INSTRUCCIONES DE MOVIMIENTO DE DATOS

INSTRUCCIÓN	SINTAXIS	TAMAÑO(t)	Privilegiada	XNZVC	COMENTARIO
MOVE	MOVE.t <ea>,<adea>	B,W,L	no	-* * 00	Mueve el contenido del operando fuente a la posición destino
MOVE to CCR	MOVE <dea>,CCR	W	no	*****	Mueve el contenido del operando fuente sobre los códigos de condición del registro de estado. Aunque el operando fuente sea de tamaño palabra, para actualizar los códigos de condición, sólo se usa el byte menos significativo, el más significativo se ignora.
MOVE from SR	MOVE SR,<adea>	W	no	- - - - -	Mueve el contenido del registro de estado (SR) a la posición destino
MOVE to SR	MOVE <dea>,SR	W	si	*****	Mueve el contenido del operando fuente sobre el registro de estado SR. Esta instrucción debe ejecutarse en modo supervisor, S=1, en caso contrario provoca una excepción
MOVE to USP	MOVE USP,An MOVE An,USP	W	si	- - - - -	El contenido del puntero de pila del usuario se transfiere a, o desde, el registro de direcciones especificado. Esta instrucción debe ejecutarse en modo supervisor, S=1, en caso contrario provoca una excepción
MOVEA	MOVEA.t <ea>,An	W,L	no	- - - - -	Mueve el contenido de la fuente al registro de direcciones especificado. Si la operación se realiza con tamaño palabra, el operando fuente se extiende en signo a 32 bits antes de realizar la escritura en el registro An
MOVEQ	MOVEQ #<d8>,Dn	L	no	- * * 00	Mueve un dato inmediato de 8 bits a un registro de datos. El dato está contenido en un campo de 8 bits dentro de la palabra de operación, por tanto, su rango de valores va desde –128 a 127. El dato se extiende en signo a 32 bits antes de ser transferido al registro de datos.
LEA	LEA <cea>,An	L	No	- - - - -	La dirección efectiva del operando fuente se carga en el registro de direcciones seleccionado.
EXG	EXG Dn,Dm EXG An,Am EXG Dn,Am	L	No	- - - - -	Intercambia el contenido de dos registros. Existen tres modos de intercambio: a) entre dos registros de datos; b) entre dos registros de direcciones y c) entre uno de datos y uno de direcciones.
SWAP	SWAP Dn	W	no	- * * 00	Intercambia las dos mitades de 16 bits de un registro de datos. Dn[32:16]←→Dn[15:0]LI
LINK	LINK An, #desp	L	no	- - - - -	Guarda el registro An en pila, se copia el puntero de pila en An e incrementa el puntero de pila con desp. La cte. despl. Debe ser siempre negativa. Este instrucción reserva espacio en la pila para el paso de argumentos en las llamandas a subrutinas.
UNLK	UNLK An	L	no	- - - - -	Copia An en el puntero de pila, extrae la dirección almacenada en la pila y la restaura en An. Junto con LINK esta instrucción permite liberar el espacio almacenado en la pila.

NOTAS:

(1) Los flags de V,C son siempre cero para las instrucciones de movimiento de datos, salvo para aquellas que tienen como destino el propio CCR (MOVE to CCR o MOVE to SR)en cuyo caso los valores de estos flags depende del dato que movamos.

(2) Z se pone a 1 si el resultado almacenado en el operando destino es 0 (salvo para MOVE to CCR y MOVE to SR)

(3) N se pone a 1 si el bit más significativo del operando destino es 1, y 0 en caso contrario (salvo para MOVE to CCR y MOVE to SR)

