Instituto Politécnico Nacional
Escuela Superior de Cómputo

Instrucciones del DSPIC

M. En C. Victor Hugo García Ortega

Av. Juan de Dios Batiz s/n
Col Lindavista, GAM
Unidad Profesional Zacatenco
07738, Ciudad de México.

Instrucciones

Los DSPIC soportan dos tipos principales de instrucciones:

Las instrucciones típicas de los microcontroladores clásicos que se denominan instrucciones MCU (65 instrucciones).

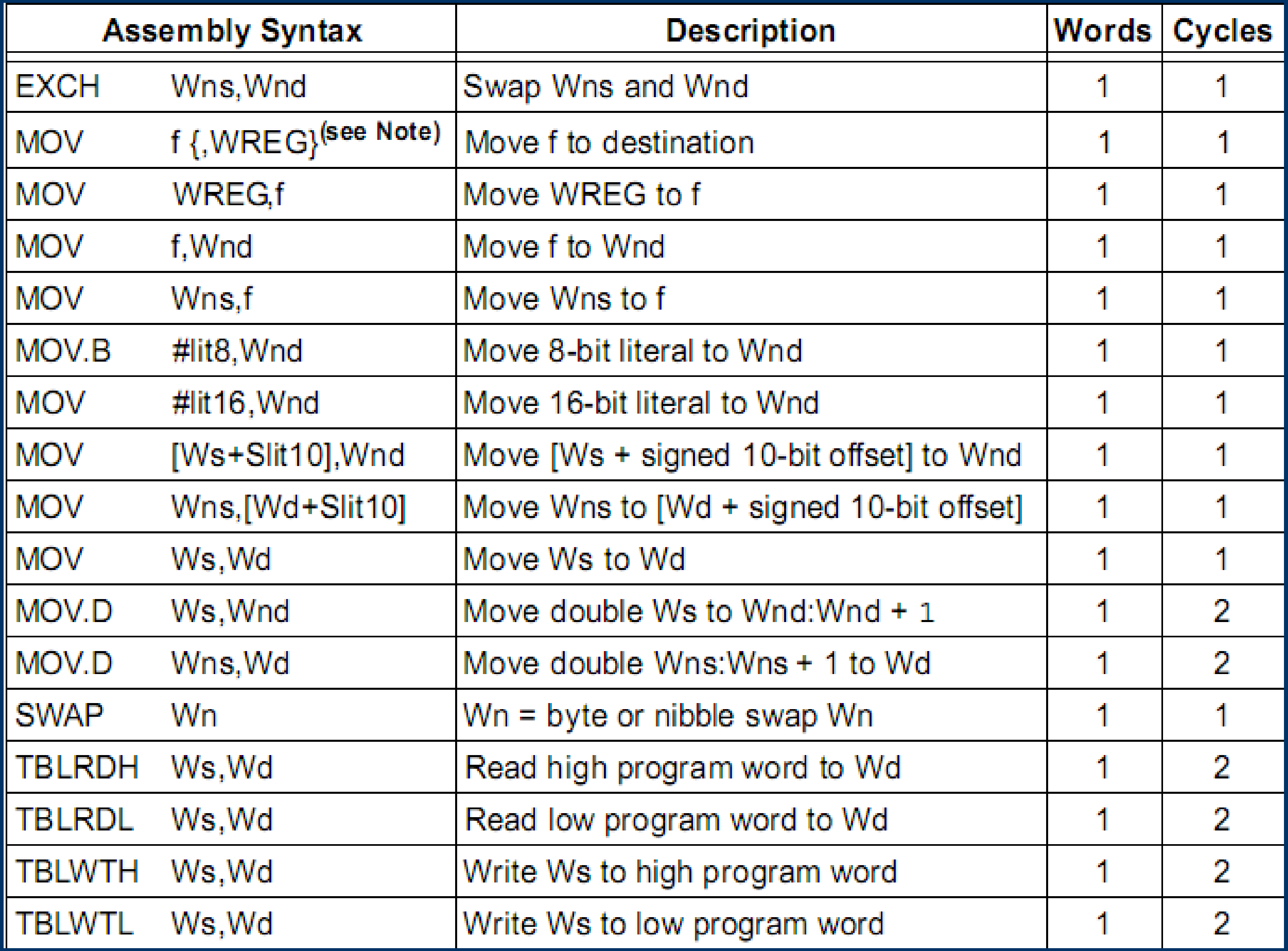
Las instrucciones específicas de los DSP (19 instrucciones).

Estas 84 instrucciones pueden ser agrupadas en 10 categorías:

Instrucciones

Instrucciones de movimiento.
Instrucciones matemáticas.
Instrucciones lógicas.
Instrucciones de rotación y corrimiento.
Instrucciones de manejo de bits.
Instrucciones de comparación y escape.
Instrucciones de control de flujo del programa
Instrucciones de pila y registros sombra.
Instrucciones de control.
Instrucciones DSP.

