal	*
No le afecta	-
Se pone a 1	1
Se pone a 0	0
Indefinido	I
Significado especial	!

Tabla C.3 Notación utilizada para expresar el modo en que la instrucción afecta al indicador.

Indicador	Condición
X	Se pone a 1 cuando se pone el C. En otro caso se pone a 0.
N	Se pone a 1 cuando el resultado es negativo. En otro caso se pone a 0.
Z	Se pone a 1 cuando el resultado de una operación es 0. En otro caso se pone a 0.
V	Se pone a 1 cuando hay sobrepasamiento en una operación en complemento a 2. En otro caso se pone a 0.
C	Se ponen a 1 cuando se genera un acarreo o un débito. En otro caso se pone a 0.

Tabla C.4 Valores que toman los códigos de condición del M68000 en el caso normal.

ABCD — Suma en código BCD con extensión

Sintaxis: ABCD D_i, D_j o ABCD $-(A_i)$, $-(A_j)$
Tamaño de los operandos: Byte
Operación:
$(D_i) + (D_j) + (X) \rightarrow D_j$ o
$(A_i) - 1 \rightarrow A_i$; $(A_j) - 1 \rightarrow A_j$; $((A_i)) + ((A_j)) + (X) \rightarrow (A_j)$
Códigos de condición:

X	N	Z	V	C
*	I	*	I	*

Observación: Aquí el signo + indica suma en BCD.

ADD — Suma

Sintaxis: ADD.S a_1, D_i o ADD.S D_i, a_3
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_1) + (D_i) \rightarrow D_i$ o $(D_i) + (a_3) \rightarrow a_3$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

ADDA — Suma dirección

Sintaxis: ADDA.S a_1, A_i
Tamaño de los operandos: S = (W, L)
Operación: $(a_1) + (A_i) \rightarrow A_i$
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

ADDI — Suma inmediata

Sintaxis: ADDI.S #n, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $n + (a_2) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

ADDQ — Suma rápida

Sintaxis: ADDQ.S #n, a_8
Tamaño de los operandos: S = (B, W, L)
Operación: $n + (a_8) \rightarrow a_8$, donde $1 \leq n \leq 8$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Esta instrucción ocupa sólo una palabra.

ADDX — Suma extendida

Sintaxis: ADDX.S D_i, D_j o ADDX.S $-(A_i)$, $-(A_j)$
Tamaño de los operandos: S = (B, W, L)
Operación:
$(D_i) + (D_j) + (X) \rightarrow D_j$ o
$(A_i) - k \rightarrow A_i$; $(A_j) - k \rightarrow A_j$; $((A_i)) + ((A_j)) + (X) \rightarrow (A_j)$
donde k depende de S (1, 2 o 4).
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

AND — Y lógica

Sintaxis: AND.S a_4, D_i o AND.S D_i, a_3
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_4) \wedge (D_i) \rightarrow D_i$ o $(D_i) \wedge (a_3) \rightarrow a_3$
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

ANDI — Y lógica inmediata

Sintaxis: ANDI.S #n, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $n \wedge (a_2) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

ANDI a CCR — Y lógica inmediata al código de condición

Sintaxis: ANDI #n, CCR
Tamaño de los operandos: Byte
Operación: n ∧ (CCR) → CCR
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

ANDI a SR — Y lógica inmediata al registro de estado

Sintaxis: ANDI #n, SR
Tamaño de los operandos: Palabra
Operación: n ∧ (SR) → SR
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Se trata de una instrucción privilegiada.

ASL — Desplazamiento aritmético a izquierda

Sintaxis: ASL.S D_i, D_j o ASL.S #n, D_j o ASL a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo tamaño palabra.
Operación: Desplaza a la izquierda los bits del operando destino. El número de desplazamientos unitarios viene indicado por el operando origen. Si éste es un registro de datos, la cuenta de desplazamiento vale (D_i) mod 64. Si es una constante, la cuenta vale n = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Se introducen ceros en los bits menos significativos.
Códigos de condición:

X	N	Z	V	C
*	*	*	!	!

V Se pone a 1 si el bit más significativo cambia en algún momento durante la operación de rotación.
C Es el valor del último bit desplazado fuera del operando destino.