(4) X no cambia de valor para las instrucciónes de movimiento de datos

INSTRUCCIONES DE ARITMÉTICA ENTERA

INSTRUCCIÓN	SINTAXIS	TAMAÑO(t)	Privilegiada	XNZVC	COMENTARIO
ADD	ADD.t <ea>,Dn ADD.t Dn,<amea>	B,W,L	No	*****	Suma binaria: Dn+<ea>→Dn o <amea>+Dn→<amea>. Suma el operando fuente con el destino y almacena el resultado en el operando destino
ADDA	ADDA.t <ea>,An	W,L	No	- - - - -	Suma binaria: suma el operando fuente con el contenido del registro An y almacena el resultado en el registro An. Si la instrucción es de tamaño palabra, el operando fuente se extiende en signo a 32 bits.
ADDI	ADDI.t #<dato>,<adea>	B,W,L	No	*****	Suma binaria del dato inmediato con el operando destino, almacenando el resultado en este último. A diferencia de ADD, que permite, también, un dato inmediato para el operando fuente y en la que el destino debe ser siempre un registro de datos, esta instrucción dispone de una gran variedad de modos de direccionamiento para el operando destino.
ADDQ	ADDQ.t #<d3>,<aea>	B,W,L	No	*****	Suma binaria. En este caso, el dato inmediato es un número comprendido entre 1 y 8 que se codifica en la propia instrucción. Admite los tres tamaños B, W y L para el operando por lo que, previamente, se hace la extensión de signo. El operando destino puede ser un registro de direcciones, en cuyo caso los flags del CCR no se afectan.
ADDX	ADDX.t Dn,Dm ADDX.t –(An),-(Am)	B,W,L	No	*****	Suma el operando fuente junto con el bit de extensión, X, al operando destino y almacena el resultado en el destino. Los flags XNVC del CCR se modifican de idéntica forma que en las instrucciones anteriores. El flag Z se pone a cero cuando el resultado de la instrucción es distinto de cero, en caso contrario, no cambia de valor. Esta utilidad permite determinar, en operaciones de múltiple precisión, si el resultado obtenido es cero. Supongamos que, inicialmente, se pone Z=1. Si el resultado de la primera suma es cero, este indicador no cambia; si los resultados de las siguientes sumas son cero, el indicador avisa de que el resultado global de la suma ha sido cero. Si alguna de las sumas parciales ha sido distinta de cero, Z se pone a 0.
SUB	SUB.t <ea>,Dn SUB.t Dn,<amea>	B,W,L	No	*****	Resta binaria: (destino)-(fuento)→(destino).
SUBA	SUBA.t <ea>,An	W,L	No	- - - - -	Restar dirección: An- (fuente)→An
SUBI	SUBI.t #<dato>,<adea>	B,W,L	No	*****	Restar inmediato: (destino)-dato→ (destino)
SUBQ	SUBQ.t #<d3>,<aea>	B,W,L	No	*****	Restar rápido: (destino)- datao→(destino). El dato inmediato es un número comprendido entre el 1 y el 8.
SUBX	SUBX.t Dn,Dm SUBX.t –(An),-(Am)	B,W,L	No	*****	Resta extendida: (destino)-(fuente)-X→(destino). Los flags del CCR se afectan de idéntica forma que en la instrucción ADDX

NOTAS:

X toma el mismo valor que C para todas estas instrucciones aritméticas.(2) N se pone a 1 si el bit más significativo del operando destino es un 1, 0 en caso contrario (3) Z se pone a 1 si el resultado almacenado en el destino es 0, y a 0 en caso contrario (salvo para las instrucciones ADDX y SUBX en las que Z se pone a 0 si el resultado es distinto de 0 y no cambia si el resultado es 0) (4) C,V se activan si en la suma o resta aritméticas se ha generado un acarreo o un desbordamiento(overflow)

INSTRUCCIONES DE ARITMÉTICA ENTERA (continuación)