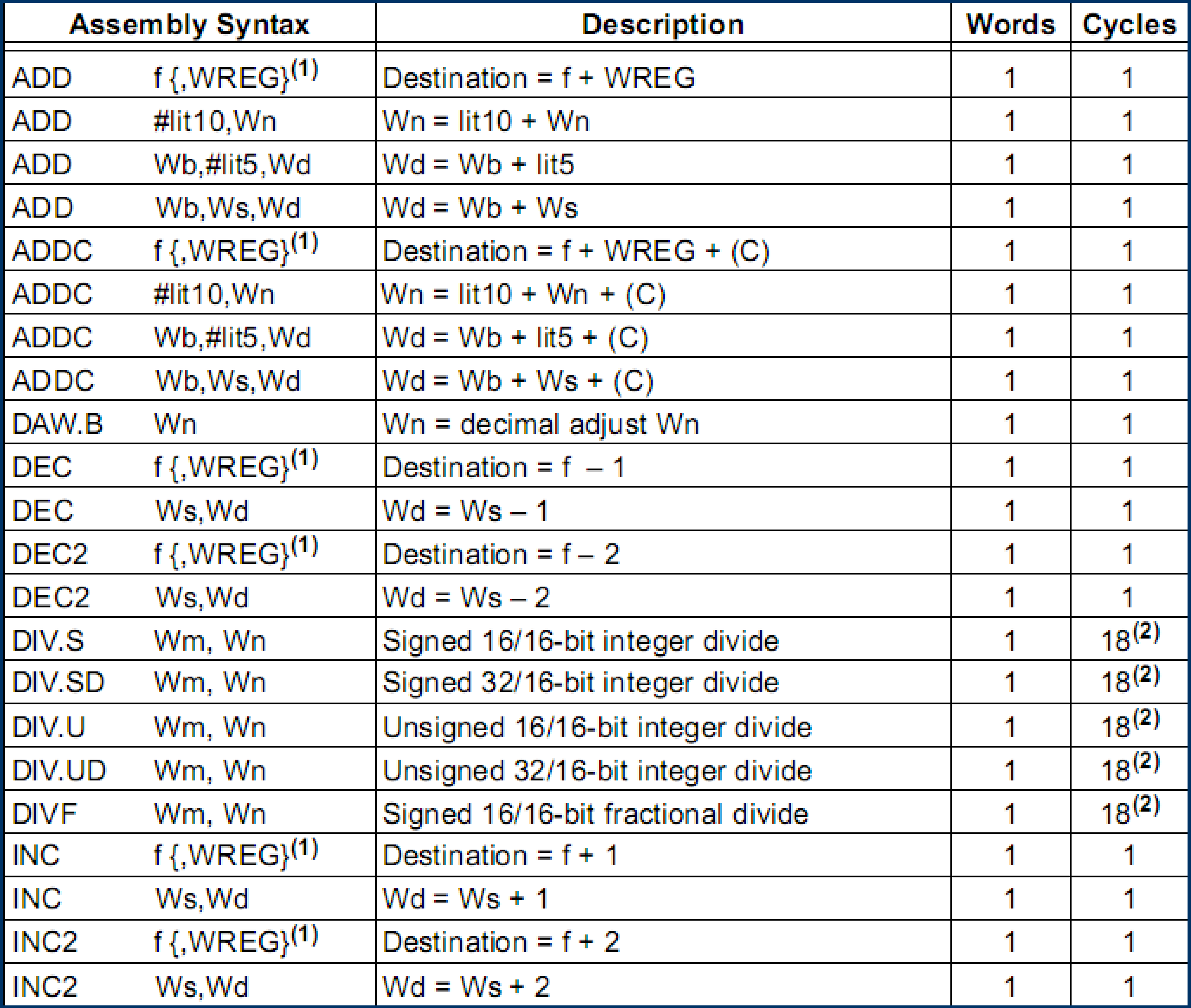
Instrucciones de movimiento



Assembly Syntax		Description	Words	Cycles
EXCH	Wns,Wnd	Swap Wns and Wnd	1	1
MOV	f {,WREG}(see Note)	Move f to destination	1	1
MOV	WREG,f	Move WREG to f	1	1
MOV	f,Wnd	Move f to Wnd	1	1
MOV	Wns,f	Move Wns to f	1	1
MOV.B	#lit8,Wnd	Move 8-bit literal to Wnd	1	1
MOV	#lit16,Wnd	Move 16-bit literal to Wnd	1	1
MOV	[Ws+Slit10],Wnd	Move [Ws + signed 10-bit offset] to Wnd	1	1
MOV	Wns,[Wd+Slit10]	Move Wns to [Wd + signed 10-bit offset]	1	1
MOV	Ws,Wd	Move Ws to Wd	1	1
MOV.D	Ws,Wnd	Move double Ws to Wnd:Wnd + 1	1	2
MOV.D	Wns,Wd	Move double Wns:Wns + 1 to Wd	1	2
SWAP	Wn	Wn = byte or nibble swap Wn	1	1
TBLRDH	Ws,Wd	Read high program word to Wd	1	2
TBLRDL	Ws,Wd	Read low program word to Wd	1	2
TBLWTH	Ws,Wd	Write Ws to high program word	1	2
TBLWTL	Ws,Wd	Write Ws to low program word	1	2

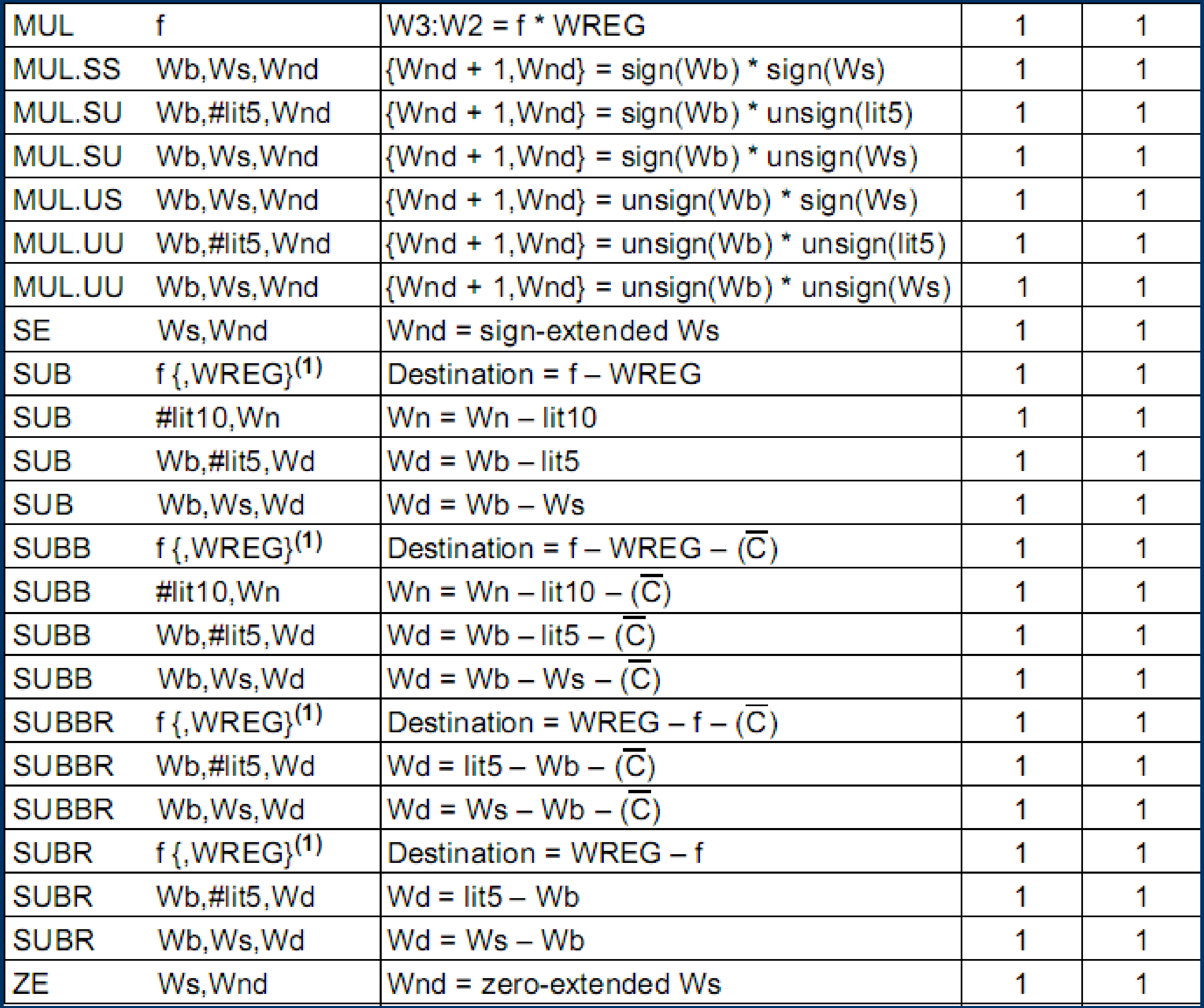
Instrucciones matemáticas

Estas instrucciones MCU soportan las operaciones aritméticas principales como la suma, resta, multiplicación y división



Assembly Syntax		Description	Words	Cycles
ADD	f {,WREG}(1)	Destination = f + WREG	1	1
ADD	#lit10,Wn	Wn = lit10 + Wn	1	1
ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1
ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1
ADDC	f {,WREG}(1)	Destination = f + WREG + (C)	1	1
ADDC	#lit10,Wn	Wn = lit10 + Wn + (C)	1	1
ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1
ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1
DAW.B	Wn	Wn = decimal adjust Wn	1	1
DEC	f {,WREG}(1)	Destination = f – 1	1	1
DEC	Ws,Wd	Wd = Ws – 1	1	1
DEC2	f {,WREG}(1)	Destination = f – 2	1	1
DEC2	Ws,Wd	Wd = Ws – 2	1	1
DIV.S	Wm, Wn	Signed 16/16-bit integer divide	1	18(2)
DIV.SD	Wm, Wn	Signed 32/16-bit integer divide	1	18(2)
DIV.U	Wm, Wn	Unsigned 16/16-bit integer divide	1	18(2)
DIV.UD	Wm, Wn	Unsigned 32/16-bit integer divide	1	18(2)
DIVF	Wm, Wn	Signed 16/16-bit fractional divide	1	18(2)
INC	f {,WREG}(1)	Destination = f + 1	1	1
INC	Ws,Wd	Wd = Ws + 1	1	1
INC2	f {,WREG}(1)	Destination = f + 2	1	1
INC2	Ws,Wd	Wd = Ws + 2	1	1

