APPENDIX A PIC24 Architecture and Instruction Set Summary

This appendix contains a summary of the PIC24 architecture and instruction set. Figure A.1 shows an architectural block diagram of the PIC24HJ32GP202/204 and PIC24HJ16GP304 processors. Figures A.4 through A.11 give the instruction set summary, with Figures A.2 through A.4 providing symbols and encodings used in these summaries. The instruction set encodings and definitions are taken from the *dsPIC30F/33F Programmer's Reference Manual* [5].

Register placeholder symbols used in instruction mnemonics are:

- **Wn**: register direct addressing; *Wn* specifies one of `W0`, `W1`, ... `W15`.
- **Wns**: register direct addressing; *Wns* specifies one of `W0`, `W1`, ... `W15`.
- **Wnd**: register direct addressing; *Wnd* specifies one of `W0`, `W1`, ... `W15`.
- **Wb**: register direct addressing; *Wb* specifies one of `W0`, `W1`, ... `W15`.
- **WREG**: the working register; specifies `W0` in file register instructions.
- **Ws**: register direct (*Ws*) and indirect addressing modes ([*Ws*], [*Ws*++], [*Ws*−−], [++*Ws*], [−−*Ws*]); *Ws* specifies one of `W0`, `W1`, ... `W15`.
- **Wd**: register direct (*Wd*) and indirect addressing modes ([*Wd*], [*Wd*++], [*Wd*−−], [++*Wd*], [−−*Wd*]); *Wd* specifies one of `W0`, `W1`, ... `W15`.
- **Wso**: all of the addressing modes of **Ws**, with the additional mode of register offset indirect [*Wso* + *Wb*]; *Wso* specifies one of `W0`, `W1`, ... `W15`.
- **Wdo**: all of the addressing modes of **Wd**, with the additional mode of register offset indirect [*Wdo* + *Wb*]; *Wdo* specifies one of `W0`, `W1`, ... `W15`.
- **Wsi**: Indirect addressing modes ([*Ws*], [*Ws*++], [*Ws*−−], [++*Ws*], [−−*Ws*]); *Ws* specifies one of `W0`, `W1`, ... `W15`. Used only by the `tblrdl`, `tblrdh` instructions.
- **Wdi**: Indirect addressing modes ([*Wd*], [*Wd*++], [*Wd*−−], [++*Wd*], [−−*Wd*]); *Wd* specifies one of `W0`, `W1`, ... `W15`. Used only by the `tblwtl`, `tblwth` instructions.