ASR — Desplazamiento aritmético a derecha

Sintaxis: ASR.S D_i, D_j o ASR.S #n, D_j o ASR a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo tamaño palabra.
Operación: Desplaza a la derecha los bits del operando destino. El número de desplazamientos unitarios viene indicado por el operando origen. Si éste es un registro de datos, la cuenta de desplazamiento vale (D_i) mod 64. Si es una constante, la cuenta vale n = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Se introducen ceros en los bits más significativos.
Códigos de condición:

X	N	Z	V	C
*	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

Bcc — Ramificación condicional

Sintaxis: Bcc *etiqueta*
Operación: Si se cumple la condición *cc*, entonces *etiqueta* → PC. La condición *cc* especifica una de las condiciones siguientes:

CC	(C)'
CS	(C)
NE	(Z)'
EQ	(Z)
VC	(V)'
VS	(V)
PL	(N)'
MI	(N)
LS	(C) ∨ (Z)
HI	(C)' ∧ (Z)'
LT	((N) ∧ (V)') ∨ ((N)' ∧ (V))
LE	(Z) ∨ (((N) ∧ (V)') ∨ ((N)' ∧ (V)))
GT	(Z)' ∧ (((N) ∧ (V)) ∨ ((N)' ∧ (V)'))
GE	((N) ∧ (V)) ∨ ((N)' ∧ (V)')

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

BCHG — Comprobar un bit e invertirlo

Sintaxis: BCHG D_i, a_2 o BCHG #n, a_2

Tamaño de los operandos: Si el operando destino es un registro de datos, la longitud del operando es una palabra larga; en cualquier otro caso la longitud es un byte.

Operación: Comprueba el bit del operando destino cuya posición viene indicada por el operando origen, y pone el indicador Z de acuerdo con el resultado de dicha comprobación. Luego se invierte el bit comprobado.

Códigos de condición:

X	N	Z	V	C
-	-	!	-	-

Z cambia según lo indicado antes.

BCLR — Comprobar un bit y ponerlo a 0

Sintaxis: BCLR D_i, a_2 o BCLR #n, a_2

Tamaño de los operandos: Si el operando destino es un registro de datos, la longitud del operando es una palabra larga; en cualquier otro caso la longitud es un byte.

Operación: Comprueba el bit del operando destino cuya posición viene indicada por el operando origen y pone el indicador Z de acuerdo con el resultado de dicha comprobación. Luego pone a cero el bit comprobado.

Códigos de condición:

X	N	Z	V	C
-	-	!	-	-

Z cambia según lo indicado antes.

BRA — Ramificación incondicional

Sintaxis: BRA *etiqueta*

Operación: *etiqueta* → PC

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

BSET — Comprobar un bit y ponerlo a 1

Sintaxis: BSET D_i, a_2 o BSET #n, a_2

Tamaño de los operandos: Si el operando destino es un registro de datos, la longitud del operando es una palabra larga; en cualquier otro caso la longitud es un byte.

Operación: Comprueba el bit del operando destino cuya posición viene indicada por el operando origen y pone el indicador Z de acuerdo con el resultado de dicha comprobación. Luego pone a uno el bit comprobado.

Códigos de condición:

X	N	Z	V	C
-	-	!	-	-

Z cambia según lo indicado antes.

BSR — Ramificación a subrutina

Sintaxis: BSR *etiqueta*

Operación: (SP) - 4 → SP; (PC) → (SP); *etiqueta* → PC

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

BTST — Comprueba un bit

Sintaxis: BTST D_i, a_2 o BTST #n, a_9

Tamaño de los operandos: Si el operando destino es un registro de datos, la longitud del operando es una palabra larga; en cualquier otro caso la longitud es un byte.

Operación: Comprueba el bit del operando destino cuya posición viene indicada por el operando origen y pone el indicador Z de acuerdo con el resultado de dicha comprobación.

Códigos de condición:

X	N	Z	V	C
-	-	!	-	-

Z cambia según lo indicado antes.

CHK — Compara un registro con unos límites

Sintaxis: CHK a_4, D_i
Tamaño de los operandos: Palabra
Operación: Si $(D_i) < 0$ ó $(D_i) > (a_4)$, sucede la trampa nº6 (número de orden en el vector de excepción).