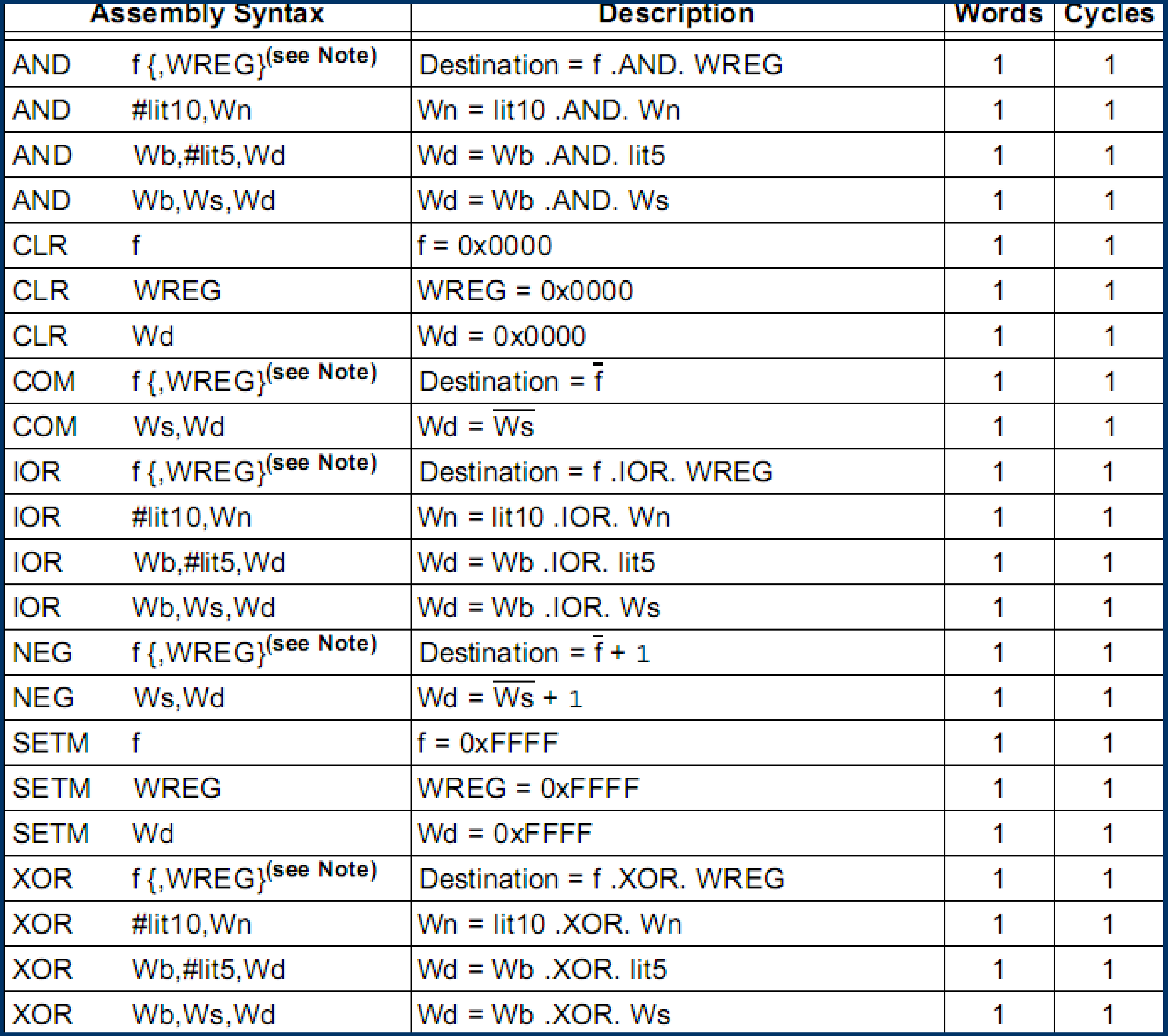
Instrucciones matemáticas



MUL	f	W3:W2 = f * WREG	1	1
MUL.SS	Wb,Ws,Wnd	{Wnd + 1,Wnd} = sign(Wb) * sign(Ws)	1	1
MUL.SU	Wb,#lit5,Wnd	{Wnd + 1,Wnd} = sign(Wb) * unsign(lit5)	1	1
MUL.SU	Wb,Ws,Wnd	{Wnd + 1,Wnd} = sign(Wb) * unsign(Ws)	1	1
MUL.US	Wb,Ws,Wnd	{Wnd + 1,Wnd} = unsign(Wb) * sign(Ws)	1	1
MUL.UU	Wb,#lit5,Wnd	{Wnd + 1,Wnd} = unsign(Wb) * unsign(lit5)	1	1
MUL.UU	Wb,Ws,Wnd	{Wnd + 1,Wnd} = unsign(Wb) * unsign(Ws)	1	1
SE	Ws,Wnd	Wnd = sign-extended Ws	1	1
SUB	f {,WREG}(1)	Destination = f – WREG	1	1
SUB	#lit10,Wn	Wn = Wn – lit10	1	1
SUB	Wb,#lit5,Wd	Wd = Wb – lit5	1	1
SUB	Wb,Ws,Wd	Wd = Wb – Ws	1	1
SUBB	f {,WREG}(1)	Destination = f – WREG – ($\overline{C}$)	1	1
SUBB	#lit10,Wn	Wn = Wn – lit10 – ($\overline{C}$)	1	1
SUBB	Wb,#lit5,Wd	Wd = Wb – lit5 – ($\overline{C}$)	1	1
SUBB	Wb,Ws,Wd	Wd = Wb – Ws – ($\overline{C}$)	1	1
SUBBR	f {,WREG}(1)	Destination = WREG – f – ($\overline{C}$)	1	1
SUBBR	Wb,#lit5,Wd	Wd = lit5 – Wb – ($\overline{C}$)	1	1
SUBBR	Wb,Ws,Wd	Wd = Ws – Wb – ($\overline{C}$)	1	1
SUBR	f {,WREG}(1)	Destination = WREG – f	1	1
SUBR	Wb,#lit5,Wd	Wd = lit5 – Wb	1	1
SUBR	Wb,Ws,Wd	Wd = Ws – Wb	1	1
ZE	Ws,Wnd	Wnd = zero-extended Ws	1	1