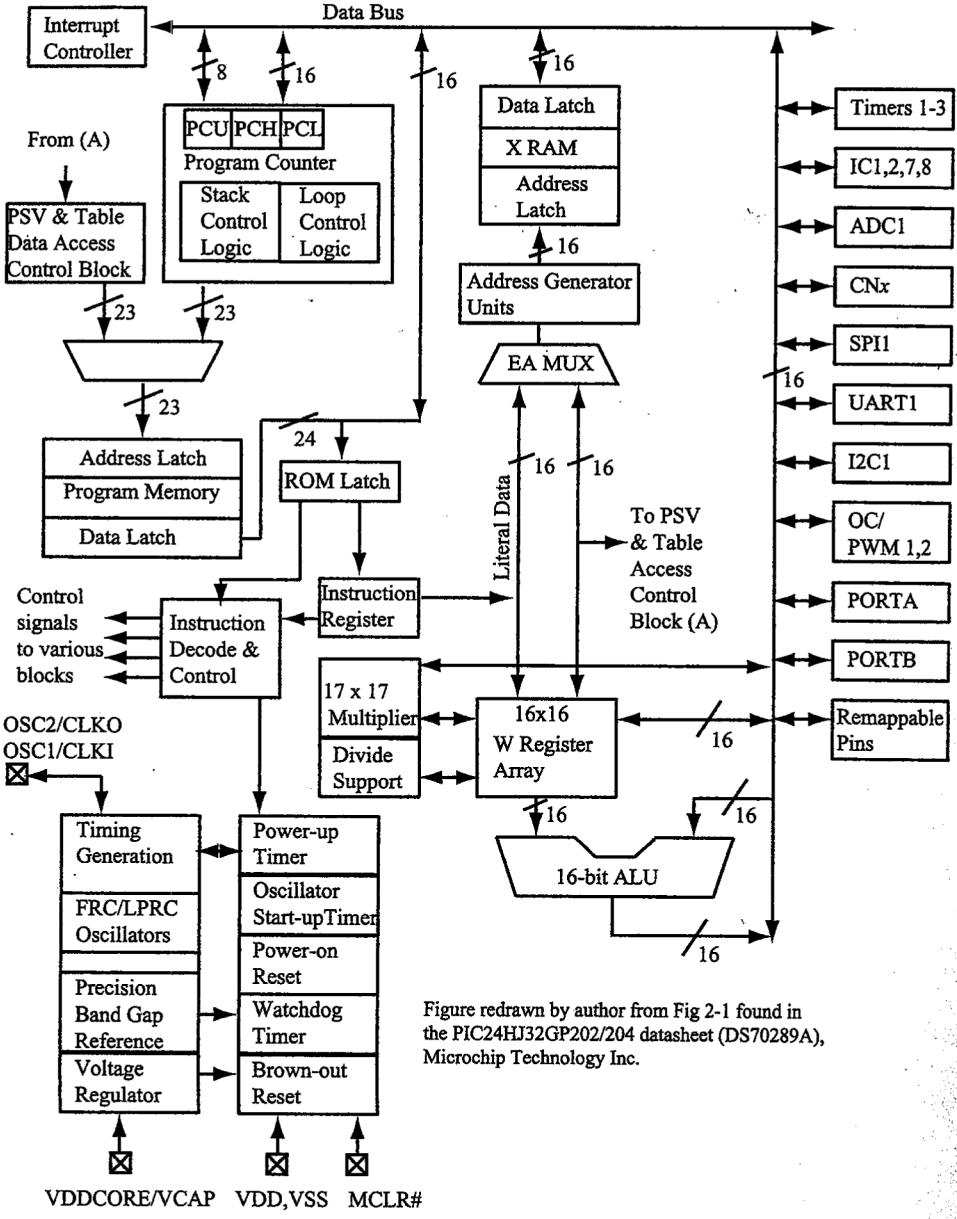

Figure redrawn by author from Fig 2-1 found in the PIC24HJ32GP202/204 datasheet (DS70289A), Microchip Technology Inc.

FIGURE A.1 PIC24HJ32GP202/204 and PIC24HJ16GP304 block diagram. (Adapted with permission of the copyright owner, Microchip Technology, Incorporated. All rights reserved. No further reprints or reproduction may be made without Microchip Inc.'s prior written consent.)

Field	Description
`b`	Byte mode selection bit : 0 = word operation; 1 = byte operation
`bbbb`	4-bit position select: 0000 = LSb; 1111 = MSb
`d`	Destination address bit: 0 = result stored in WREG; 1 = result stored in file register
`dddd`	*Wd* destination register select: 0000 = W0; 1111 = W15
`f ffff ffff ffff`	13-bit register file byte address (0x0000 to 0x1FFF)
`fff ffff ffff ffff`	15-bit register file word address (implied 0 LSb, 0x0000 to 0xFFFE)
`ffff ffff ffff ffff`	16-bit register file byte address (0x0000 to 0xFFFF)
`ggg`	Register offset addressing mode for *Ws* source register (Figure A.3)
`hhh`	Register offset addressing mode for *Wd* source register (Figure A.3)
`kkk ... k`	literal field, constant data or expression
`nnn nnnn nnn nnn0 nnn nnnn`	23-bit program address for `call` and `goto` instructions
`nnnn nnnn nnnn nnnn`	16-bit program offset field for relative branch/call instructions
`ppp`	Addressing mode for *Ws* source register (see Figure A.3)
`qqq`	Addressing mode for *Wd* destination register (see Figure A.3)
`rrrr`	Barrel shift count
`ssss`	*Ws* source register select: 0000 = W0; 1111 = W15
`tttt`	Dividend select, most significant word
`vvvv`	Dividend select, least significant word
`w`	Double Word mode selection bit: 0 = word operation; 1 = double word operation
`wwww`	*Wb* base register select: 0000 = W0; 1111 = W15
`xxxx xxxx xxxx xxxx`	16-bit unused field (don't care)

FIGURE A.2 Machine code symbols.

Addressing modes for *Ws* (source operand) and *Wd* (destination operand)

ppp/qqq	Addressing mode	Source Operand (ppp)	Destination Operand (qqq)
000	Register Direct	Ws	Wd
001	Indirect	[Ws]	[Wd]
010	Indirect with Post-Decrement	[Ws--]	[Wd--]
011	Indirect with Post-Increment	[Ws++]	[Wd++]
100	Indirect with Pre-Decrement	[--Ws]	[--Wd]
101	Indirect with Pre-Increment	[++Ws]	[++Wd]
11x	Unused	Software reset if used	

Addressing modes for *Wso* (source operand) and *Wdo* (destination operand)

ggg/hhh	Addressing mode	Source Operand (ggg)	Destination Operand (hhh)
000	Register Direct	Ws	Wd
001	Indirect	[Ws]	[Wd]
010	Indirect with Post-Decrement	[Ws--]	[Wd--]
011	Indirect with Post-Increment	[Ws++]	[Wd++]
100	Indirect with Pre-Decrement	[--Ws]	[--Wd]
101	Indirect with Pre-Increment	[++Ws]	[++Wd]
11x	Indirect with Register Offset	[Ws+Wb]	[Wd+Wb]

FIGURE A.3 Addressing mode encodings.

