Vector Floating Point Instruction Set
Quick Reference Card

Key to Tables				
{cond}	See Table **Condition Field**		Fd, Fn, Fm	Sd, Sn, Sm (single precision), or Dd, Dn, Dm (double precision).
<S/D>	S (single precision) or D (double precision).		{E}	E : raise exception on any NaN. Without E : raise exception only on signaling NaNs.
<S/D/X>	As above, or X (unspecified precision).		{Z}	Round towards zero. Overrides FPSCR rounding mode.
<VFPsysreg>	FPSCR or FPSID.		<VFPregs>	A comma separated list of *consecutive* VFP registers, enclosed in braces ({ and }).
§	2: VFPv2 and above. 3: VFPv3 and above.		<BinLoc>	Number of integer bits in fixed-point number, 0-16 or 0-31.
			<type>	SH, SL, UH, or UL, for Signed or Unsigned, Halfword (16-bit) or Longword (32-bit).

Operation		§	Assembler	Exceptions	Action	Notes
Vector arithmetic	Multiply		FMUL<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX	Fd := Fn * Fm	
	and negate		FNMUL<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX	Fd := – (Fn * Fm)	
	and accumulate		FMAC<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX	Fd := Fd + (Fn * Fm)	
	negate and accumulate		FNMAC<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX	Fd := Fd – (Fn * Fm)	
	and subtract		FMSC<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX	Fd := – Fd + (Fn * Fm)	
	negate and subtract		FNMSC<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX	Fd := – Fd – (Fn * Fm)	
	Add		FADD<S/D>{cond} Fd, Fn, Fm	IO, OF, IX	Fd := Fn + Fm	
	Subtract		FSUB<S/D>{cond} Fd, Fn, Fm	IO, OF, IX	Fd := Fn – Fm	
	Divide		FDIV<S/D>{cond} Fd, Fn, Fm	IO, DZ, OF, UF, IX	Fd := Fn / Fm	
	Copy		FCPY<S/D>{cond} Fd, Fm		Fd := Fm	
	Absolute		FABS<S/D>{cond} Fd, Fm		Fd := abs(Fm)	
	Negative		FNEG<S/D>{cond} Fd, Fm		Fd := – Fm	
	Square root		FSQRT<S/D>{cond} Fd, Fm	IO, IX	Fd := sqrt(Fm)	
Scalar compare	Two values		FCMP{E}<S/D>{cond} Fd, Fm	IO	Set FPSCR flags on Fd – Fm	Use FMSTAT to transfer flags.
	Value with zero		FCMP{E}Z<S/D>{cond} Fd	IO	Set FPSCR flags on Fd – 0	Use FMSTAT to transfer flags.
Scalar convert	Single to double		FCVTDS{cond} Dd, Sm	IO	Dd := convertStoD(Sm)	
	Double to single		FCVTSD{cond} Sd, Dm	IO, OF, UF, IX	Sd := convertDtoS(Dm)	
	Unsigned integer to float		FUITO<S/D>{cond} Fd, Sm	IX	Fd := convertUItoF(Sm)	
	Signed integer to float		FSITO<S/D>{cond} Fd, Sm	IX	Fd := convertSItoF(Sm)	
	Fixed-point to float	3	F<type>TO<S/D>{<cond>} Fd, #<BinLoc>	IO, IX	Fd := convert<type>toF(Fd)	Source is in bottom 16 or 32 bits of Fd.
	Float to unsigned integer		FTOUI{Z}<S/D>{cond} Sd, Fm	IO, IX	Sd := convertFtoUI(Fm)	
	Float to signed integer		FTOSI{Z}<S/D>{cond} Sd, Fm	IO, IX	Sd := convertFtoSI(Fm)	
	Float to fixed-point	3	FTO<type><S/D>{<cond>} Fd, #<BinLoc>	IO, IX	Fd := convertFto<type>(Fd)	Destination is bottom 16 or 32 bits of Fd.
Insert constant	Insert constant in register	3	FCONST{cond} Fd, #<imm8>		Fd := convertIMtoF(imm8)	
Transfer registers	ARM® to single		MOV{cond} Sn, Rd		Sn := Rd	
	Single to ARM		MOV{cond} Rd, Sn		Rd := Sn	
	Two ARM to two singles	2	MOV{cond} {Sn,Sm}, Rd, Rn		Sn := Rd, Sm := Rn	Sm must be S(n+1)
	Two singles to two ARM	2	MOV{cond} Rd, Rn, {Sn,Sm}		Rd := Sn, Rn := Sm	Sm must be S(n+1)
	Two ARM to double	2	MOV{cond} Dn, Rd, Rn		Dn[31:0] := Rd, Dn[63:32] := Rn	
	Double to two ARM	2	MOV{cond} Rd, Rn, Dn		Rd := Dn[31:0], Rn := Dn[63:32]	
	ARM to lower half of double		MOV{cond} Dn[0], Rd		Dn[31:0] := Rd	
	Lower half of double to ARM		MOV{cond} Rd, Dn[0]		Rd := Dn[31:0]	
	ARM to upper half of double		MOV{cond} Dn[1], Rd		Dn[63:32] := Rd	
	Upper half of double to ARM		MOV{cond} Rd, Dn[1]		Rd := Dn[63:32]	
	ARM to VFP system register		MSR{cond} <VFPsysreg>, Rd		VFPsysreg := Rd	Stalls ARM until all VFP ops complete.
	VFP system register to ARM		MRS{cond} Rd, <VFPsysreg>		Rd := VFPsysreg	Stalls ARM until all VFP ops complete.
	FPSCR flags to CPSR		MRS{cond} R15, FPSCR		CPSR flags := FPSCR flags	