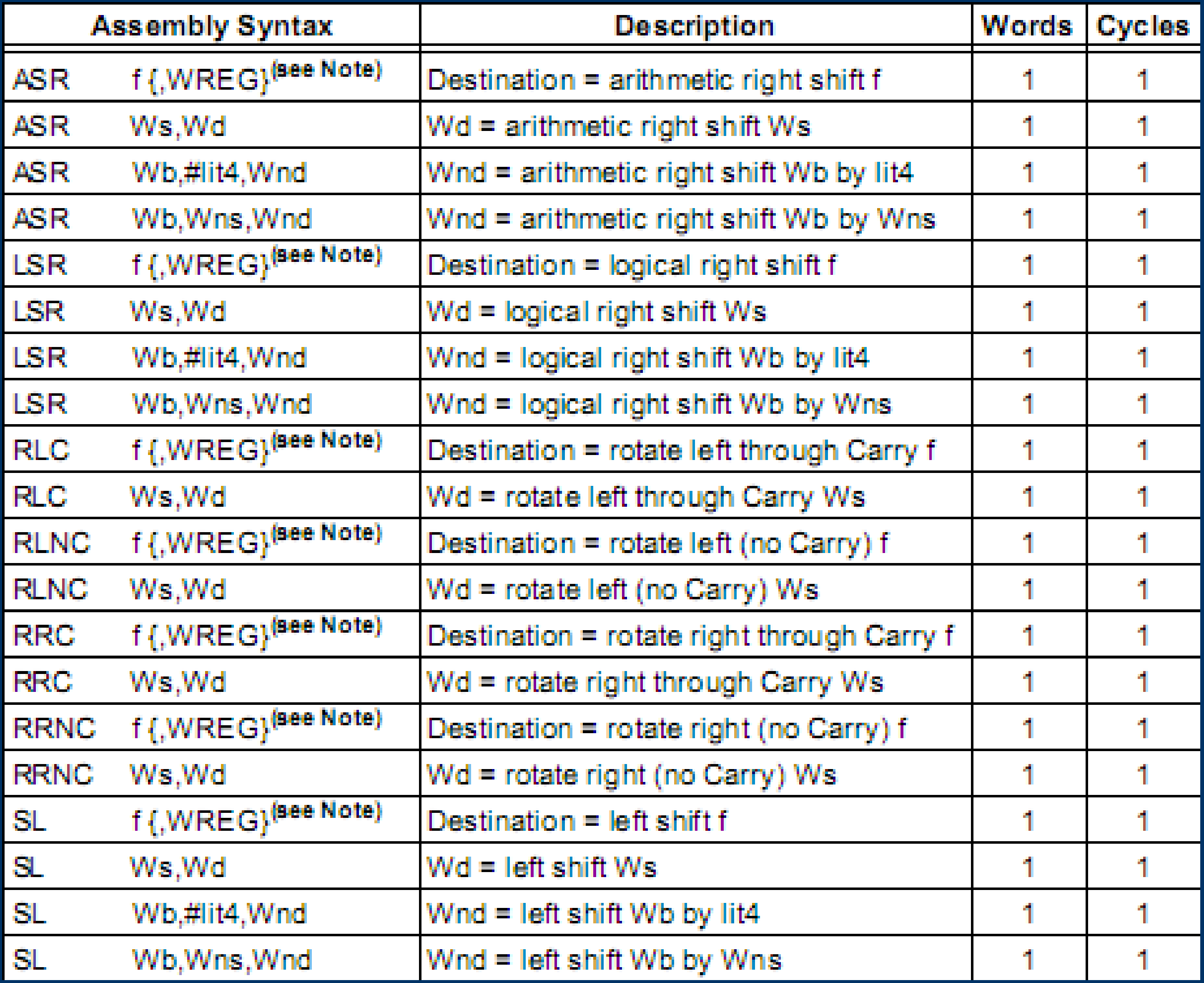
Instrucciones lógicas

Estas instrucciones realizan una operación lógica bit a bit entre sus dos operandos



Assembly Syntax		Description	Words	Cycles
AND	f {,WREG}(see Note)	Destination = f .AND. WREG	1	1
AND	#lit10,Wn	Wn = lit10 .AND. Wn	1	1
AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1
AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1
CLR	f	f = 0x0000	1	1
CLR	WREG	WREG = 0x0000	1	1
CLR	Wd	Wd = 0x0000	1	1
COM	f {,WREG}(see Note)	Destination = $\overline{f}$	1	1
COM	Ws,Wd	Wd = $\overline{Ws}$	1	1
IOR	f {,WREG}(see Note)	Destination = f .IOR. WREG	1	1
IOR	#lit10,Wn	Wn = lit10 .IOR. Wn	1	1
IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1
IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1
NEG	f {,WREG}(see Note)	Destination = $\overline{f}$ + 1	1	1
NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1
SETM	f	f = 0xFFFF	1	1
SETM	WREG	WREG = 0xFFFF	1	1
SETM	Wd	Wd = 0xFFFF	1	1
XOR	f {,WREG}(see Note)	Destination = f .XOR. WREG	1	1
XOR	#lit10,Wn	Wn = lit10 .XOR. Wn	1	1
XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1
XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1