Instr	Descr	#W	#Cyc	Status	Machine Code
ADD{.B} f	f = f + WREG	1	1	C,DC,N,V,Z	1011 0100 0BDf ffff ffff ffff
ADD{.B} f,WREG	WREG = f + WREG	1	1	C,DC,N,V,Z	1011 0100 0BDf ffff ffff ffff
ADD{.B} #lit10,Wn	Wn = lit10 + Wn	1	1	C,DC,N,V,Z	1011 0000 0Bkk kkkk kkkk dddd
ADD{.B} Wb,Ws,Wd	Wd = Wb + Ws	1	1	C,DC,N,V,Z	0100 0www wBqq qddd dppp ssss
ADD{.B} Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C,DC,N,V,Z	0100 0www wBqq qddd d11k kkkk
ADDC{.B} f	f = f + WREG + (C)	1	1	C,DC,N,V,Z	1011 0100 1BDf ffff ffff ffff
ADDC{.B} f,WREG	WREG = f + WREG + (C)	1	1	C,DC,N,V,Z	1011 0100 1BDf ffff ffff ffff
ADDC{.B} #lit10,Wn	Wn = lit10 + Wn + (C)	1	1	C,DC,N,V,Z	1011 0000 1Bkk kkkk kkkk dddd
ADDC{.B} Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	C,DC,N,V,Z	0100 1www wBqq qddd dppp ssss
ADDC{.B} Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	C,DC,N,V,Z	0100 1www wBqq qddd d11k kkkk
AND{.B} f	f = f & WREG	1	1	N,Z	1011 0110 0BDf ffff ffff ffff
AND{.B} f,WREG	WREG = f & WREG	1	1	N,Z	1011 0110 0BDf ffff ffff ffff
AND{.B} #lit10,Wn	Wn = lit10 & Wn	1	1	N,Z	1011 0010 0Bkk kkkk kkkk dddd
AND{.B} Wb,Ws,Wd	Wd = Wb & Ws	1	1	N,Z	0110 0www wBqq qddd dppp ssss
AND{.B} Wb,#lit5,Wd	Wd = Wb & lit5	1	1	N,Z	0110 0www wBqq qddd d11k kkkk
ASR{.B} f	f = arith. >> f	1	1	N,Z,C	1101 0101 1BDf ffff ffff ffff
ASR{.B} f,WREG	WREG = arith. >> f	1	1	N,Z,C	1101 0101 1BDf ffff ffff ffff
ASR{.B} Ws,Wd	Wd = arith. >> Ws	1	1	N,Z,C	1101 0001 1Bqq qddd dppp ssss
ASR Wb,Wns,Wnd	Wnd = arith >> Wb by Wns	1	1	N,Z	1101 1110 1www wddd d000 ssss
ASR Wb,#lit4,Wnd	Wnd = arith >> Wb by lit4	1	1	N,Z	1101 1110 1www wddd d100 kkkk
BCLR{.B} f,#bit4	Bit Clear f	1	1	none	1010 1001 bbbf ffff ffff fffb
BCLR{.B} Ws,#bit4	Bit Clear Ws	1	1	none	1010 0001 bbbb 0B00 0ppp ssss
BRA C,Expr	Branch if Carry	1	1 (2)	none	0011 0001 nnnn nnnn nnnn nnnn
BRA GE,Expr	Branch if signed ≥	1	1 (2)	none	0011 1101 nnnn nnnn nnnn nnnn
BRA GEU,Expr	Branch if unsigned ≥	1	1 (2)	none	0011 0001 nnnn nnnn nnnn nnnn
BRA GT,Expr	Branch if signed >	1	1 (2)	none	0011 1100 nnnn nnnn nnnn nnnn
BRA GTU,Expr	Branch if unsigned >	1	1 (2)	none	0011 1110 nnnn nnnn nnnn nnnn