INSTRUCCIÓN	SINTAXIS	TAMAÑO(t)	Privilegiada	XNZVC	COMENTARIO
NEG	NEG.t <adea>	B,W,L	No	* ****	Calcula la diferencia entre 0 y el operando destino, almacenando el resultado sobre la misma posición destino: 0-(destino)→ (destino)
NEGX	NEGX.t <adea>	B,W,L	no	*****	Negación con extensión. 0-(destino)-X→ (destino). Todos los flags, salvo Z, cambian igual que en la instrucción NEG. Z se pone a 0 si el resultado no es cero y no cambia en caso contrario.
DIVU	DIVU <dea>,Dn	W	No	-* * * 0	División sin signo. (destino)/(fuente)→(destino) resto→destino[31:16]; cociente→destino[15:0]. Usando aritmética sin signo, divide el operando destino entre el fuente y el resultado se almacena en el operando destino. El operando destino es de tamaño palabra larga (y siempre un registro de datos) mientras que el fuente es de tamaño palabra. El cociente generado se almacena en la palabra baja del registro de datos y el resto en la palabra alta. En la división puede generarse dos situaciones especiales: a) división por cero, en cuyo caso se produce una excepción b) Detección de desbordamiento, en cuyo caso el bit V se pone a 1 pero los operandos no se ven afectados.
DIVS	DIVS <dea>,Dn	W	No	-* * * 0	División con signo. (destino)/(fuente)→(destino) resto→destino[31:16]; cociente→destino[15:0]. Usando aritmética con signo, divide el operando destino entre el fuente y el resultado se almacena en el operando destino. El operando destino es de tamaño palabra larga (y siempre un registro de datos) mientras que el fuente es de tamaño palabra. El cociente generado se almacena en la palabra baja del registro de datos y el resto en la palabra alta. El signo del resto es el mismo que el del difidendo, a no ser que el resto sea cero.En la división puede generarse dos situaciones especiales: c) división por cero, en cuyo caso se produce una excepción d) Detección de desbordamiento, en cuyo caso el bit V se pone a 1 pero los operandos no se ven afectados.
MULU	MULU <dea>,Dn	W	No	-**00	(destino)*(fuente)→ (destino). Usando aritmética sin signo, multiplica el operando fuente por el destino (ambos de 16 bits), almacenando el resultado en este último. El resultado generado es de 32 bits.
MULS	MULS <dea>,Dn	W	No	-**00	(destino)*(fuente)→ (destino). Usando aritmética con signo, multiplica el operando fuente por el destino (ambos de 16 bits), almacenando el resultado en este último. El resultado generado es de 32 bits.

NOTAS:

(1) X toma el mismo valor que C para las operaciones NEG y NEGX, para la división y multiplicación, X no cambia (2) N toma el valor del bit más signficativo del operando destino; (3) Z se pone a 1 si el resultado es cero, salvo para la instrucción NEGX (ver ADDX o SUBX); (4) V se activa en las operaciones de división si el resultado es mayor de 16 bits y V=0 para la multiplicación. (5) C=0 para las operaciones de division y multiplicación

INSTRUCCIONES DE ARITMÉTICA ENTERA (continuación)

INSTRUCCIÓN	SINTAXIS	TAMAÑO(t)	Privilegiada	XNZVC	COMENTARIO
CMP	CMP.t <ea>,Dn	B,W,L	no	-****	(destino)-(fuente). Resta el operando destino con el fuente y actúa sobre los flags del registro CCR de acuerdo con el resultado generado; este resultado no se almacena y por tanto el operando destino permanece inalterado
CMPA	CMPA.t <ea>,An	W,L	no	-****	(destino)-(fuente). Si la instrucción es de tamaño palabra, el operando fuente se extiende en signo a 32 bits y la operación se realiza usando los 32 bits del registro de direcciones
CMPI	CMPI.t #<dato>,<adea>	B,W,L	no	-****	Dato-(destino). Resta el dato inmediato con el operando destino.
CMPM	CMPM.t (An)+,(Am)+	B,W,L	no	-****	(destino)-(fuente).
EXT	EXT.t Dn	W,L	no	-**00	(destino) extendido en signo→(destino). Extiende el bit de signo de un registro de datos de un byte a una palabra o desde una palabra a palabra larga, según el tamaño seleccionado. Si la operación es tamaño palabra, el byte [7] del registro de datos se copia sobre los bits [15:8] del mismo registro. Si la operación es de tamaño palabra larga, el byte [15] del registro de datos se copia sobre los bits [31:16] del mismo registro
CLR	CLR.t <adea>	B,W,L	no	-0100	0→destino. Pone todos los bts del operando destino a cero.