Códigos de condición:

X	N	Z	V	C
-	!	I	I	I

N Se pone a 1 si $(D_i) < 0$; se pone a 0 si $(D_i) > (a_4)$. Indefinido en cualquier otro caso.

CLR — Poner a cero un operando

Sintaxis: CLR.S a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $0 \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
-	0	1	0	0

CMP — Comparación

Sintaxis: CMP.S a_1, D_i
Tamaño de los operandos: S = (B, W, L)
Operación: $(D_i) - (a_1)$
Códigos de condición:

X	N	Z	V	C
-	*	*	*	*

CMPA — Comparación de direcciones

Sintaxis: CMPA.S a_1, A_i
Tamaño de los operandos: S = (W, L)
Operación: $(A_i) - (a_1)$
Códigos de condición:

X	N	Z	V	C
-	*	*	*	*

CMPI — Comparación inmediata

Sintaxis: CMPI.S #n, a_9
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_9) - n$
Códigos de condición:

X	N	Z	V	C
-	*	*	*	*

CMPM — Comparación de posiciones de memoria

Sintaxis: CMPM.S $(A_i)+$, $(A_j)+$
Tamaño de los operandos: S = (B, W, L)
Operación: $((A_i)) - ((A_j))$; $(A_i)+k \rightarrow A_i$; $(A_j)+k \rightarrow A_j$
donde k depende del tamaño del operando (1, 2 o 4).

Códigos de condición:

X	N	Z	V	C
-	*	*	*	*

DBcc — Test de condición, decrementa y ramifica

Sintaxis: DB*cc* D_i, *etiqueta*
Tamaño de los operandos: Palabra
Operación: Si se cumple la condición *cc*, no hace nada. Si no se cumple: $(D_i) - 1 \rightarrow D_i$; si $(D_i) \neq -1$ entonces *etiqueta* $\rightarrow$ PC.
La condición *cc* es cualquiera de las indicadas en la instrucción B*cc*, y además puede ser:

F siempre FALSO
T siempre VERDADERO

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

DIVS — División con signo

Sintaxis: DIVS.W a_i, D_i
Tamaño de los operandos: Palabra
Operación: Divide el operando destino con signo (32 bits) por el operando origen con signo (16 bits). El resultado es un cociente con signo que se guarda en los 16 bits menos significativos, y un resto que se guarda en los 16 bits más significativos. El signo del resto es el del dividendo. Se genera una trampa si el divisor es cero.

Códigos de condición:

X	N	Z	V	C
-	!	!	!	0

N y Z siguen sus casos normales pero se vuelven indefinidos si hay sobrepasamiento.
V Es el caso normal pero se vuelve indefinido si hay división por cero.

DIVU — División sin signo

Sintaxis: DIVU.W a_i, D_i
Tamaño de los operandos: Palabra
Operación: Divide el operando destino sin signo (32 bits) por el operando ori-gen sin signo (16 bits). El resultado es un cociente sin signo que se guarda en los 16 bits menos significativos, y un resto que se guarda en los 16 bits más significativos. Se genera una trampa si el divisor es cero.

Códigos de condición:

X	N	Z	V	C
-	!	!	!	0

N y Z siguen sus casos normales pero se vuelven indefinidos si hay sobrepasamiento.
V Es el caso normal pero se vuelve indefinido si hay división por cero.

EOR — O exclusivo

Sintaxis: EOR.S D_i, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $(D_i) \oplus (a_2) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

EORI — O exclusivo inmediato

Sintaxis: EORI.S #n, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $n \oplus (a_2) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

EORI a CCR — O exclusivo inmediato a códigos de condición

Sintaxis: EORI.S #n, CCR
Tamaño de los operandos: Byte
Operación: $n \oplus (CCR) \rightarrow CCR$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

EORI a SR — O exclusivo inmediato al registro de estado

Sintaxis: EORI.S #n, SR
Tamaño de los operandos: Palabra
Operación: $n \oplus (SR) \rightarrow SR$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Instrucción privilegiada.

EXG — Intercambio de registros

Sintaxis: EXG D_i, D_j o EXG A_i, A_j o EXG D_i, A_j o EXG A_i, D_j
Tamaño de los operandos: Palabra larga
Operación: Intercambia los contenidos de los operandos origen y destino.