Instr	Descr	#W	#Cyc	Status	Machine Code
BRA LE,Expr	Branch if signed ≤	1	1 (2)	none	0011 0100 nnnn nnnn nnnn nnnn
BRA LEU,Expr	Branch if unsigned ≤	1	1 (2)	None	0011 0110 nnnn nnnn nnnn nnnn
BRA LT,Expr	Branch if signed <	1	1 (2)	none	0011 0101 nnnn nnnn nnnn nnnn
BRA LTU,Expr	Branch if unsigned <	1	1 (2)	none	0011 1001 nnnn nnnn nnnn nnnn
BRA N,Expr	Branch if Negative	1	1 (2)	none	0011 0011 nnnn nnnn nnnn nnnn
BRA NC,Expr	Branch if Not Carry	1	1 (2)	none	0011 1001 nnnn nnnn nnnn nnnn
BRA NN,Expr	Branch if Not Negative	1	1 (2)	none	0011 1011 nnnn nnnn nnnn nnnn
BRA NOV,Expr	Branch if Not Overflow	1	1 (2)	none	0011 1000 nnnn nnnn nnnn nnnn
BRA NZ,Expr	Branch if Not Zero	1	1 (2)	none	0011 1010 nnnn nnnn nnnn nnnn
BRA OV,Expr	Branch if Overflow	1	1 (2)	none	0011 0000 nnnn nnnn nnnn nnnn
BRA Expr	Branch Unconditionally	1	2	none	0011 0111 nnnn nnnn nnnn nnnn
BRA Z,Expr	Branch if Zero	1	1 (2)	none	0011 0010 nnnn nnnn nnnn nnnn
BRA Wn	Computed Branch	1	2	none	0000 0001 0110 0000 0000 ssss
BSET{.B} f,#bit4	Bit Set f	1	1	none	1010 1000 bbbf ffff ffff fffb
BSET{.B} Ws,#bit4	Bit Set Ws	1	1	none	1010 0000 bbbb 0B00 0ppp ssss
BSW.C Ws,Wb	Write C bit Ws<Wb>	1	1	none	1010 1101 0www w000 0ppp ssss
BSW.Z Ws,Wb	Write Z bit Ws<Wb>	1	1	none	1010 1101 1www w000 0ppp ssss
BTG{.B} f,#bit4	Bit Toggle f	1	1	none	1010 1010 bbbf ffff ffff fffb
BTG{.B} Ws,#bit4	Bit Toggle Ws	1	1	none	1010 0010 bbbb 0B00 0ppp ssss
BTSC{.B} f,#bit4	Bit Test f, Skip If Clear	1	1 (2 or 3)	none	1010 1111 bbbf ffff ffff fffb
BTSC{.B} Ws,#bit4	Bit Test Ws, Skip If Clear	1	1 (2 or 3)	none	1010 0111 bbbb 0000 0ppp ssss
BTSS{.B} f,#bit4	Bit Test f, Skip if Set	1	1 (2 or 3)	none	1010 1110 bbbf ffff ffff fffb
BTSS{.B} Ws,#bit4	Bit Test Ws, Skip If Set	1	1 (2 or 3)	none	1010 0110 bbbb 0000 0ppp ssss
BTST f,#bit4	Bit Test f	1	1	Z	1010 1011 bbbf ffff ffff fffb
BTST.C Ws,#bit4	Bit Test Ws to C	1	1	C	1010 0011 bbbb 0000 0ppp ssss
BTST.Z Ws,#bit4	Bit Test Ws to Z	1	1	Z	1010 0011 bbbb 1000 0ppp ssss
BTST.C Ws,Wb	Bit Test Ws<Wb> to C	1	1	C	1010 0101 0www w000 0ppp ssss
BTST.Z Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z	1010 0101 1www w000 0ppp ssss

Instr	Descr	#W	#Cyc	Status	Machine Code
BTSTS f,#bit4	Bit Test then Set f	1	1	Z	1010 1100 bbbf ffff ffff fffb
BTSTS.C Ws,#bit4	Bit Test Ws to C, then Set	1	1	C	1010 0100 bbbb 0000 0ppp ssss
BTSTS.Z Ws,#bit4	Bit Test Ws to Z, then Set	1	1	Z	1010 0100 bbbb 1000 0ppp ssss
CALL Expr	Call subroutine	2	2	none	0000 0010 nnnn nnnn nnnn nnn0 0000 0000 0000 0000 0nnn nnnn
CALL Wn	Call Indirect subroutine	1	2	none	0000 0001 0000 0000 0000 ssss
CLR{.B} f	f = 0x0000	1	1	none	1110 1111 0BDf ffff ffff ffff
CLR{.B} WREG	WREG = 0x0000	1	1	none	1110 1111 0BDf ffff ffff ffff
CLR{.B} Wd	Wd = 0x0000	1	1	none	1110 1011 0Bqq qddd d000 0000
CLRWDT	Clear Watchdog Timer	1	1	none	1111 1110 0110 0000 0000 0000
COM{.B} f	f = ~f	1	1	N,Z	1110 1110 1BDf ffff ffff ffff
COM{.B} f,WREG	WREG = ~f	1	1	N,Z	1110 1110 1BDf ffff ffff ffff
COM{.B} Ws,Wd	Wd = ~Ws	1	1	N,Z	1110 1010 1Bqq qddd dppp ssss
CP{.B} f	f - WREG	1	1	C,DC,N,V,Z	1110 0011 0B0f ffff ffff ffff
CP{.B} Wb,#lit5	Wb - #lit5	1	1	C,DC,N,V,Z	1110 0001 0www wB00 011k kkkk
CP{.B} Wb,Ws	Wb - Ws	1	1	C,DC,N,V,Z	1110 0001 0www wB00 0ppp ssss
CP0{.B} f	f - 0	1	1	C,DC,N,V,Z	1110 0010 0B0f ffff ffff ffff
CP0{.B} Ws	Ws - 0	1	1	C,DC,N,V,Z	1110 0000 0000 0B00 0ppp ssss
CPB{.B} f	f – WREG – BORROW (~C)	1	1	C,DC,N,V,Z	1110 0011 1B0f ffff ffff ffff
CBB{.B} Wb,#lit5	Wb - #lit5 – BORROW (~C)	1	1	C,DC,N,V,Z	1110 0001 1www wB00 011k kkkk
CPB{.B} Wb,Ws	Wb – Ws - BORROW (~C)	1	1	C,DC,N,V,Z	1110 0001 1www wB00 0ppp ssss
CPSEQ Wb,Wn	Wb – Wn; skip if equal	1	1 (2 or 3)	none	1110 0111 1www wB00 0000 ssss
CPSGT Wb,Wn	Wb – Wn; skip if signed >	1	1 (2 or 3)	none	1110 0110 0www wB00 0000 ssss
CPSLT Wb,Wn	Wb – Wn; skip if signed <	1	1 (2 or 3)	none	1110 0110 1www wB00 0000 ssss
CPSNE Wb,Wn	Wb – Wn; skip if not equal	1	1 (2 or 3)	none	1110 0111 0www wB00 0000 ssss
DAW Wn	Wn = decimal adjust Wn	1	1	C	1111 1101 0100 0000 0000 ssss
DEC{.B} f	f = f - 1	1	1	C,DC,N,V,Z	1110 1101 0BDf ffff ffff ffff
DEC{.B} f,WREG	WREG = f - 1	1	1	C,DC,N,V,Z	1110 1101 0BDf ffff ffff ffff

