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

<table>
<tr><th>Instruction</th><th>Operands</th><th>Description and Action</th></tr>
<tr><td>VABS.F32</td><td>Sd, Sm</td><td>Absolute value of floats, Sd ← |Sm|</td></tr>
<tr><td>VADD.F32</td><td>{Sd,} Sn, Sm</td><td>Add floating points, Sd ← Sn + Sm</td></tr>
<tr><td>VCMP.F32</td><td>Sd, <Sm | #0.0></td><td>Compare two floating-point registers, or one floating-point register and zero</td></tr>
<tr><td>VCMPE.F32</td><td>Sd, <Sm | #0.0></td><td>Compare two floating-point registers, or one floating-point register and zero, and raise exception for a signaling NaN</td></tr>
<tr><td>VCVT{R}.S32.F32</td><td>Sd, Sm</td><td rowspan="2">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.</td></tr>
<tr><td>VCVT{R}.U32.F32</td><td>Sd, Sm</td></tr>
<tr><td>VCVT{R}.F32.S32</td><td>Sd, Sm</td><td rowspan="2">Convert to single-precision from signed 32-bit (S32) or unsigned 32-bit (U32) integer. See above for R.</td></tr>
<tr><td>VCVT{R}.F32.U32</td><td>Sd, Sm</td></tr>
<tr><td>VCVT{R}.Td.F32</td><td>Sd, Sm, #fbits</td><td rowspan="4">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.</td></tr>
<tr><td>VCVT{R}.Td.F32</td><td>Sd, Sd, #fbits</td></tr>
<tr><td>VCVT{R}.F32.Td</td><td>Sd, Sm, #fbits</td></tr>
<tr><td>VCVT{R}.F32.Td</td><td>Sd, Sd, #fbits</td></tr>
<tr><td>VCVT<B|T>.F32.F16</td><td>Sd, Sm</td><td>Converts half-precision float to single-precision
(B = bottom half of Sm, T = top half of Sm)</td></tr>
<tr><td>VCVT<B|T>.F16.F32</td><td>Sd, Sm</td><td>Converts single-precision float to half-precision
(B = bottom half of Sd, T = top half of Sd)</td></tr>
<tr><td>VDIV.F32</td><td>{Sd,} Sn, Sm</td><td>Divide single-precision floats, Sd = Sn/Sm</td></tr>
<tr><td>VFMA.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply (fused) then accumulate float, Sd = Sd + Sn*Sm</td></tr>
<tr><td>VFMS.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply (fused) then subtract float, Sd = Sd - Sn*Sm</td></tr>
<tr><td>VFNMA.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply (fused) then accumulate then negate float,
Sd = -1 * Sd + Sn * Sm</td></tr>
<tr><td>VFNMS.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply (fused) then subtract then negate float,
Sd = -1 * Sd - Sn * Sm</td></tr>
<tr><td>VLDM.64</td><td>Rn{!}, list</td><td>Load multiple double-precision floats</td></tr>
<tr><td>VLDM.32</td><td>Rn{!}, list</td><td>Load multiple single-precision floats</td></tr>
<tr><td>VLDR.F64</td><td><Dd|Sd>, [Rn]</td><td>Load one double-precision float</td></tr>
<tr><td>VLDR.F32</td><td><Dd|Sd>, [Rn]</td><td>Load one single-precision float</td></tr>
<tr><td>VLMA.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply float then accumulate float, Sd = Sd + Sn*Sm</td></tr>
<tr><td>VLMS.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply float then subtract float, Sd = Sd - Sn*Sm</td></tr>
<tr><td>VMOV.F32</td><td>Sd, #imm</td><td>Move immediate to float-register</td></tr>
<tr><td>VMOV</td><td>Sd, Sm</td><td>Copy from float register to float register</td></tr>
<tr><td>VMOV</td><td>Sn, Rt</td><td>Copy ARM core register to float register</td></tr>
<tr><td>VMOV</td><td>Sm, Sm1, Rt, Rt2</td><td>Copy 2 ARM core registers to 2 float registers</td></tr>
<tr><td>VMOV</td><td>Dd[x], Rt</td><td>Copy ARM core register to a half of a double-precision floating-point register, where x is 0 or 1.</td></tr>
<tr><td>VMOV</td><td>Rt, Dn[x]</td><td>Copy a half of a double-precision floating-point register to ARM core register, where x is 0 or 1.</td></tr>
<tr><td>VMRS</td><td>Rt, FPSCR</td><td>Move FPSCR to ARM core register or APSR</td></tr>
<tr><td>VMSR</td><td>FPSCR, Rt</td><td>Move to FPSCR from ARM Core register</td></tr>
<tr><td>VMUL.F32</td><td>{Sd,} Sn, Sm</td><td>Multiply float, Sd = Sn * Sm</td></tr>
<tr><td>VNEG.F32</td><td>Sd, Sm</td><td>Negate float, Sd = -1 * Sm</td></tr>
<tr><td>VNMLA.F32</td><td>Sd, Sn, Sm</td><td>Multiply float then accumulate then negate float
Sd = -1 * (Sd + Sn * Sm)</td></tr>
<tr><td>VNMLS.F32</td><td>Sd, Sn, Sm</td><td>Multiply float then subtract then negate float
Sd = -1 * (Sd - Sn * Sm)</td></tr>
<tr><td>VNMUL.F32</td><td>{Sd,} Sn, Sm</td><td>Negate and multiply float, Sd = -1 * Sn * Sm</td></tr>
<tr><td>VPOP.64</td><td>list</td><td>Pop double registers from stack</td></tr>
</table>

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