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

EXT — Extensión de signo

Sintaxis: EXT.S D_i
Tamaño de los operandos: S = (W, L)
Operación: Extiende el bit de signo del operando. Si el tamaño del operando es una palabra, el signo de los 8 bits menos significativos se extiende a una palabra. Si el tamaño del operando es una palabra larga, el signo de los 16 bits menos significativos se extiende a una palabra larga.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

JMP — Salto

Sintaxis: JMP a_7
Operación: $a_7 \rightarrow$ PC
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

JSR — Salto a una subrutina

Sintaxis: JSR a_7
Operación: (SP) - 4 → SP; (PC) → (SP); $a_7 \rightarrow$ PC
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

LEA — Carga una dirección efectiva

Sintaxis: LEA a_7, A_i
Tamaño de los operandos: Palabra larga
Operación: $a_7 \rightarrow A_i$
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

LSL — Desplazamiento lógico a izquierda

Sintaxis: LSL.S D_i, D_j o LSL.S #n, D_j o LSL a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo formato palabra.
Operación: Desplaza a la izquierda los bits del operando destino. El número de desplazamientos unitarios viene indicado por el operando origen. Si éste es un registro de datos, la cuenta de desplazamiento vale (D_i) mod 64. Si es una constante, la cuenta vale n = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Se introducen ceros en los bits menos significativos.
Códigos de condición:

X	N	Z	V	C
*	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

LSR — Desplazamiento lógico a derecha

Sintaxis: LSR.S D_i, D_j o LSR.S #n, D_j o LSR a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo formato palabra.
Operación: Desplaza a la derecha los bits del operando destino. El número de desplazamientos unitarios viene indicado por el operando origen. Si éste es un registro de datos, la cuenta de desplazamiento vale (D_i) mod 64. Si es una constante, la cuenta vale n = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Se introducen ceros en los bits más significativos.
Códigos de condición:

X	N	Z	V	C
*	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

MOVE — Transferir datos de origen a destino

Sintaxis: MOVE.S a_1, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: (a_1) → a_2
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

MOVEA — Transferir una dirección

Sintaxis: MOVEA.S a_1, A_i
Tamaño de los operandos: S = (W, L)
Operación: $(a_1) \rightarrow A_i$
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

MOVE a CCR — Transferir al registro de códigos de condición

Sintaxis: MOVE a_4, CCR
Tamaño de los operandos: Palabra
Operación: $(a_4) \rightarrow$ CCR
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

MOVE a SR — Transferir al registro de estado

Sintaxis: MOVE a_4, SR
Tamaño de los operandos: Palabra
Operación: $(a_4) \rightarrow$ SR
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Instrucción privilegiada.

MOVE el USP — Transferir el puntero de la pila de usuario

Sintaxis: MOVE A_i, USP o MOVE USP, A_i
Tamaño de los operandos: Palabra larga
Operación: $(A_i) \rightarrow$ USP o (USP) $\rightarrow A_i$
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

Observación: Instrucción privilegiada.

MOVEM — Transferencia múltiple de registros

Sintaxis: MOVEM.S *lista de registros*, a_5 o
MOVEM.S a_6, *lista de registros*
Tamaño de los operandos: S = (W, L)
Operación: Transfiere el contenido de los registros seleccionados a (en el caso de la primera forma) o desde (caso de la segunda forma) posiciones consecutivas de memoria.
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

MOVEQ — Transferencia rápida

Sintaxis: MOVEQ #n, D_i
Tamaño de los operandos: Palabra larga
Operación: n $\rightarrow D_i$, donde *n* es un número de 8 bits en complemento a 2 cuyo signo se extiende en el destino.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

Observación: Esta instrucción ocupa sólo una palabra.