Instr	Descr	#W	#Cyc	Status	Machine Code
DEC{.B} Ws,Wd	Wd = Ws - 1	1	1	C,DC,N,V,Z	1110 1001 0Bqq qddd dppp ssss
DEC2{.B} f	f = f - 2	1	1	C,DC,N,V,Z	1110 1101 1BDf ffff ffff ffff
DEC2{.B} f,WREG	WREG = f - 2	1	1	C,DC,N,V,Z	1110 1101 1BDf ffff ffff ffff
DEC2{.B} Ws,Wd	Wd = Ws - 2	1	1	C,DC,N,V,Z	1110 1001 1Bqq qddd dppp ssss
DISI #lit14	Disable interrupts (through L6) for #lit14+1 cycles	1	1	none	1111 1100 00kk kkkk kkkk kkkk
DIV.S{W} Wm,Wn	Signed 16/16-bit divide	1	18	C,N,V,Z	1101 1000 0ttt tvvv vW00 ssss
DIV.SD Wm, Wn	Signed 32/16-bit Integer Divide	1	18	C,N,V,Z	1101 1000 0ttt tvvv vW00 ssss
DIV.U{W} Wm,Wn	Unsigned 16/16-bit Integer Divide	1	18	C,N,V,Z	1101 1000 1ttt tvvv vW00 ssss
DIV.UD WM,Wn	Unsigned 32/16-bit Integer Divide	1	18	C,N,V,Z	1101 1000 1ttt tvvv vW00 ssss
EXCH Wns,Wnd	Swap Wns with Wnd	1	1	none	1111 1101 0000 0ddd d000 ssss
FBCL Ws,Wnd	Find Bit Change from Left (MSb)	1	1	C	1101 1111 0000 0ddd dppp ssss
FF1L Ws,Wnd	Find First One from Left (MSb)	1	1	C	1100 1111 1000 0ddd dppp ssss
FF1R Ws,Wnd	Find First One from Right (LSb)	1	1	C	1100 1111 0000 0ddd dppp ssss
GOTO Expr	Go to address	2	2	none	0000 0100 nnnn nnnn nnnn nnn0 0000 0000 0000 0000 0nnn nnnn
GOTO Wn	Go to indirect	1	2	none	0000 0001 0100 0000 0000 ssss
INC{.B} f	f = f + 1	1	1	C,DC,N,V,Z	1110 1100 0BDf ffff ffff ffff
INC{.B} f,WREG	WREG = f + 1	1	1	C,DC,N,V,Z	1110 1100 0BDf ffff ffff ffff
INC{.B} Ws,Wd	Wd = Ws + 1	1	1	C,DC,N,V,Z	1110 1000 0Bqq qddd dppp ssss
INC2{.B} f	f = f + 2	1	1	C,DC,N,V,Z	1110 1100 1BDf ffff ffff ffff
INC2{.B} f,WREG	WREG = f + 2	1	1	C,DC,N,V,Z	1110 1100 1BDf ffff ffff ffff
INC2{.B} Ws,Wd	Wd = Ws + 2	1	1	C,DC,N,V,Z	1110 1000 1Bqq qddd dppp ssss
IOR{.B} f	f = f | WREG	1	1	N,Z	1011 0111 0BDf ffff ffff ffff
IOR{.B} f,WREG	WREG = f | WREG	1	1	N,Z	1011 0111 0BDf ffff ffff ffff
IOR{.B} #lit10,Wn	Wn = lit10 | Wn	1	1	N,Z	1011 0011 0Bkk kkkk kkkk dddd
IOR{.B} Wb,Ws,Wd	Wd = Wb | Ws	1	1	N,Z	0111 0www wBqq qddd dppp ssss
IOR{.B} Wb,#lit5,Wd	Wd = Wb | #lit5	1	1	N,Z	0111 0www wBqq qddd d11k kkkk
LNK #lit14	Link Frame Pointer	1	1	none	1111 1010 00kk kkkk kkkk kkk0