NOTAS:
(1) C y V son cero en operaciones de un solo operando y cambian en las operaciones aritméticas
(2) X no cambia de valore para estas instrucciones

INSTRUCCIONES LÓGICAS

INSTRUCCIÓN	SINTAXIS	TAMAÑO(t)	Privilegiada	XNZVC	COMENTARIO
AND	AND.t <dea>,Dn AND.t Dn,<amea>	B,W,L	no	- **00	(destino) AND (fuente) →(destino). No pueden usarse como operandos los registros de direcciones
ANDI	ANDI,t #<dato>,<adea>	B,W,L	no	-**00	(destino) AND dato →(destino). Aunque AND permite direccionamiento inmediato para el operando fuente, esta instrucción permite una gran variedad de modos de direccionamiento para el operando destino, cosa que AND no.
ANDI to CCR	ANDI #<d8>,CCR	B	no	*****	Dato AND CCR→CCR. Realiza la AND lógica del dato con el contenido del CCR almacenando el resultado en CCR
ANDI to SR	ANDI #<d16>,SR	W	si	*****	Dato AND SR → SR. Si S=1, se realiza la and lógica del dato de 16 bits con el SR, almacenando el resultado en SR. Si S=0, se produce una excepción
OR	OR.t <dea>,Dn OR.t Dn,<amea>	B,W,L	no	- **00	(destino) OR (fuente) →(destino). No pueden usarse como operandos los registros de direcciones
ORI	ORI,t #<dato>,<adea>	B,W,L	no	-**00	(destino) OR dato →(destino).
ORI to CCR	ORI #<d8>,CCR	B	no	*****	Dato OR CCR→CCR. Realiza la OR lógica del dato con el contenido del CCR almacenando el resultado en CCR
ORI to SR	ORI #<d16>,SR	W	si	*****	Dato OR SR → SR. Si S=1, se realiza la OR lógica del dato de 16 bits con el SR, almacenando el resultado en SR. Si S=0, se produce una excepción
EOR	EOR.t Dn,<adea>	B,W,L	no	- **00	(destino) EXOR (fuente) →(destino). No pueden usarse como operandos los registros de direcciones
EORI	EORI,t #<dato>,<adea>	B,W,L	no	-**00	(destino) EXOR dato →(destino).
EORI to CCR	EORI #<d8>,CCR	B	no	*****	Dato EXOR CCR→CCR. Realiza la EXOR lógica del dato con el contenido del CCR almacenando el resultado en CCR
EORI to SR	EORI #<d16>,SR	W	si	*****	Dato EXOR SR → SR. Si S=1, se realiza la EXOR lógica del dato de 16 bits con el SR, almacenando el resultado en SR. Si S=0, se produce una excepción
NOT	NOT.t <adea>	B,W,L	no	-**00	Realiza el complemento lógico del operando destino, almacenando el resultado en este. NOT (destino)→(destino)

