Appendix B: Floating-point Instructions (Optional on Cortex-M4 and Cortex-M7)

Instruction	Operands	Description and Action
VABS.F32	Sd, Sm	Absolute value of floats, Sd ← |Sm|
VADD.F32	{Sd,} Sn, Sm	Add floating points, Sd ← Sn + Sm
VCMP.F32	Sd, <Sm | #0.0>	Compare two floating-point registers, or one floating-point register and zero
VCMPE.F32	Sd, <Sm | #0.0>	Compare two floating-point registers, or one floating-point register and zero, and raise exception for a signaling NaN
VCVT{R}.S32.F32	Sd, Sm	Convert from single-precision to signed 32-bit (S32) or unsigned 32-bit (U32) integer. If R is specified, it uses the rounding mode specified by FPSCR. If R is omitted, it uses round towards zero.
VCVT{R}.U32.F32	Sd, Sm	
VCVT{R}.F32.S32	Sd, Sm	Convert to single-precision from signed 32-bit (S32) or unsigned 32-bit (U32) integer. See above for R.
VCVT{R}.F32.U32	Sd, Sm	
VCVT{R}.Td.F32	Sd, Sm, #fbits	Convert between single-precision and fixed-point. Td can be S16 (signed 16-bit), U16 (unsigned 16-bit), S32 (signed 32-bit), U32 (unsigned 32-bit). fbits is the number of fraction bits in the fixed-point number. See above for R.
VCVT{R}.Td.F32	Sd, Sd, #fbits	
VCVT{R}.F32.Td	Sd, Sm, #fbits	
VCVT{R}.F32.Td	Sd, Sd, #fbits	
VCVT<B|T>.F32.F16	Sd, Sm	Converts half-precision float to single-precision (B = bottom half of Sm, T = top half of Sm)
VCVT<B|T>.F16.F32	Sd, Sm	Converts single-precision float to half-precision (B = bottom half of Sd, T = top half of Sd)
VDIV.F32	{Sd,} Sn, Sm	Divide single-precision floats, Sd = Sn/Sm
VFMA.F32	{Sd,} Sn, Sm	Multiply (fused) then accumulate float, Sd = Sd + Sn*Sm
VFMS.F32	{Sd,} Sn, Sm	Multiply (fused) then subtract float, Sd = Sd - Sn*Sm
VFNMA.F32	{Sd,} Sn, Sm	Multiply (fused) then accumulate then negate float, Sd = -1 * Sd + Sn * Sm
VFNMS.F32	{Sd,} Sn, Sm	Multiply (fused) then subtract then negate float, Sd = -1 * Sd - Sn * Sm
VLDM.64	Rn{!}, list	Load multiple double-precision floats
VLDM.32	Rn{!}, list	Load multiple single-precision floats
VLDR.F64	<Dd|Sd>, [Rn]	Load one double-precision float
VLDR.F32	<Dd|Sd>, [Rn]	Load one single-precision float
VLMA.F32	{Sd,} Sn, Sm	Multiply float then accumulate float, Sd = Sd + Sn*Sm
VLMS.F32	{Sd,} Sn, Sm	Multiply float then subtract float, Sd = Sd - Sn*Sm
VMOV.F32	Sd, #imm	Move immediate to float-register
VMOV	Sd, Sm	Copy from float register to float register
VMOV	Sn, Rt	Copy ARM core register to float register
VMOV	Sm, Sm1, Rt, Rt2	Copy 2 ARM core registers to 2 float registers
VMOV	Dd[x], Rt	Copy ARM core register to a half of a double-precision floating-point register, where x is 0 or 1.
VMOV	Rt, Dn[x]	Copy a half of a double-precision floating-point register to ARM core register, where x is 0 or 1.
VMRS	Rt, FPSCR	Move FPSCR to ARM core register or APSR
VMSR	FPSCR, Rt	Move to FPSCR from ARM Core register
VMUL.F32	{Sd,} Sn, Sm	Multiply float, Sd = Sn * Sm
VNEG.F32	Sd, Sm	Negate float, Sd = -1 * Sm
VNMLA.F32	Sd, Sn, Sm	Multiply float then accumulate then negate float Sd = -1 * (Sd + Sn * Sm)
VNMLS.F32	Sd, Sn, Sm	Multiply float then subtract then negate float Sd = -1 * (Sd - Sn * Sm)
VNMUL.F32	{Sd,} Sn, Sm	Negate and multiply float, Sd = -1 * Sn * Sm
VPOP.64	list	Pop double registers from stack

VPOP.32	list	Pop float registers from stack
VPUSH.64	list	Push double registers to stack
VPUSH.32	list	Push float registers to stack
VSQRT.F32	Sd, Sm	Square-root of float
VSTM.64	Rn{!}, list	Store multiple double registers
VSTM.32	Rn{!}, list	Store multiple float registers
VSTR.64	Sd, [Rn]	Store one double register
VSTR.32	Sd, [Rn]	Store one float registers
VSUB.F32	{Sd,} Sn, Sm	Subtract float, Sd = Sn - Sm