Instr	Descr	#W	#Cyc	Status	Machine Code
LSR{.B} f	f = Logical >> f by 1	1	1	N,Z,C	1101 0101 0BDf ffff ffff ffff
LSR{.B} f,WREG	WREG = Logical >> f by 1	1	1	N,Z,C	1101 0101 0BDf ffff ffff ffff
LSR{.B} Ws,Wd	Wd = Logical >> Ws by 1	1	1	N,Z,C	1101 0001 0Bqq qddd dppp ssss
LSR Wb,Ws,Wd	Wd = Logical >> Wb by Ws	1	1	N,Z	1101 1110 0www wddd d000 ssss
LSR Wb,#lit4,Wd	Wd = Logical >> Wb by lit4	1	1	N,Z	1101 1110 0www wddd d100 kkkk
MOV f,Wnd	Move f to Wnd	1	1	none	1000 0fff ffff ffff ffff dddd
MOV{.B} f	Move f to f	1	1	N,Z	1011 1111 1BDf ffff ffff ffff
MOV{.B} f,WREG	Move f to WREG	1	1	N,Z	1011 1111 1BDf ffff ffff ffff
MOV #lit16,Wn	Move 16-bit literal to Wn	1	1	none	0010 kkkk kkkk kkkk kkkk dddd
MOV.B #lit8,Wn	Move 8-bit literal to Wn	1	1	none	1011 0011 1100 kkkk kkkk dddd
MOV Wns,f	Move Wns to f	1	1	none	1000 1fff ffff ffff ffff ssss
MOV{.B} Wso,Wdo	Move Wso to Wdo	1	1	none	0111 1www wBhh hddd dggg ssss
MOV{.B} [Wns+#slit10],Wnd	Move [Wns with offset] to Wnd	1	1	none	1001 0kkk kBkk kddd dkkk ssss
MOV{.B} Wns,[Wnd+#slit10]	Move Wns to [Wnd with offset]	1	1	none	1001 1kkk kBkk kddd dkkk ssss
MOV{.B} WREG,f	Move WREG to f	1	1	none	1011 0111 1B1f ffff ffff ffff
MOV.D Wns,Wd	Move double from Wns,Wns+1 to Wd	1	2	none	1011 1110 10qq qddd d000 sss0
MOV.D Ws,Wnd	Move double from Ws to Wnd,Wnd+1	1	2	none	1011 1110 0000 0ddd 0ppp ssss
MUL.SS Wb,Ws,Wnd	Wnd+1:Wnd = sign(Wb)*sign(Ws)	1	1	none	1011 1001 1www wddd dppp ssss
MUL.SU Wb,Ws,Wnd	Wnd+1:Wnd = sign(Wb)*unsign(Ws)	1	1	none	1011 1001 0www wddd dppp ssss
MUL.US Wb,Ws,Wnd	Wnd+1:Wnd = unsign(Wb)*sign(Ws)	1	1	none	1011 1000 1www wddd dppp ssss
MUL.UU Wb,Ws,Wnd	Wnd+1:Wnd = unsign(Wb)*unsign(Ws)	1	1	none	1011 1000 0www wddd dppp ssss
MUL.SU Wb,#lit5,Wnd	Wnd+1:Wnd = sign(Wb)*unsign(lit5)	1	1	none	1011 1001 0www wddd d11k kkkk
MUL.UU Wb,#lit5,Wnd	Wnd+1:Wnd = unsign(Wb)*unsign(lit5)	1	1	none	1011 1000 0www wddd d11k kkkk
MUL{.B} f	W3:W2 = f*WREG (unsigned mult)	1	1	none	1011 1100 0B0f ffff ffff ffff
NEG{.B} f	f = ~f + 1	1	1	C,DC,N,V,Z	1110 1110 0BDf ffff ffff ffff
NEG{.B} f,WREG	WREG = ~f + 1	1	1	C,DC,N,V,Z	1110 1110 0BDf ffff ffff ffff
NEG{.B} Ws,Wd	Wd = ~Ws + 1	1	1	C,DC,N,V,Z	1110 1010 0Bqq qddd dppp ssss