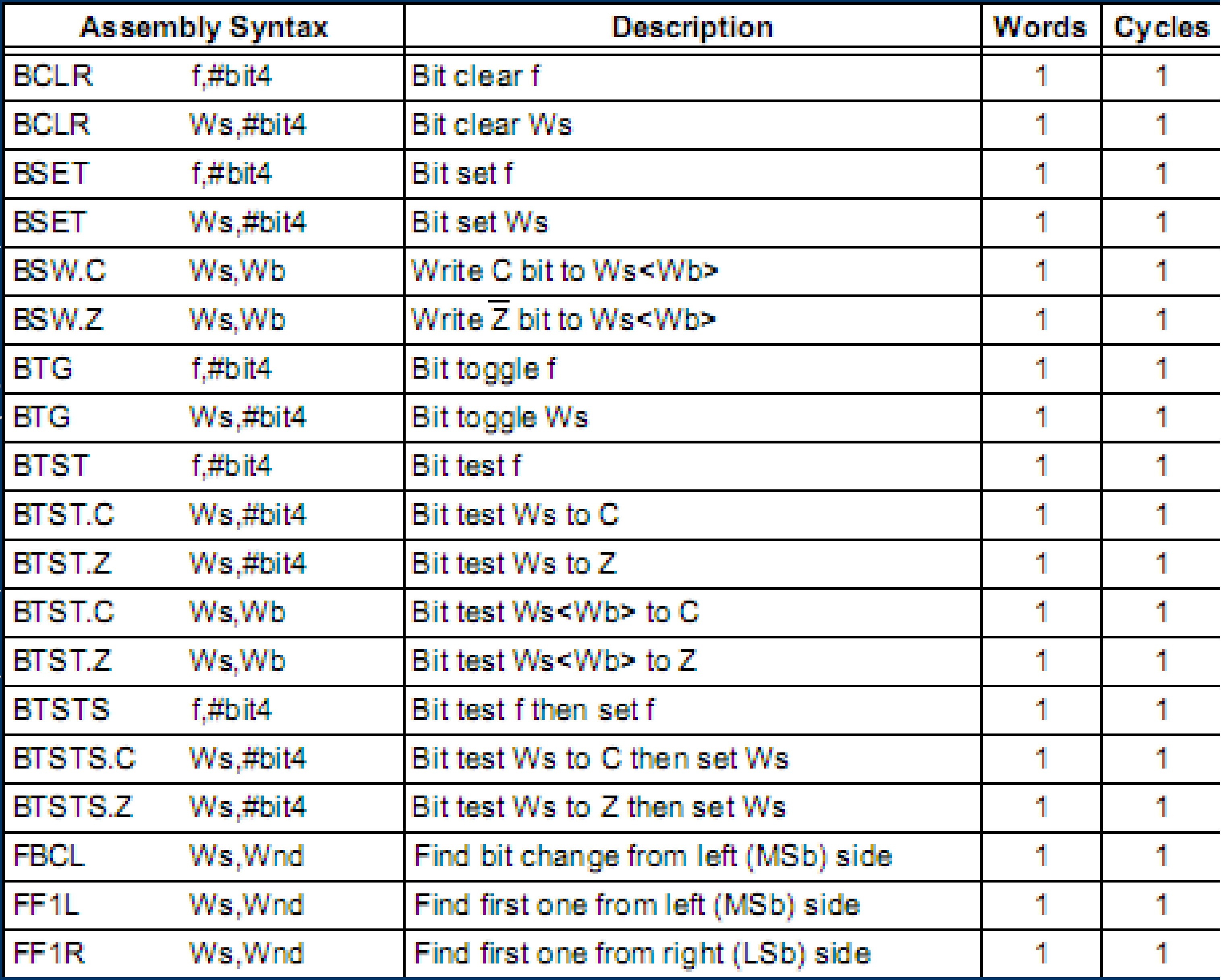
Instrucciones de rotación y corrimiento



Assembly Syntax		Description	Words	Cycles
ASR	f {,WREG}(see Note)	Destination = arithmetic right shift f	1	1
ASR	Ws,Wd	Wd = arithmetic right shift Ws	1	1
ASR	Wb,#lit4,Wnd	Wnd = arithmetic right shift Wb by lit4	1	1
ASR	Wb,Wns,Wnd	Wnd = arithmetic right shift Wb by Wns	1	1
LSR	f {,WREG}(see Note)	Destination = logical right shift f	1	1
LSR	Ws,Wd	Wd = logical right shift Ws	1	1
LSR	Wb,#lit4,Wnd	Wnd = logical right shift Wb by lit4	1	1
LSR	Wb,Wns,Wnd	Wnd = logical right shift Wb by Wns	1	1
RLC	f {,WREG}(see Note)	Destination = rotate left through Carry f	1	1
RLC	Ws,Wd	Wd = rotate left through Carry Ws	1	1
RLNC	f {,WREG}(see Note)	Destination = rotate left (no Carry) f	1	1
RLNC	Ws,Wd	Wd = rotate left (no Carry) Ws	1	1
RRC	f {,WREG}(see Note)	Destination = rotate right through Carry f	1	1
RRC	Ws,Wd	Wd = rotate right through Carry Ws	1	1
RRNC	f {,WREG}(see Note)	Destination = rotate right (no Carry) f	1	1
RRNC	Ws,Wd	Wd = rotate right (no Carry) Ws	1	1
SL	f {,WREG}(see Note)	Destination = left shift f	1	1
SL	Ws,Wd	Wd = left shift Ws	1	1
SL	Wb,#lit4,Wnd	Wnd = left shift Wb by lit4	1	1
SL	Wb,Wns,Wnd	Wnd = left shift Wb by Wns	1	1

Instrucciones de manejo de bits

Estas instrucciones realizan una operación sobre un bit concreto de una posición de la memoria de datos o de registro W.

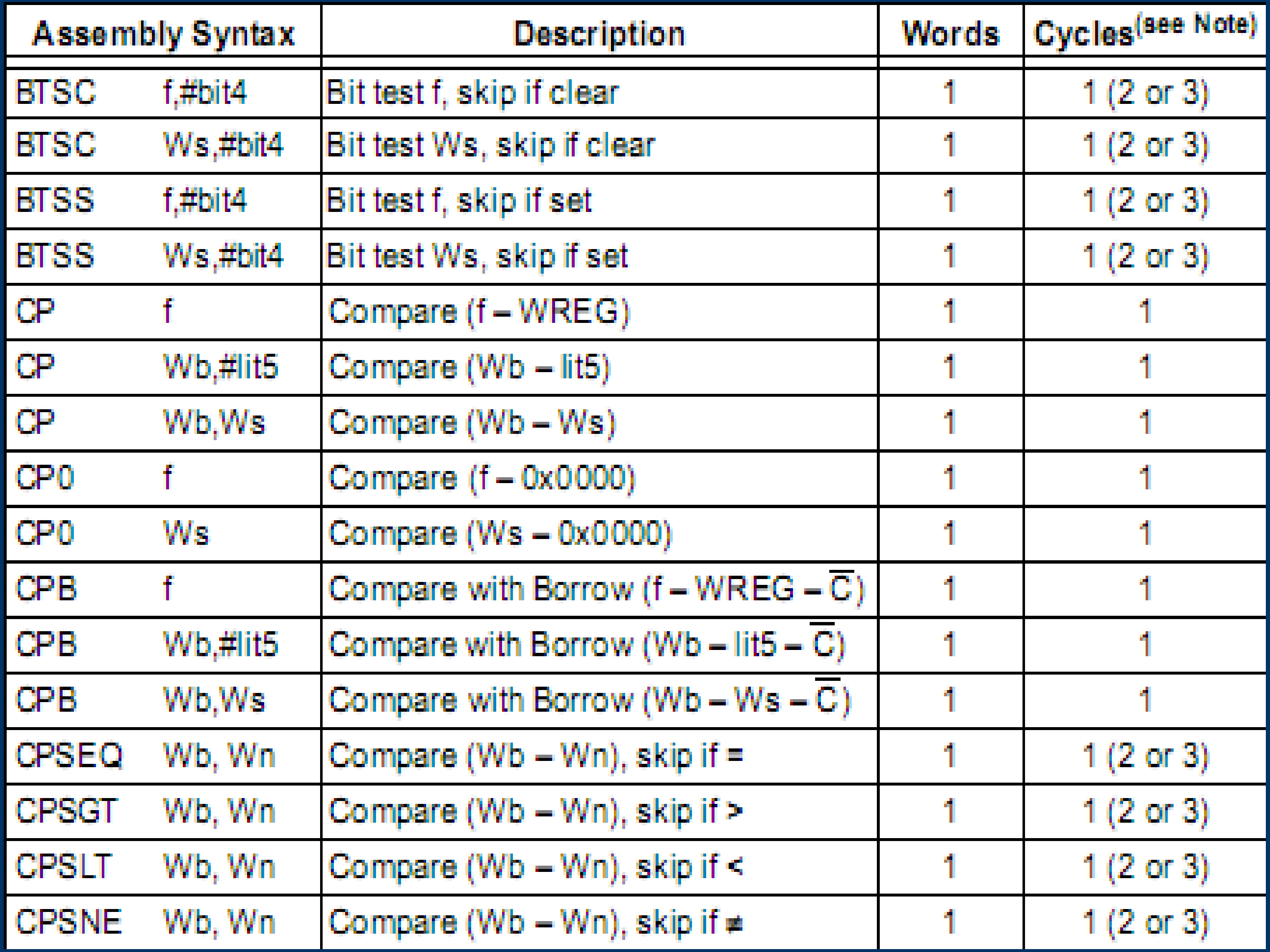


Assembly Syntax		Description	Words	Cycles
BCLR	f,#bit4	Bit clear f	1	1
BCLR	Ws,#bit4	Bit clear Ws	1	1
BSET	f,#bit4	Bit set f	1	1
BSET	Ws,#bit4	Bit set Ws	1	1
BSW.C	Ws,Wb	Write C bit to Ws<Wb>	1	1
BSW.Z	Ws,Wb	Write $\overline{Z}$ bit to Ws<Wb>	1	1
BTG	f,#bit4	Bit toggle f	1	1
BTG	Ws,#bit4	Bit toggle Ws	1	1
BTST	f,#bit4	Bit test f	1	1
BTST.C	Ws,#bit4	Bit test Ws to C	1	1
BTST.Z	Ws,#bit4	Bit test Ws to Z	1	1
BTST.C	Ws,Wb	Bit test Ws<Wb> to C	1	1
BTST.Z	Ws,Wb	Bit test Ws<Wb> to Z	1	1
BTSTS	f,#bit4	Bit test f then set f	1	1
BTSTS.C	Ws,#bit4	Bit test Ws to C then set Ws	1	1
BTSTS.Z	Ws,#bit4	Bit test Ws to Z then set Ws	1	1
FBCL	Ws,Wnd	Find bit change from left (MSb) side	1	1
FF1L	Ws,Wnd	Find first one from left (MSb) side	1	1
FF1R	Ws,Wnd	Find first one from right (LSb) side	1	1

Instrucciones de comparación y escape

Estas instrucciones comprueban una condición o comparan un valor que puede ser el de un bit concreto y según sea su valor binario se brinca o no. El brinco “skip” consiste en saltarse la ejecución de la instrucción siguiente, ejecutando en lugar de ella un NOP. Cuando no hay brinco la instrucción se ejecuta en un ciclo de instrucción, pero si lo hay la duración es de 2 ciclos si el brinco se efectúa sobre una instrucción de una palabra y de 3, si se trata de una instrucción de 2 palabras