MULS — Multiplicación con signo

Sintaxis: MULS.W a_4, D_i
Tamaño de los operandos: Palabra
Operación: Multiplica dos operandos con signo de 16 bits, dando lugar a un resultado con signo de 32 bits.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

MULU — Multiplicación sin signo

Sintaxis: MULU.W a_4, D_i
Tamaño de los operandos: Palabra
Operación: Multiplica dos operandos sin signo de 16 bits, dando lugar a un resultado sin signo de 32 bits.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

NBCD — Negación en BCD con extensión

Sintaxis: NBCD a_2
Tamaño de los operandos: Byte
Operación: $0 - (a_2) - (X) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
*	I	!	I	*

Z Se pone a cero si el resultado es distinto de cero. En caso contrario se queda como estaba.

Observación: Aquí el signo - indica resta en BCD.

NEG — Negación

Sintaxis: NEG.S a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $0 - (a_2) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

NEGX — Negación con extensión

Sintaxis: NEGX.S a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $0 - (a_2) - (X) \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
*	*	!	*	*

Z Se pone a cero si el resultado es distinto de cero. En caso contrario se queda como estaba.

NOP — No operación

Sintaxis: NOP
Operación: No realiza ninguna operación.
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

NOT — Inversión lógica

Sintaxis: NOT.S a_2
Tamaño de los operandos: S = (B, W, L)
Operación: Invierte todos los bits del operando.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

OR — O lógica

Sintaxis: OR.S a_4, D_i o OR.S D_i, a_3
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_4) \vee (D_i) \rightarrow D_i$ o $(D_i) \vee (a_3) \rightarrow a_3$
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

ORI — O lógica inmediata

Sintaxis: ORI.S #n, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: n ∨ (a_2) → a_2
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

ORI a CCR — O lógica inmediata al código de condición

Sintaxis: ORI #n, CCR
Tamaño de los operandos: Byte
Operación: n ∨ (CCR) → CCR
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

ORI a SR — O lógica inmediata al registro de estado

Sintaxis: ORI #n, SR
Tamaño de los operandos: Palabra
Operación: n ∨ (SR) → SR
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Se trata de una instrucción privilegiada.

PEA — Guarda en pila una dirección efectiva

Sintaxis: PEA a_7
Tamaño de los operandos: Palabra larga
Operación: (SP) - 4 → SP; a_7 → (SP)
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

ROL — Rotación a izquierda

Sintaxis: ROL.S D_i, D_j o ROL.S #n, D_j o ROL a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo formato palabra.
Operación: Rota a la izquierda los bits del operando destino. El número de rotaciones unitarias viene indicado por el operando origen. Si éste es un registro de datos, la cuenta de rotación vale (D_i) mod 64. Si es una constante, la cuenta vale n = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Los bits más significativos que se desplazan fuera del operando, se introducen por el bit menos significativo del operando.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

ROR — Rotación a derecha

Sintaxis: ROR.S D_i, D_j o ROR.S #n, D_j o ROR a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo formato palabra.
Operación: Rota a la derecha los bits del operando destino. El operando origen indica el número de rotaciones unitarias: si es un registro de datos, la cuenta de rotación vale (D_i) mod 64; si es una constante, la cuenta vale n = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Los bits menos significativos desplazados fuera del operando se introducen por el bit más significativo del operando.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

ROXL — Rotación a izquierda con extensión

Sintaxis: ROXL.S D_i, D_j o ROXL.S #n, D_j o ROXL a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo formato palabra.
Operación: Rota a la izquierda los bits del operando destino. El operando origen indica el número de rotaciones unitarias: si es un registro de datos, la cuenta de rotación vale (D_i) mod 64; si es una constante, la cuenta vale *n* = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Los bits más significativos desplazados fuera del operando, se introducen en el indicador X, y éste se introduce a su vez en el bit menos significativo del operando.

Códigos de condición:

X	N	Z	V	C
*	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

ROXR — Rotación a derecha con extensión

Sintaxis: ROXR.S D_i, D_j o ROXR.S #n, D_j o ROXR a_3
Tamaño de los operandos: S = (B, W, L). En el tercer caso, solo formato palabra.
Operación: Rota a la derecha los bits del operando destino. El operando origen indica el número de rotaciones unitarias: si es un registro de datos, la cuenta de rotación vale (D_i) mod 64; si es una constante, la cuenta vale *n* = [1,8]; y si el destino es una palabra de memoria, la cuenta vale 1. Los bits menos significativos desplazados fuera del operando, se introducen en el indicador X, y éste se introduce a su vez en el bit más significativo del operando.