Instr	Descr	#W	#Cyc	Status	Machine Code
NOP	No operation	1	1	none	0000 0000 xxxx xxxx xxxx xxxx
NOPR	No operation	1	1	none	1111 1111 xxxx xxxx xxxx xxxx
POP f	Pop f from top-of-stack	1	1	none	1111 1001 ffff ffff ffff fff0
POP Wdo	Pop from top-of-stack to Wdo	1	1	none	0111 1www w0hh hddd d100 1111
POP.D Wnd	Pop from top-of-stack to Wnd,Wnd+1	1	2	none	1011 1110 0000 0ddd 0100 1111
POP.S	Pop shadow registers	1	1	C,DC,N,V,Z	1111 1110 1000 0000 0000 0000
PUSH f	Push f to top-of-stack	1	1	none	1111 1000 ffff ffff ffff fff0
PUSH Wso	Push Wso to top-of-stack	1	1	none	0111 1www w001 1111 1ggg ssss
PUSH.D Wns	Push Wns,Wns+1 to top-of-stack	1	2	none	1011 1110 1001 1111 1000 sss0
PUSH.S	Push shadow registers	1	1	none	1111 1110 1010 0000 0000 0000
PWRSAV #lit1	Go into Sleep (#lit1 = 0) or Idle (#lit1 = 1) mode	1	1	none	1111 1110 0100 0000 0000 000k
RCALL Expr	Relative Call	1	2	none	0000 0111 nnnn nnnn nnnn nnnn
RCALL Wn	Computed Call	1	1	none	0000 0001 0010 0000 0000 ssss
REPEAT #lit4	Repeat next instr. lit14+1 times	1	1	none	0000 1001 00kk kkkk kkkk kkkk
REPEAT Wn	Repeat next instr. (Wn)+1 times	1	1	none	0000 1001 1000 0000 0000 ssss
RESET	Software reset	1	1	all	1111 1110 0000 0000 0000 0000
RETFIE	Return from interrupt	1	3 (2)	IPL<3:0>, N,V,Z,C	0000 0110 0100 0000 0000 0000
RETLW{.B} #lit10,Wn	Return with unsigned lit10 in Wn	1	3 (2)	none	0000 0101 0Bkk kkkk kkkk dddd
RETURN	Return from subroutine	1	3 (2)	none	0000 0110 0000 0000 0000 0000
RLC{.B} f	f = Rotate << through Carry f	1	1	N,Z,C	1101 0110 1BDf ffff ffff ffff
RLC{.B} f,WREG	WREG = Rotate << through Carry f	1	1	N,Z,C	1101 0110 1BDf ffff ffff ffff
RLC{.B} Ws,Wd	Wd = Rotate << through Carry Ws	1	1	N,Z,C	1101 0010 1Bqq qddd dppp ssss
RLNC{.B} f	f = Rotate << (no Carry) f	1	1	N,Z	1101 0110 0BDf ffff ffff ffff
RLNC{.B} f,WREG	WREG = Rotate << (no Carry) f	1	1	N,Z	1101 0110 0BDf ffff ffff ffff
RLNC{.B} Ws,Wd	Wd = Rotate << (no Carry) Ws	1	1	N,Z	1101 0010 0Bqq qddd dppp ssss
RRC{.B} f	f = Rotate >> through Carry f	1	1	N,Z,C	1101 0111 1BDf ffff ffff ffff
RRC{.B} f,WREG	WREG = Rotate >> through Carry f	1	1	N,Z,C	1101 0111 1BDf ffff ffff ffff