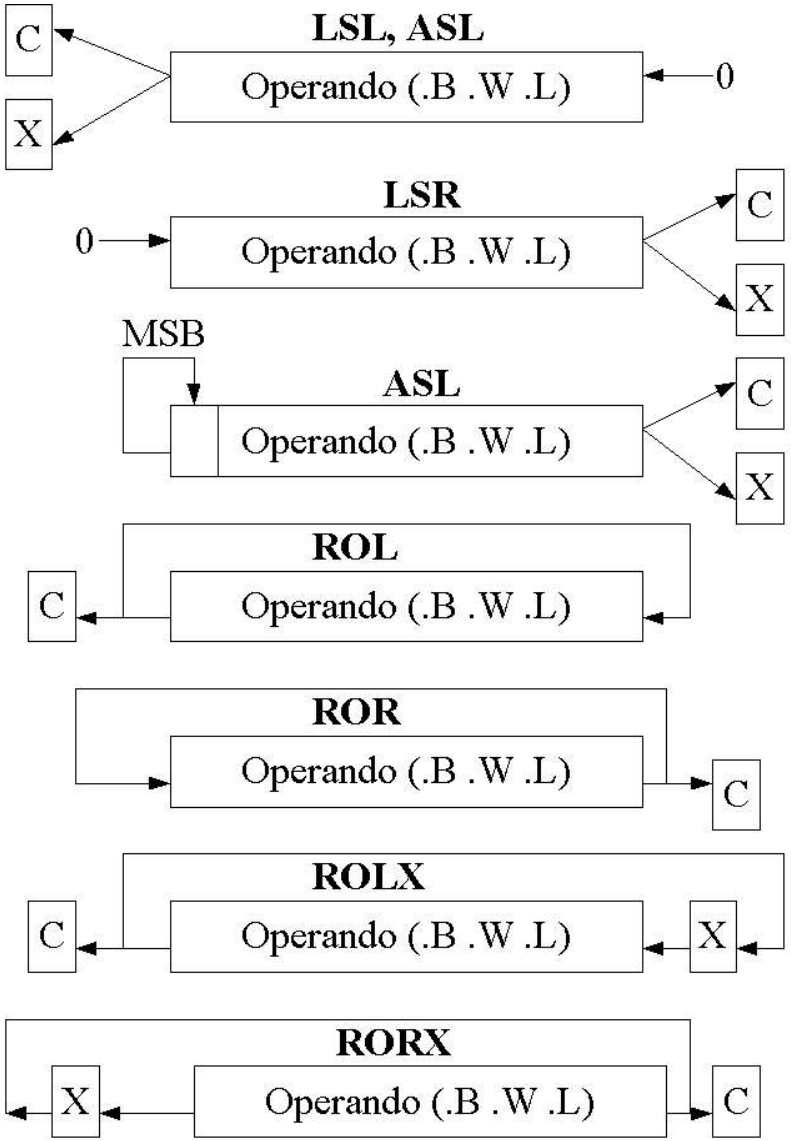
NOTAS:
(1) C y V son cero en operaciones lógicas (*)
(2) X no cambia de valor para las instrucciones lógicas (*)
(3) N toma el mismo valor que el bit más signficativo del resultado (*)
(4) Z se pone a 1 si el resultado es 0, 0 en caso contrario (*)
(*) salvo para aquellas instrucciones que tienen como destino el registro SR (ANDI to CCR, ANDI to SR, ORI to CCR, ORI to SR, EORI to CCR y EORI to SR)

INSTRUCCIONES DE DESPLAZAMIENTO Y ROTACIÓN

INSTRUCCIÓN	SINTAXIS	TAMAÑO(t)	Privilegiada	XNZVC	COMENTARIO
LSL	LSL.t Dn,Dm LSL.t #<d3>,Dm LSL.W <amea>	B,W,L	no	***0*	Desplazamiento lógico a la izquierda. Desplaza hacia la izquierda el contenido del operando destino tantas veces como se indique en el operando fuente, si éste existe, o una sóla vez si no existe (caso de LSL.W <amea>). En un desplazamiento a izquierda las posiciones que van quedando vacías en la derecha se rellenan con 0's, mientras que los bits que salen por la izquierda pasan al flag C y X. Existen varios formatos: a) Mediante registro de datos. Se desplazan tantos bits como indique el registro de datos Dn b) Mediante un dato inmediato, cuyo rango sólonpuede ir de 1 a 8. c) El desplazamiento puede ser de 1 bit implícito en la propia instrucción. En este caso, el operando destino es la memoria y su tamaño es de palabra.
ASL	ASL.t Dn,Dm ASL.t #<d3>,Dm ASL.W <amea>	B,W,L	no	*****	Desplazamiento aritmético a la izquierda. Es igual que LSL. En este caso, el bit V se pone a 1 cuando hay un cambio de signo en el bit más significativo del operando destino
ROL	ROL.t Dn,Dm ROL.t #<d3>,Dm ROL.W <amea>	B,W,L	no	-**0*	Rotación a la izquierda. Es un desplazamiento a la izquierda en el que el bit más significativo además de pasar al flag C se introduce en la posición de la derecha del operando que se queda vacía.
ROXL	ROXL.t Dn,Dm ROXL.t #<d3>,Dm ROXL.W <amea>	B,W,L	no	***0*	Rotación a izquierda con extensión. Es una rotación a la izquierda en la que interviene el flag X. El bit desplazado de mayor peso se introduce en C y X. A su vez, la posición menos significativa del operando destino se ocupa con el contenido del flag X.
LSR	LSR.t Dn,Dm LSR.t #<d3>,Dm LSR.W <amea>	B,W,L	no	***0*	Desplazamiento lógico a la derecha. Desplaza hacia la derecha el contenido del operando fuente tantas veces como se indique en el operando fuente, si éste existe, o una sóla vez si no existe(caso de LSR.W <amea>). En un desplazamiento a la derecha las posiciones que van quedando vacías en la izquierda se rellenan con 0's, mientras que los bits que salen por la derecha pasan al flag C y X. Existen varios formatos: (Igual que LSL)
ASR	ASR.t Dn,Dm ASR.t #<d3>,Dm ASR.W <amea>	B,W,L	no	***0*	Desplazamiento aritmético a la derecha. Es igual que LSR pero la posición de mayor peso del operando destino se rellena con el contenido de esa misma posición., en lugar de 0.
ROR	ROR.t Dn,Dm ROR.t #<d3>,Dm ROR.W <amea>	B,W,L	no	-**0*	Rotación a la derecha. Es un desplazamiento a la derecha en el que el bit menos significativo además de pasar al flag C se introduce en la posición de la derecha del operando que se queda vacía.
ROXR	ROXR.t Dn,Dm ROXR.t #<d3>,Dm ROXR.W <amea>	B,W,L	no	***0*	Rotación a derecha con extensión. Es una rotación a la derecha en la que interviene el flag X. El bit desplazado de menor peso se introduce en C y X. A su vez, la posición más significativa del operando destino se ocupa con el contenido del flag X.