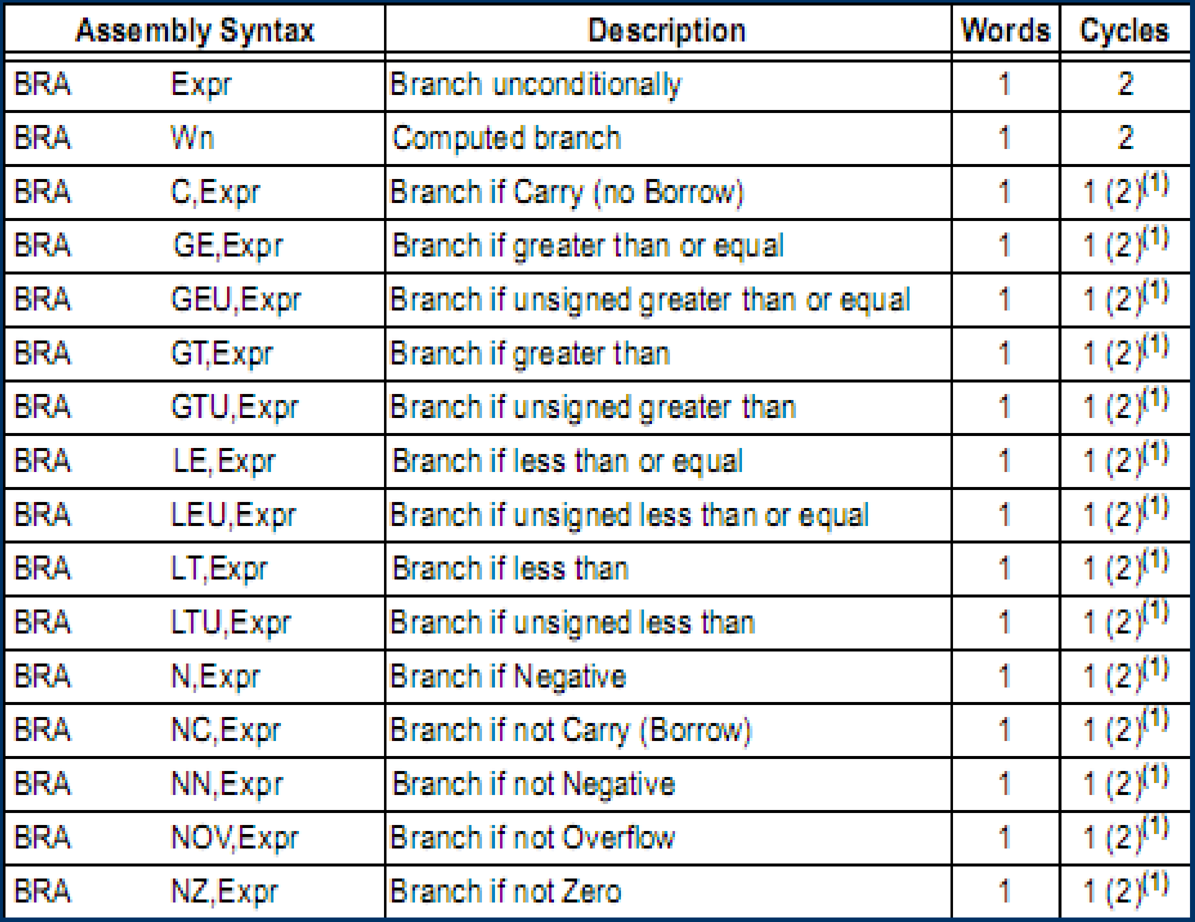
Instrucciones de comparación y escape



Assembly Syntax		Description	Words	Cycles(see Note)
BTSC	f,#bit4	Bit test f, skip if clear	1	1 (2 or 3)
BTSC	Ws,#bit4	Bit test Ws, skip if clear	1	1 (2 or 3)
BTSS	f,#bit4	Bit test f, skip if set	1	1 (2 or 3)
BTSS	Ws,#bit4	Bit test Ws, skip if set	1	1 (2 or 3)
CP	f	Compare (f – WREG)	1	1
CP	Wb,#lit5	Compare (Wb – lit5)	1	1
CP	Wb,Ws	Compare (Wb – Ws)	1	1
CP0	f	Compare (f – 0x0000)	1	1
CP0	Ws	Compare (Ws – 0x0000)	1	1
CPB	f	Compare with Borrow (f – WREG – $\overline{C}$)	1	1
CPB	Wb,#lit5	Compare with Borrow (Wb – lit5 – $\overline{C}$)	1	1
CPB	Wb,Ws	Compare with Borrow (Wb – Ws – $\overline{C}$)	1	1
CPSEQ	Wb, Wn	Compare (Wb – Wn), skip if =	1	1 (2 or 3)
CPSGT	Wb, Wn	Compare (Wb – Wn), skip if >	1	1 (2 or 3)
CPSLT	Wb, Wn	Compare (Wb – Wn), skip if <	1	1 (2 or 3)
CPSNE	Wb, Wn	Compare (Wb – Wn), skip if ≠	1	1 (2 or 3)

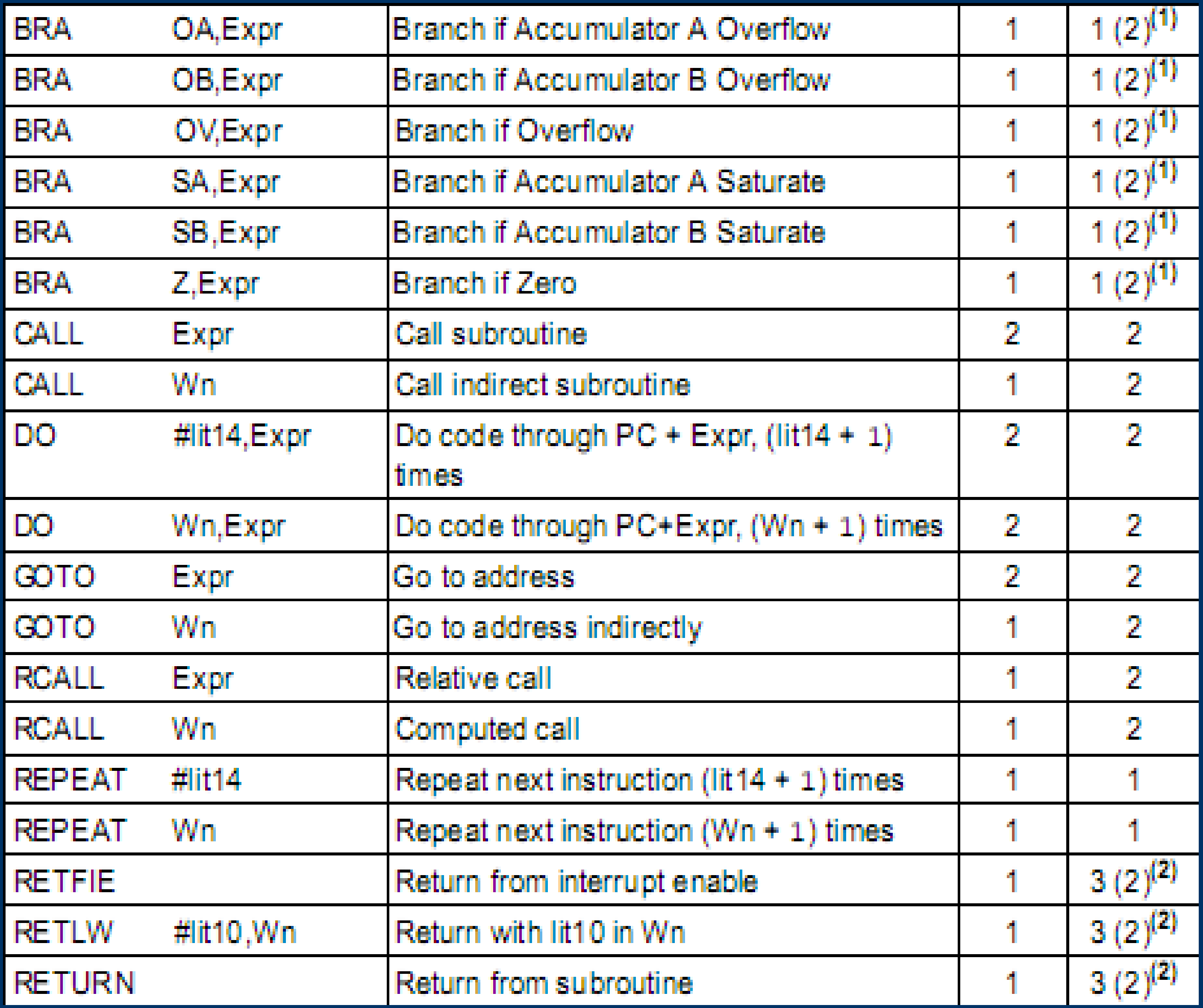
Instrucciones de control de flujo del programa

Estas instrucciones controlan el valor del PC y por tanto el flujo del programa en curso.



