2.5.4 ALU Instruction Summary

Instruction	*ASTAT Status Flags*								*STKY Status Flags*			
Fixed-point:	AZ	AV	AN	AC	AS	AI	AF	CACC	AUS	AVS	AOS	AIS
c Rn = Rx + Ry	*	*	*	*	0	0	0	–	–	–	**	–
c Rn = Rx – Ry	*	*	*	*	0	0	0	–	–	–	**	–
c Rn = Rx + Ry + CI	*	*	*	*	0	0	0	–	–	–	**	–
c Rn = Rx – Ry + CI – 1	*	*	*	*	0	0	0	–	–	–	**	–
Rn = (Rx + Ry)/2	*	0	*	*	0	0	0	–	–	–	–	–
COMP(Rx, Ry)	*	0	*	0	0	0	0	*	–	–	–	–
Rn = Rx + CI	*	*	*	*	0	0	0	–	–	–	**	–
Rn = Rx + CI – 1	*	*	*	*	0	0	0	–	–	–	**	–
Rn = Rx + 1	*	*	*	*	0	0	0	–	–	–	**	–
Rn = Rx – 1	*	*	*	*	0	0	0	–	–	–	**	–
c Rn = –Rx	*	*	*	*	0	0	0	–	–	–	**	–
c Rn = ABS Rx	*	*	0	0	*	0	0	–	–	–	**	–
Rn = PASS Rx	*	0	*	0	0	0	0	–	–	–	–	–
c Rn = Rx AND Ry	*	0	*	0	0	0	0	–	–	–	–	–
c Rn = Rx OR Ry	*	0	*	0	0	0	0	–	–	–	–	–
c Rn = Rx XOR Ry	*	0	*	0	0	0	0	–	–	–	–	–
c Rn = NOT Rx	*	0	*	0	0	0	0	–	–	–	–	–
Rn = MIN(Rx, Ry)	*	0	*	0	0	0	0	–	–	–	–	–
Rn = MAX(Rx, Ry)	*	0	*	0	0	0	0	–	–	–	–	–
Rn = CLIP Rx BY Ry	*	0	*	0	0	0	0	–	–	–	–	–
Floating–point:												
Fn = Fx + Fy	*	*	*	0	0	*	1	–	**	**	–	**
Fn = Fx – Fy	*	*	*	0	0	*	1	–	**	**	–	**
Fn = ABS (Fx + Fy)	*	*	0	0	0	*	1	–	**	**	–	**
Fn = ABS (Fx – Fy)	*	*	0	0	0	*	1	–	**	**	–	**
Fn = (Fx + Fy)/2	*	0	*	0	0	*	1	–	**	–	–	**
COMP(Fx, Fy)	*	0	*	0	0	*	1	*	–	–	–	**
Fn = –Fx	*	*	*	0	0	*	1	–	–	**	–	**
Fn = ABS Fx	*	*	0	0	*	*	1	–	–	**	–	**
Fn = PASS Fx	*	0	*	0	0	*	1	–	–	–	–	**
Fn = RND Fx	*	*	*	0	0	*	1	–	–	**	–	**
Fn = SCALB Fx BY Ry	*	*	*	0	0	*	1	–	**	**	–	**
Rn = MANT Fx	*	*	0	0	*	*	1	–	–	**	–	**
Rn = LOGB Fx	*	*	*	0	0	*	1	–	–	**	–	**
Rn = FIX Fx BY Ry	*	*	*	0	0	*	1	–	**	**	–	**
Rn = FIX Fx	*	*	*	0	0	*	1	–	**	**	–	**
Fn = FLOAT Rx BY Ry	*	*	*	0	0	0	1	–	**	**	–	–
Fn = FLOAT Rx	*	0	*	0	0	0	1	–	–	–	–	–
Fn = RECIPS Fx	*	*	*	0	0	*	1	–	**	**	–	**
Fn = RSQRTS Fx	*	*	*	0	0	*	1	–	–	**	–	**
Fn = Fx COPYSIGN Fy	*	0	*	0	0	*	1	–	–	–	–	**
Fn = MIN(Fx, Fy)	*	0	*	0	0	*	1	–	–	–	–	**
Fn = MAX(Fx, Fy)	*	0	*	0	0	*	1	–	–	–	–	**
Fn = CLIP Fx BY Fy	*	0	*	0	0	*	1	–	–	–	–	**