NOTAS:

(1) V es cero en operaciones de rotación/desplazamiento salvo en las aritméticas(ASL) en cuyo caso se activa si se produce un cambio de signo del operando destino (pasa de ser positivo a negativo o viceversa).
(2) X toma el mismo valor que C para todas las instrucciones salvo para las de rotación (ROL,ROX), en cuyo caso mantiene su valor.
(3) C toma el valor del último bit desplazado del operando destino
(4) Z se pone a 1 si el resultado de la operación provoca que el operando destino sea cero. En caso contrario, Z=0
(5) N toma el mismo valor que el bit más signficativo del resultado


LSL, ASL
C
X
Operando (.B .W .L)
0
LSR
0
Operando (.B .W .L)
C
X
MSB
ASL
Operando (.B .W .L)
C
X
ROL
C
Operando (.B .W .L)
ROR
Operando (.B .W .L)
C
ROLX
C
Operando (.B .W .L)
X
RORX
X
Operando (.B .W .L)
C

INSTRUCCIONES DE CONTROL DE PROGRAMA

INSTRUCCIÓN	SINTAXIS	Privilegiada	XNZVC	COMENTARIO
Bcc	Bcc <etiqueta> Bcc.S <etiqueta>	no	- - - - -	Bifurcación condicional. Se genera un salto a la posición del programa ensamblador que contenga el identificador de etiqueta. Si dicha posición se puede expresar como un desplazamiento de 8 bits (Ca2), usaremos Bcc.S, en caso contrario, desplazamiento de 16 bits(Ca2), Bcc. Un desplazamiento de 8 bits permite saltar a una posición de memoria situada a una distancia de ±127 de la posición actual, mientras que un desplazamiento de 16 bits, a una distancia de ±32767 bytes. Los códigos de condición vienen resumidos en la Tabla 1
BRA	BRA <etiqueta> BRA.S <etiqueta>	no	- - - - -	Bifurcación incondicional. Salto corto BRA.S, salto largo BRA.
BSR	BSR <etiqueta> BSR.S <etiqueta>	no	- - - - -	Salto a subrutina. Salto corto BSR.S, salto largo BSR. PC→-(SP) y PC + desplazamiento (8 o 16 bits) →PC. La dirección de la instrucción inmediatamente siguiente a la instrucción BSR se almacena en la pila con tamaño palabra larga. La ejecución del programa continúa en la posición PC+desplazamiento.
JMP	JMP <cea>	no	- - - - -	(destino)→PC. La ejecución del programa continúa en la dirección efectiva especificada en la instrucción
JSR	JSR <cea>	no	- - - - -	PC→-(SP) y (destino)→ PC. La dirección de la instrucción inmediatamente siguiente a la instrucción BSR se almacena en la pila con tamaño palabra larga. La ejecución del programa continúa en (destino).
RTS	RTS	no	- - - - -	Regreso de subrutina. Se recupera el valor del contador de programa de la pila. El antiguo valor de PC se pierde. (SP)+→PC
RTR	RTR	no	- - - - -	Regreso de subrutina y restaura el registro CCR. Esta instrucción, situada al final de la subrutina, primero restaura los valores de CCR y luego carga la dirección al PC. Esta instrucciónes es útil para aquellos saltos a subrutinas que salvan los contenidos del registro de estado. Tira de una palabra de la pila, y carga su byte menos significativo en CCR y despúes opera igual que RTS.