Vector Floating Point Instruction Set
Quick Reference Card

Operation		§	Assembler	Synonyms	Action
Save VFP registers	Single		`STR{cond} Fd, [Rn{, #<immed>}]`		[address] := Fd. Immediate range 0-1020, multiple of 4.
	Multiple, unindexed / increment after		`STM{cond} Rn{!}, <VFPregs>`	`STMIA, STMEA`	Saves list of VFP registers, starting at address in Rn.
	decrement before		`STMDB{cond} Rn!, <VFPregs>`	`STMFD` (full descending)	
	Push onto stack		`PUSH{cond} <VFPregs>`	`STMFD SP!`	
Load VFP registers	Single		`LDR{cond} Fd, [Rn{, #<immed>}]`		Fd := [address]. Immediate range 0-1020, multiple of 4.
	Multiple, unindexed / increment after		`LDM{cond} Rn{!}, <VFPregs>`	`LDMIA, LDMFD`	Loads list of VFP registers, starting at address in Rn.
	decrement before		`LDMDB{cond} Rn!, <VFPregs>`	`LDMEA` (empty ascending)	
	Pop from stack		`POP{cond} <VFPregs>`	`LDM SP!`	

FPSCR format								Rounding		(Stride – 1)*3			Vector length – 1					Exception trap enable bits							Cumulative exception bits				
31	30	29	28				24	23	22	21	20		18	17	16			12	11	10	9	8				4	3	2	1	0
N	Z	C	V				FZ	RMODE		STRIDE			LEN					IXE	UFE	OFE	DZE	IOE				IXC	UFC	OFC	DZC	IOC

FZ: 1 = flush to zero mode. Rounding: 0 = round to nearest, 1 = towards +∞, 2 = towards –∞, 3 = towards zero. (Vector length * Stride) must not exceed 4 for double precision operands.

Register banks, and sector, scalar and mixed operations
If Fd is S0-S7, D0-D3, or D16-D19, operation is Scalar (regardless of vector length).
If Fd is S8-S31, D4-D15, or D20-D31, and Fm is S8-S31, D4-D15, or D20-D31, operation is Vector.
If Fd is S8-S31, D4-D15, or D20-D31, and Fm is S0-S7, D0-D3, or D16-D31, operation is Mixed (Fm scalar, others vector).
S0-S7 (or D0-D3), S8-S15 (D4-D7), S16-S23 (D8-D11), S24-S31 (D12-D15), D16-D19, D20-D23, D24-D27, D28-D31, each form a circulating bank of registers.

Exceptions	
`IO`	Invalid operation
`OF`	Overflow
`UF`	Underflow
`IX`	Inexact result
`DZ`	Division by zero

Condition Field					
Mnemonic	**Description (VFP)**	**Description (ARM or Thumb®)**	**Mnemonic**	**Description (VFP)**	**Description (ARM or Thumb)**
`EQ`	Equal	Equal	`HI`	Greater than, or unordered	Unsigned higher
`NE`	Not equal, or unordered	Not equal	`LS`	Less than or equal	Unsigned lower or same
`CS / HS`	Greater than or equal, or unordered	Carry Set / Unsigned higher or same	`GE`	Greater than or equal	Signed greater than or equal
`CC / LO`	Less than	Carry Clear / Unsigned lower	`LT`	Less than, or unordered	Signed less than
`MI`	Less than	Negative	`GT`	Greater than	Signed greater than
`PL`	Greater than or equal, or unordered	Positive or zero	`LE`	Less than or equal, or unordered	Signed less than or equal
`VS`	Unordered (at least one NaN operand)	Overflow	`AL`	Always (normally omitted)	Always (normally omitted)
`VC`	Not unordered	No overflow			

Proprietary Notice

Document Number

ARM QRC 0007A Change Log

Issue	Date	Change
A	November 2004	First Release
B	May 2005	Release for RVCT 2.2 SP1
C	March 2006	Release for RVCT 3.0

www.arm.com