Rn, Rx, Ry = Any register file location; treated as fixed-point
Fn, Fx, Fy = Any register file location; treated as floating-point
c = ADSP-21xx-compatible instruction

* set or cleared, depending on results of instruction
** may be set (but not cleared), depending on results of instruction
– no effect

2.6.6 Multiplier Instruction Summary

Instruction	*ASTAT Flags*				*STKY Flags*			
	MU	MN	MV	MI	MUS	MOS	MVS	MIS
Fixed-Point:								
Rn / MRF / MRB = Rx * Ry (S/U, S/U, F/I/FR)	*	*	*	0	–	**	–	–
Rn / MRF / MRB = MRF / MRB + Rx * Ry (S/U, S/U, F/I/FR)	*	*	*	0	–	**	–	–
Rn / MRF / MRB = MRF / MRB – Rx * Ry (S/U, S/U, F/I/FR)	*	*	*	0	–	**	–	–
Rn / MRF = SAT MRF; Rn / MRB = SAT MRB (SI, UI, SF, UF)	*	*	*	0	–	**	–	–
Rn / MRF = RND MRF; Rn / MRB = RND MRB (SF, UF)	*	*	*	0	–	**	–	–
MRF / MRB = 0	0	0	0	0	–	–	–	–
MRxF / MRxB = Rn	0	0	0	0	–	–	–	–
Rn = MRxF / MRxB	0	0	0	0	–	–	–	–
Floating-Point:								
Fn = Fx * Fy	*	*	*	*	**	–	**	**

Note: For floating-point multiply/accumulates, see "Multifunction Computations"

* set or cleared, depending on results of instruction
** may be set (but not cleared), depending on results of instruction
– no effect

Rn, Rx, Ry = R15-R0; register file location, treated as fixed-point
Fn, Fx, Fy = F15-F0; register file location, treated as floating-point
MRxF = MR2F, MR1F, MR0F; multiplier result accumulators, foreground
MRxB = MR2B, MR1B, MR0B; multiplier result accumulators, background

Dual Add/Subtract

Ra = Rx + Ry , Rs = Rx – Ry
Fa = Fx + Fy , Fs = Fx – Fy

Fixed-Point Multiply/Accumulate and Add, Subtract or Average

Rm=R3-0 * R7-4 (SSFR)	,	Ra=R11-8 + R15-12
MRF=MRF + R3-0 * R7-4 (SSF)	,	Ra=R11-8 – R15-12
Rm=MRF + R3-0 * R7-4 (SSFR)	,	Ra=(R11-8 + R15-12)/2
MRF=MRF – R3-0 * R7-4 (SSF)	,	
Rm=MRF – R3-0 * R7-4 (SSFR)	,	

Floating-Point Multiplication and ALU Operation

Fm=F3-0 * F7-4 ,

Fa=F11-8 + F15-12
Fa=F11-8 – F15-12
Fa=FLOAT R11-8 by R15-12
Ra=FIX F11-8 by R15-12
Fa=(F11-8 + F15-12)/2
Fa=ABS F11-8
Fa=MAX (F11-8, F15-12)
Fa=MIN (F11-8, F15-12)