Instr	Descr	#W	#Cyc	Status	Machine Code
RRC{.B} Ws,Wd	Wd = Rotate >> through Carry Ws	1	1	N,Z,C	1101 0011 1Bqq qddd dppp ssss
RRNC{.B} f	f = Rotate >> (no Carry) f	1	1	N,Z	1101 0111 0BDf ffff ffff ffff
RRNC{.B} f,WREG	WREG = Rotate >> (no Carry) f	1	1	N,Z	1101 0111 0BDf ffff ffff ffff
RRNC{.B} Ws,Wd	Wd = Rotate >> (no Carry) Ws	1	1	N,Z	1101 0011 0Bqq qddd dppp ssss
SE Ws,Wnd	Wnd = sign extend Ws	1	1	C,N,Z	1111 1011 0000 0ddd dppp ssss
SETM{.B} f	f = 0xFFFF	1	1	none	1110 1111 1BDf ffff ffff ffff
SETM{.B} WREG	WREG = 0xFFFF	1	1	none	1110 1111 1BDf ffff ffff ffff
SETM{.B} Wd	Wd = 0xFFFF	1	1	none	1110 1011 1Bqq qddd d000 0000
SL{.B} f	f = left shift f by 1 (into C)	1	1	C,N,Z	1101 0100 0BDf ffff ffff ffff
SL{.B} Ws,Wd	Wd = left shift Ws by 1 (into C)	1	1	C,N,Z	1101 0000 0Bqq qddd dppp ssss
SL{.B} f,WREG	WREG = left shift f by 1 (into C)	1	1	C,N,Z	1101 0100 0BDf ffff ffff ffff
SL Wb,#lit4,Wnd	Wnd = left shift by lit4 (no C)	1	1	N,Z	1101 1101 0www wddd d100 kkkk
SL Wb,Ws,Wd	Wd = left shift by Ws (no C)	1	1	N,Z	1101 1101 0www wddd d000 ssss
SUB{.B} f	f = f - WREG	1	1	C,DC,N,V,Z	1011 0101 0BDf ffff ffff ffff
SUB{.B} f,WREG	WREG = f - WREG	1	1	C,DC,N,V,Z	1011 0101 0BDf ffff ffff ffff
SUB{.B} #lit10,Wd	Wd = Wd – lit10	1	1	C,DC,N,V,Z	1011 0001 0Bkk kkkk kkkk dddd
SUB{.B} Wb,Ws,Wd	Wd = Wb - Ws	1	1	C,DC,N,V,Z	0101 0www wBqq qddd dppp ssss
SUB{.B} Wb,#lit5,Wd	Wd = Wb – lit5	1	1	C,DC,N,V,Z	0101 0www wBqq qddd d11k kkkk
SUBB{.B} f	f = f – WREG - BORROW (~C)	1	1	C,DC,N,V,Z	1011 0101 1BDf ffff ffff ffff
SUBB{.B} f,WREG	WREG = f – WREG - BORROW (~C)	1	1	C,DC,N,V,Z	1011 0101 1BDf ffff ffff ffff
SUBB{.B} #lit10,Wn	Wn = Wn – lit10 - BORROW (~C)	1	1	C,DC,N,V,Z	1011 0001 1Bkk kkkk kkkk dddd
SUBB{.B} Wb,Ws,Wd	Wd = Wb - Ws - BORROW (~C)	1	1	C,DC,N,V,Z	0101 1www wBqq qddd dppp ssss
SUBB{.B} Wb,#lit5,Wd	Wd = Wb – lit5 - BORROW (~C)	1	1	C,DC,N,V,Z	0101 1www wBqq qddd d11k kkkk
SUBR{.B} f	f = WREG - f	1	1	C,DC,N,V,Z	1011 1101 0BDf ffff ffff ffff
SUBR{.B} f,WREG	WREG = WREG - f	1	1	C,DC,N,V,Z	1011 1101 0BDf ffff ffff ffff
SUBR{.B} Wb,Ws,Wd	Wd = Ws - Wb	1	1	C,DC,N,V,Z	0001 0www wBqq qddd dppp ssss
SUBR{.B} Wb,#lit5,Wd	Wd = lit5 - Wb	1	1	C,DC,N,V,Z	0001 0www wBqq qddd d11k kkkk
SUBBR{.B} f	f = WREG - f - BORROW (~C)	1	1	C,DC,N,V,Z	1011 1101 1BDf ffff ffff ffff

Instr	Descr	#W	#Cyc	Status	Machine Code
SUBBR{.B} f,WREG	WREG = WREG - f - BORROW (~C)	1	1	C,DC,N,V,Z	1011 1101 1BDf ffff ffff ffff
SUBBR{.B} Wb,Ws,Wd	Wd = Ws - Wb - BORROW (~C)	1	1	C,DC,N,V,Z	0001 1www wBqq qddd dppp ssss
SUBBR{.B} Wb,#lit5,Wd	Wd = lit5 - Wb - BORROW (~C)	1	1	C,DC,N,V,Z	0001 1www wBqq qddd d11k kkkk
SWAP.B Wn	Wn = nibble swap Wn	1	1	none	1111 1101 1100 0000 0000 ssss
SWAP Wn	Wn = byte swap Wn	1	1	none	1111 1101 1000 0000 0000 ssss
TBLRDH{.B} Wsi,Wd	Read Prog<23:16> to Wd<7:0>	1	2	none	1011 1010 1Bqq qddd dppp ssss
TBLRDL{.B} Wsi,Wd	Read Prog<15:0> to Wd	1	2	none	1011 1010 0Bqq qddd dppp ssss
TBLWTH{.B} Ws,Wdi	Write Ws<7:0> to Prog<23:16>	1	2	none	1011 1011 1Bqq qddd dppp ssss
TBLWTL{.B} Ws,Wdi	Write Ws to Prog<15:0>	1	2	none	1011 1011 0Bqq qddd dppp ssss
ULNK	Unlink frame pointer	1	1	none	1111 1010 1000 0000 0000 0000
XOR{.B} f	f = f ^ WREG	1	1	N,Z	1011 0110 1BDf ffff ffff ffff
XOR{.B} f,WREG	WREG = f ^ WREG	1	1	N,Z	1011 0110 1BDf ffff ffff ffff
XOR{.B} #lit10,Wn	Wn = lit10 ^ Wn	1	1	N,Z	1011 0010 1Bkk kkkk kkkk dddd
XOR{.B} Wb,Ws,Wd	Wd = Wb ^ Ws	1	1	N,Z	0110 1www wBqq qddd dppp ssss
XOR{.B} Wb,#lit5,Wd	Wd = Wb ^ lit5	1	1	N,Z	0110 1www wBqq qddd d11k kkkk
ZE Ws,Wnd	Wnd = Zero extend Ws	1	1	N,Z,C	1111 1011 10qq qddd dppp ssss