Assembly Syntax		Description	Words	Cycles
BRA	Expr	Branch unconditionally	1	2
BRA	Wn	Computed branch	1	2
BRA	C,Expr	Branch if Carry (no Borrow)	1	1 (2)(1)
BRA	GE,Expr	Branch if greater than or equal	1	1 (2)(1)
BRA	GEU,Expr	Branch if unsigned greater than or equal	1	1 (2)(1)
BRA	GT,Expr	Branch if greater than	1	1 (2)(1)
BRA	GTU,Expr	Branch if unsigned greater than	1	1 (2)(1)
BRA	LE,Expr	Branch if less than or equal	1	1 (2)(1)
BRA	LEU,Expr	Branch if unsigned less than or equal	1	1 (2)(1)
BRA	LT,Expr	Branch if less than	1	1 (2)(1)
BRA	LTU,Expr	Branch if unsigned less than	1	1 (2)(1)
BRA	N,Expr	Branch if Negative	1	1 (2)(1)
BRA	NC,Expr	Branch if not Carry (Borrow)	1	1 (2)(1)
BRA	NN,Expr	Branch if not Negative	1	1 (2)(1)
BRA	NOV,Expr	Branch if not Overflow	1	1 (2)(1)
BRA	NZ,Expr	Branch if not Zero	1	1 (2)(1)

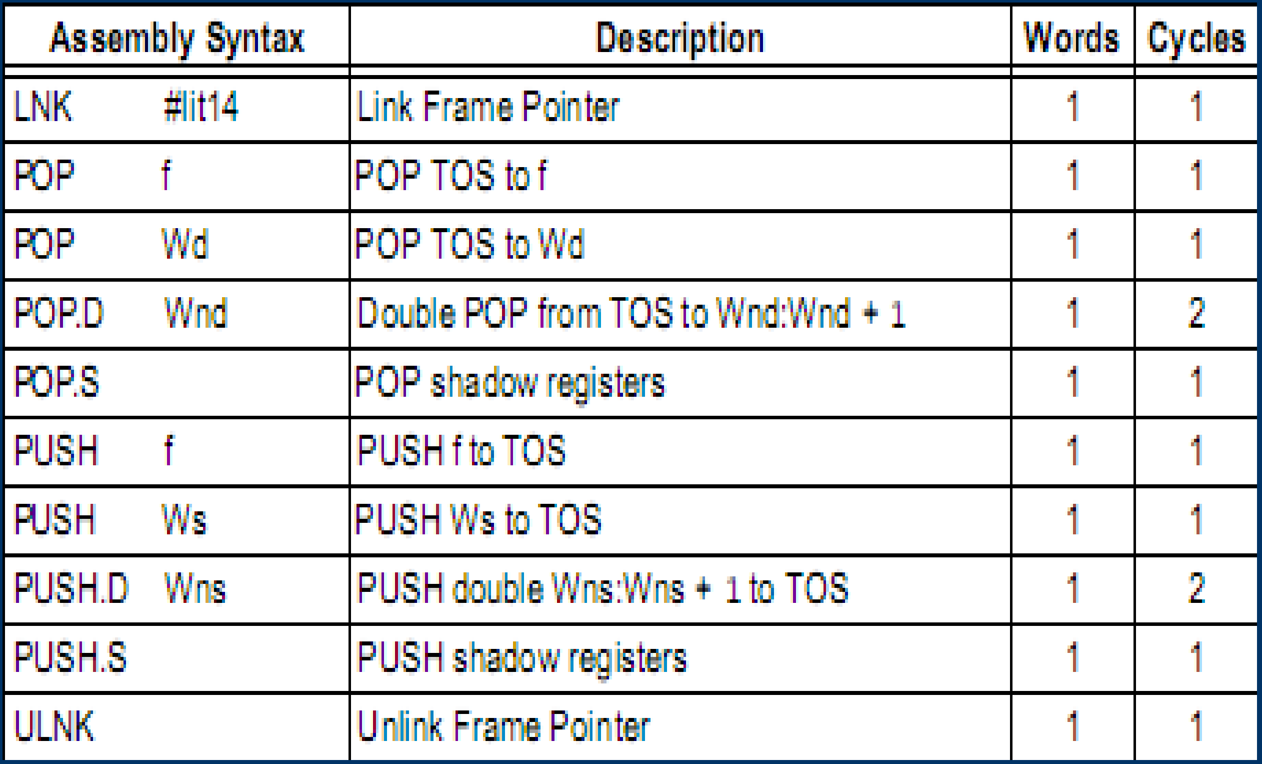
Instrucciones de control de flujo del programa



BRA	OA,Expr	Branch if Accumulator A Overflow	1	1 (2)[1]
BRA	OB,Expr	Branch if Accumulator B Overflow	1	1 (2)[1]
BRA	OV,Expr	Branch if Overflow	1	1 (2)[1]
BRA	SA,Expr	Branch if Accumulator A Saturate	1	1 (2)[1]
BRA	SB,Expr	Branch if Accumulator B Saturate	1	1 (2)[1]
BRA	Z,Expr	Branch if Zero	1	1 (2)[1]
CALL	Expr	Call subroutine	2	2
CALL	Wn	Call indirect subroutine	1	2
DO	#lit14,Expr	Do code through PC + Expr, (lit14 + 1) times	2	2
DO	Wn,Expr	Do code through PC+Expr, (Wn + 1) times	2	2
GOTO	Expr	Go to address	2	2
GOTO	Wn	Go to address indirectly	1	2
RCALL	Expr	Relative call	1	2
RCALL	Wn	Computed call	1	2
REPEAT	#lit14	Repeat next instruction (lit14 + 1) times	1	1
REPEAT	Wn	Repeat next instruction (Wn + 1) times	1	1
RETFIE		Return from interrupt enable	1	3 (2)[2]
RETLW	#lit10,Wn	Return with lit10 in Wn	1	3 (2)[2]
RETURN		Return from subroutine	1	3 (2)[2]