Códigos de condición:

X	N	Z	V	C
*	*	*	0	!

C Es el valor del último bit desplazado fuera del operando destino.

RTE — Retorna de una excepción

Sintaxis: RTE
Operación: ((SP)) → SR; (SP) + 2 → SP; ((SP)) → PC; (SP) + 4 → SP
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Se trata de una instrucción privilegiada.

RTR — Retorna y restaura los códigos de condición

Sintaxis: RTR
Operación: ((SP)) → CCR; (SP) + 2 → SP; ((SP)) → PC; (SP) + 4 → SP
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

RTS — Retorna de una subrutina

Sintaxis: RTS
Operación: ((SP)) → PC; (SP) + 4 → SP
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

SBCD — Resta en código BCD con extensión

Sintaxis: SBCD D_i, D_j o SBCD -(A_i), -(A_j)
Tamaño de los operandos: Byte
Operación: (D_j) - (D_i) - (X) → D_j o
(A_j) - 1 → A_i ; (A_j) - 1 → A_j ; ((A_j)) - ((A_i)) - (X) → (A_j)
Códigos de condición:

X	N	Z	V	C
*	I	*	I	*

Observación: El signo - de la primera y segunda línea de la descripción de la operación indica resta en BCD.

Scc — Se pone según condición

Sintaxis: Scc a_2
Tamaño de los operandos: Byte
Operación: La condición *cc* es cualquiera de las de la instrucción DB*cc*: si se cumple, entonces 11111111 (binario) → a_2; si no se cumple, 0 → a_2.
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

STOP — Carga el registro de estado y para

Sintaxis: STOP #n
Operación: n → SR y para la ejecución. La ejecución se puede reanudar mediante una excepción.
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

SUB — Resta

Sintaxis: SUB.S a_1, D_i o SUB.S D_i, a_3
Tamaño de los operandos: S = (B, W, L)
Operación: $(D_i) - (a_1) \rightarrow D_i$ o $(a_3) - (D_i) \rightarrow a_3$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

SUBA — Resta dirección

Sintaxis: SUBA.S a_1, A_i
Tamaño de los operandos: S = (W, L)
Operación: $(A_i) - (a_1) \rightarrow A_i$
Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

SUBI — Resta inmediata

Sintaxis: SUBI.S #n, a_2
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_2) - n \rightarrow a_2$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

SUBQ — Resta rápida

Sintaxis: SUBQ.S #n, a_8
Tamaño de los operandos: S = (B, W, L)
Operación: $(a_8) - n \rightarrow a_8$, donde $1 \leq n \leq 8$
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

Observación: Esta instrucción ocupa sólo una palabra.

SUBX — Resta extendida

Sintaxis: SUBX.S D_i, D_j o SUBX.S $-(A_i)$, $-(A_j)$
Tamaño de los operandos: S = (B, W, L)
Operación: $(D_j) - (D_i) - (X) \rightarrow D_j$ o
$(A_i) - k \rightarrow A_i$; $(A_j) - k \rightarrow A_j$; $((A_j)) - ((A_i)) - (X) \rightarrow (A_j)$
donde k depende de S (1, 2 o 4).
Códigos de condición:

X	N	Z	V	C
*	*	*	*	*

SWAP — Intercambia las dos mitades de un registro

Sintaxis: SWAP D_i
Tamaño de los operandos: Palabra
Operación: Intercambia el contenido de los 16 bits más significativos con el de los 16 menos significativos.
Códigos de condición:

X	N	Z	V	C
-	*	*	0	0

TRAP — Trampa

Sintaxis: TRAP #n

Operación: Genera una excepción tipo trampa con el número de orden *n* en el vector de excepción, donde *n* = [0,15].

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

TRAPV — Trampa por sobrepasamiento

Sintaxis: TRAPV

Operación: Si (V) = 1 genera una excepción tipo trampa con el número de orden 7 en el vector de excepción.

Códigos de condición:

X	N	Z	V	C
-	-	-	-	-

TST — Comprueba un operando

Sintaxis: TST.S a_9

Tamaño de los operandos: S = (B, W, L)

Operación: (a_9) - 0

Códigos de condición:

X	N	Z	V	C
-	*	*	0	0