CODIGO DE CONDICIONES PARA Bcc

a) Aritmética con signo

CONDICION	SIGNIFICADO	CALCULO
GT	“Greater than” – mayor que	Z+(N xor V) =0
LT	“Less than” – menor que	(N xor V) =1
GE	“Greater or equal” – mayor o igual	(N xor V) =0
LE	“Less or equal” -- menor o igual	Z+(N xor V) =1
VS	“Overflow” – desbordamiento	V=1
VC	“No overflow” – sin desbordamiento	V=0
PL	“Plus” – más	N=0
MI	“Minus” -- menos	N=1

b) Aritmética sin signo

CONDICION	SIGNIFICADO	CALCULO
HI	“Higher”-- mayor	Z +C =0
CS	“Carry Set”—menor	C=1
CC	“Carry Clear”—mayor o igual	C=0
LS	“Low or same”—menro o igual	Z+C=1

c) Aritmética con signo o sin signo

CONDICION	SIGNIFICADO	CALCULO
EQ	“Equal”-- igual	Z =1
NE	“Not equal”— distinto	Z=0

En el siguiente cuadro aparecen las distintas condiciones simples que se pueden utilizar en ensamblador. La instrucción CMP activa los flags y a continuación la instrucción Bcc permite el salto en función de los flags activados

CONDICIÓN	INSTRUCCIÓN QUE EVALÚA	CÓDIGO PARA BIFURCAR SI SE CUMPLE LA CONDICIÓN	
		CON SIGNO	SIN SIGNO
A>B	CMP B,A	GT	HI
A>=B	CMP B,A	GE	CC
A<B	CMP B,A	LT	CS
A<=B	CMP B,A	LE	LS
A=B	CMP B,A	EQ	EQ
A<>B	CMP B,A	NE	NE
A positivo	TST A	PL	-
A negativo		MI	-
Desbordamiento		VS	CS
No desbordamiento		VC	CC

INSTRUCCIONES DE CONTROL DEL SISTEMA

INSTRUCCIÓN	SINTAXIS	Privilegiada	XNZVC	COMENTARIO
RESET	RESET	si	- - - - -	Si S=1 activa la línea RESET causando la inicialización de todos los dispositivos externos. El estado del microprocesador no se ve afectado, salvo el registro PC, que se incrementa en dos unidades para pasar a ejecutar la siguiente instrucción. Si S=0 se produce una excepción
RTE	RTE	si	- - - - -	Si S=1, (SP)+→SR; (SP)+→PC. Es similar a RTR, pero en este caso la palabra entera se escribe en SR. Los antiguos valores de SR y PC se pierden. Si S=0, se produce una excepción.
NOP	NOP	no	- - - - -	No se realiza ninguna operación
STOP	STOP #<d16>	si	- - - - -	Si S=1 datoinmediato→SR y STOP. El operando inmediato se mueve al registro de estado;el PC mantiene la dirección de la siguiente instrucción a ejecutar y el procesador detiene la decodificación y ejecución de instrucciones.
CHK	CHK <dea>,Dn	no	-*UUU	If Dn <0 or Dn > dea then EXCEPCION(nº6). El registro de datos se compara con 0 y con el operando fuente. Si dicho contenido sobrepasa estos valores se produce una excepción
ILLEGAL	ILLEGAL	no	- - - - -	PC→-(SSP);SR→-(SSP) (vector de intrucción ilegal: nº4→PC
TRAP	TRAP #<d4>	no	- - - - -	El procesador inicia el procesamiento de una excepción. El número de vector de excepción se especifica en la instrucción. Existen 16 vectores de trap disponibles.
TRAPV	TRAPV	no	- - - - -	If V=1 Then EXCEPCION. Si el flag V está a 1, el procesador inicia el procesamiento de la excepción nº7.