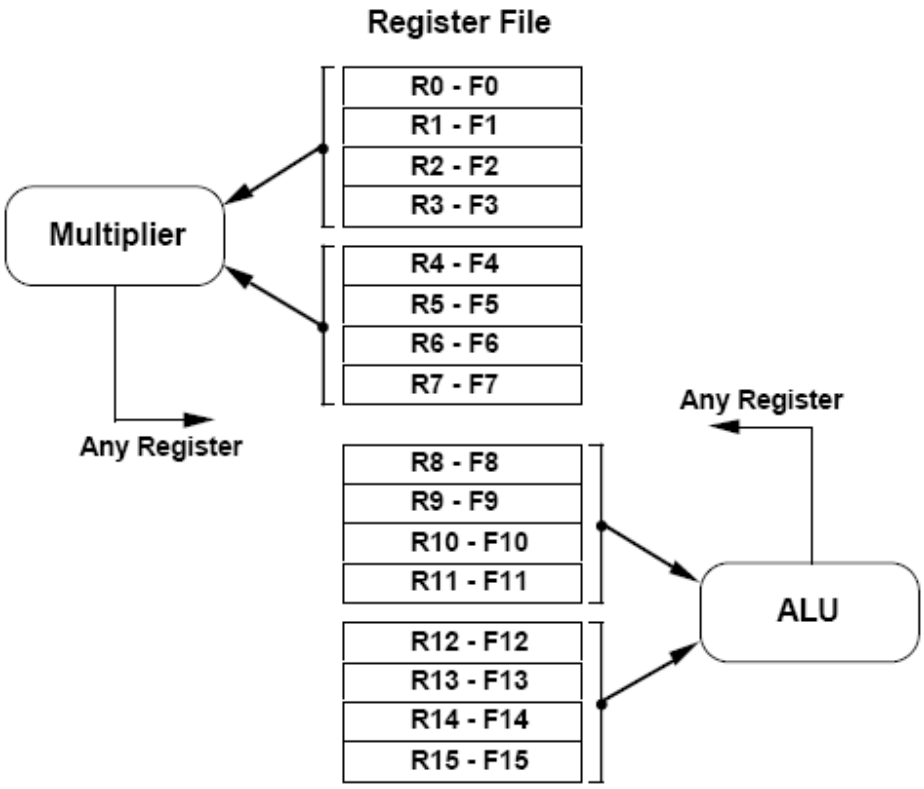
Multiplication and Dual Add/Subtract

Rm = R3-0 * R7-4 (SSFR) , Ra = R11-8 + R15-12 , Rs = R11-8 – R15-12
Fm = F3-0 * F7-4 , Fa = F11-8 + F15-12 , Fs = F11-8 – F15-12

Rm, Ra, Rs, Rx, Ry –Any register file location; fixed-point
Fm, Fa, Fs, Fx, Fy –Any register file location; floating-point

R3-0	–R3, R2, R1, R0	F3-0	–F3, F2, F1, F0
R7-4	–R7, R6, R5, R4	F7-4	–F7, F6, F5, F4
R11-8	–R11, R10, R9, R8	F11-8	–F11, F10, F9, F8
R15-12	–R15, R14, R13, R12	F15-12	–F15, F14, F13, F12

SSFR –X-input signed, Y-input signed, Fractional input, Rounded-to-nearest output
SSF –X-input signed, Y-input signed, Fractional input


Register File
R0 - F0
R1 - F1
R2 - F2
R3 - F3
R4 - F4
R5 - F5
R6 - F6
R7 - F7
Multiplier
Any Register
Any Register
R8 - F8
R9 - F9
R10 - F10
R11 - F11
R12 - F12
R13 - F13
R14 - F14
R15 - F15
ALU

Compute & Move or Modify Instructions

1. *compute,* | DM(Ia,Mb) = dreg1 / dreg1 = DM(Ia,Mb) | , | PM(Ic,Md) = dreg2 / dreg2 = PM(Ic,Md) | ;

2. *IF condition* compute;

3a. *IF condition* *compute,* | DM(Ia,Mb) / PM(Ic,Md) | = ureg ;

3b. *IF condition* *compute,* | DM(Mb,Ia) / PM(Md,Ic) | = ureg ;

3c. *IF condition* *compute,* ureg = | DM(Ia,Mb) / PM(Ic,Md) | ;

3d. *IF condition* *compute,* ureg = | DM(Mb,Ia) / PM(Md,Ic) | ;

4a. *IF condition* *compute,* | DM(Ia,<data6>) / PM(Ic,<data6>) | = dreg ;

4b. *IF condition* *compute,* | DM(<data6>,Ia) / PM(<data6>,Ic) | = dreg ;

4c. *IF condition* *compute,* dreg = | DM(Ia,<data6>) / PM(Ic,<data6>) | ;

4d. *IF condition* *compute,* dreg = | DM(<data6>,Ia) / PM(<data6>,Ic) | ;

5. *IF condition* *compute,* ureg1 = ureg2 ;

6a. *IF condition* shiftimm, | *DM(Ia,Mb)* / *PM(Ic,Md)* | = *dreg* ;

6b. *IF condition* shiftimm, *dreg* = | *DM(Ia,Mb)* / *PM(Ic,Md)* | ;

7. *IF condition* *compute,* MODIFY | (Ia,Mb) / (Ic,Md) | ;