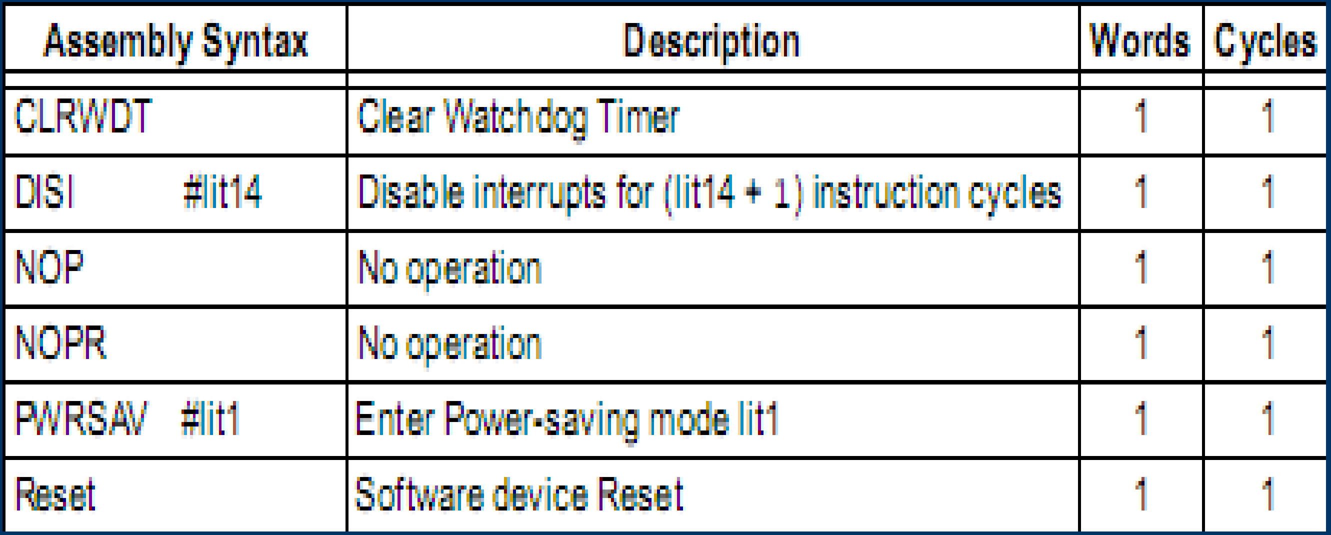
Instrucciones de pila y registros sombra

Estas instrucciones controlan el stack pointer de la pila cuya dirección esta almacenada en el registro W15. La dirección por omisión es 0x800.



Assembly Syntax		Description	Words	Cycles
LNK	#lit14	Link Frame Pointer	1	1
POP	f	POP TOS to f	1	1
POP	Wd	POP TOS to Wd	1	1
POP.D	Wnd	Double POP from TOS to Wnd:Wnd + 1	1	2
POP.S		POP shadow registers	1	1
PUSH	f	PUSH f to TOS	1	1
PUSH	Ws	PUSH Ws to TOS	1	1
PUSH.D	Wns	PUSH double Wns:Wns + 1 to TOS	1	2
PUSH.S		PUSH shadow registers	1	1
ULNK		Unlink Frame Pointer	1	1

Instrucciones de control



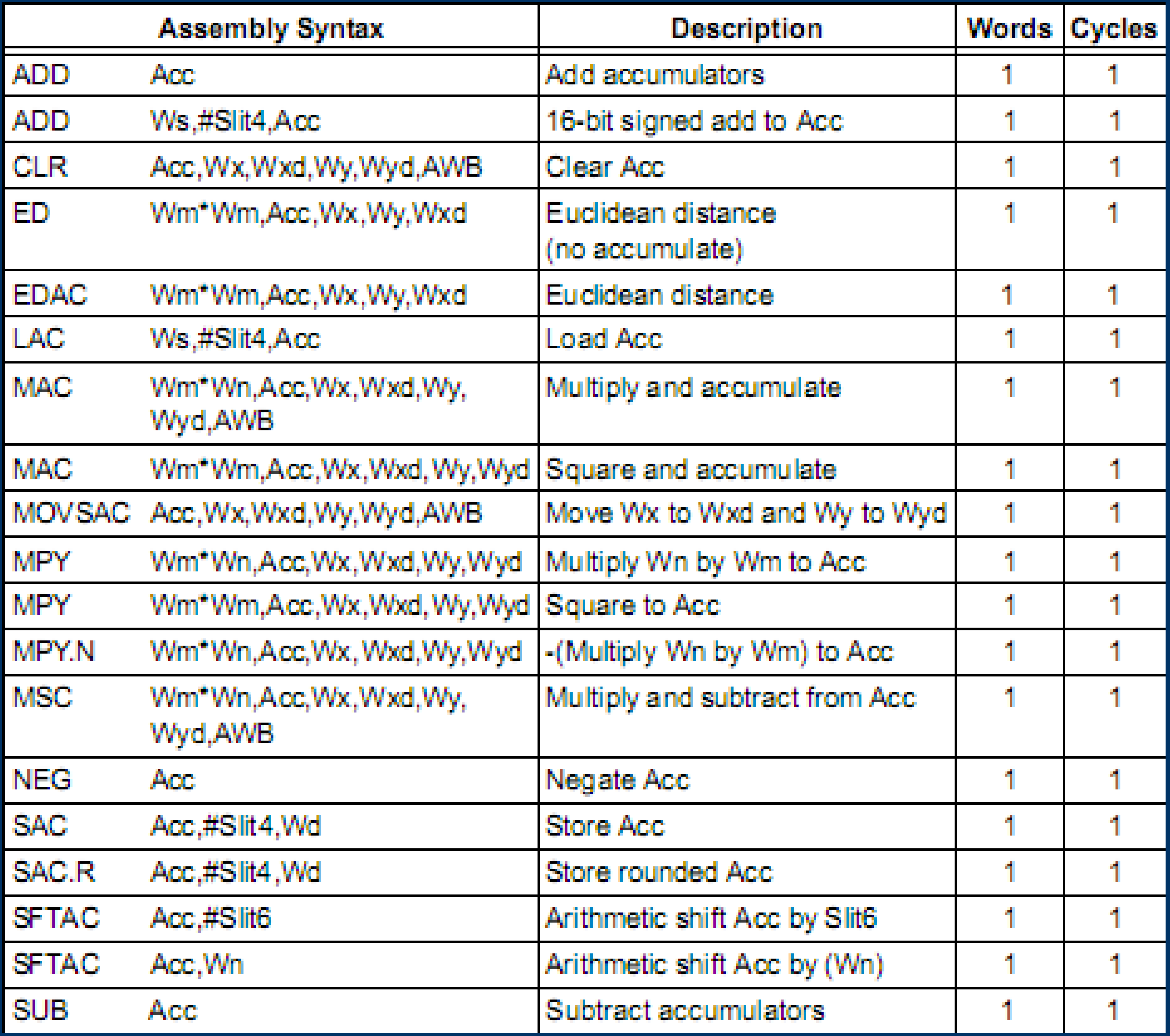
Assembly Syntax	Description	Words	Cycles
CLRWDT	Clear Watchdog Timer	1	1
DISI #lit14	Disable interrupts for (lit14 + 1) instruction cycles	1	1
NOP	No operation	1	1
NOPR	No operation	1	1
PWRSAV #lit1	Enter Power-saving mode lit1	1	1
Reset	Software device Reset	1	1

Contacto

Gracias por su atención...

e-mail:
vgarciaortega@yahoo.com.mx

Instrucciones DSP



Assembly Syntax		Description	Words	Cycles
ADD	Acc	Add accumulators	1	1
ADD	Ws,#Slit4,Acc	16-bit signed add to Acc	1	1
CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Acc	1	1
ED	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance (no accumulate)	1	1
EDAC	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance	1	1
LAC	Ws,#Slit4,Acc	Load Acc	1	1
MAC	Wm*Wn,Acc,Wx,Wxd,Wy, Wyd,AWB	Multiply and accumulate	1	1
MAC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and accumulate	1	1
MOVSAC	Acc,Wx,Wxd,Wy,Wyd,AWB	Move Wx to Wxd and Wy to Wyd	1	1
MPY	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wn by Wm to Acc	1	1
MPY	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square to Acc	1	1
MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wn by Wm) to Acc	1	1
MSC	Wm*Wn,Acc,Wx,Wxd,Wy, Wyd,AWB	Multiply and subtract from Acc	1	1
NEG	Acc	Negate Acc	1	1
SAC	Acc,#Slit4,Wd	Store Acc	1	1
SAC.R	Acc,#Slit4,Wd	Store rounded Acc	1	1
SFTAC	Acc,#Slit6	Arithmetic shift Acc by Slit6	1	1
SFTAC	Acc,Wn	Arithmetic shift Acc by (Wn)	1	1
SUB	Acc	Subtract accumulators	1	1