Vector Floating Point Instruction Set
Quick Reference Card

Key to Tables			
{C}	See Table **Condition Field**	<fpconst>	+/– *m* * 2^{-n} where *m* and *n* are integers, 16 <= *m* <= 31, 0 <= *n* <=7
<P>	F32 (single precision) or F64 (double precision).	Fd, Fn, Fm	Sd, Sn, Sm (single precision), or Dd, Dn, Dm (double precision).
S, D, H	Single, double, or half-precision (F16).	{E}	E : raise exception on any NaN. Without E : raise exception only on signaling NaNs.
F	Single or double-precision floating point.	{R}	Use FPSCR rounding mode. Otherwise, round towards zero.
SI, UI	Signed or unsigned integer.	<VFPregs>	A comma separated list of *consecutive* VFP registers, enclosed in braces ({ and }).
<VFPsysreg>	FPSCR or FPSID.	<fbits>	Number of fraction bits in fixed-point number, 0-16 or 1-32.
§	2: VFPv2 and above. 3: VFPv3 and above. 3H: VFPv3 and above with half-precision extension.	<type>	S16, S32, U16, or U32, for Signed or Unsigned, 16-bit or 32-bit.

Operation		§	Assembler	Exceptions	Action	Notes
Vector arithmetic	Multiply		VMUL{C}.<P> Fd, Fn, Fm	IO, OF, UF, IX	Fd := Fn * Fm	
	and negate		VNMUL{C}.<P> Fd, Fn, Fm	IO, OF, UF, IX	Fd := – (Fn * Fm)	
	and accumulate		VMLA{C}.<P> Fd, Fn, Fm	IO, OF, UF, IX	Fd := Fd + (Fn * Fm)	
	negate and accumulate		VMLS{C}.<P> Fd, Fn, Fm	IO, OF, UF, IX	Fd := Fd – (Fn * Fm)	
	and subtract		VNMLS{C}.<P> Fd, Fn, Fm	IO, OF, UF, IX	Fd := – Fd + (Fn * Fm)	
	negate and subtract		VNMLA{C}.<P> Fd, Fn, Fm	IO, OF, UF, IX	Fd := – Fd – (Fn * Fm)	
	Add		VADD{C}.<P> Fd, Fn, Fm	IO, OF, IX	Fd := Fn + Fm	
	Subtract		VSUB{C}.<P> Fd, Fn, Fm	IO, OF, IX	Fd := Fn – Fm	
	Divide		VDIV{C}.<P> Fd, Fn, Fm	IO, DZ, OF, UF, IX	Fd := Fn / Fm	
	Absolute		VABS{C}.<P> Fd, Fm		Fd := abs(Fm)	
	Negative		VNEG{C}.<P> Fd, Fm		Fd := – Fm	
	Square root		VSQRT{C}.<P> Fd, Fm	IO, IX	Fd := sqrt(Fm)	
Scalar compare	Two values		VCMP{E}{C}.<P> Fd, Fm	IO	Set FPSCR flags on Fd – Fm	Use VMRS APSR_nzcv, FPSCR to transfer flags.
	Value with zero		VCMP{E}{C}.<P> Fd, #0.0	IO	Set FPSCR flags on Fd – 0	
Scalar convert	Single to double		VCVT{C}.F64.F32 Dd, Sm	IO	Dd := convertStoD(Sm)	
	Double to single		VCVT{C}.F32.F64 Sd, Dm	IO, OF, UF, IX	Sd := convertDtoS(Dm)	
	Unsigned integer to float		VCVT{C}.<P>.U32 Fd, Sm	IX	Fd := convertUItoF(Sm)	
	Signed integer to float		VCVT{C}.<P>.S32 Fd, Sm	IX	Fd := convertSItoF(Sm)	
	Float to unsigned integer		VCVT{R}{C}.U32.<P> Sd, Fm	IO, IX	Sd := convertFtoUI(Fm)	
	Float to signed integer		VCVT{R}{C}.S32.<P> Sd, Fm	IO, IX	Sd := convertFtoSI(Fm)	
	Fixed-point to float	3	VCVT{C}.<P>.<type> Fd, Fd, #<fbits>	IO, IX	Fd := convert<type>toF(Fd)	Source is in bottom 16 or 32 bits of Fd.
	Float to fixed-point	3	VCVT{C}.<type>.<P> Fd, Fd, #<fbits>	IO, IX	Fd := convertFto<type>(Fd)	Destination is bottom 16 or 32 bits of Fd.
	Single to half-precision	3H	VCVTT{C}.F16.F32 Sd,Sm	ID, IO, OF, UF, IX	Sd:=convertStoH(Sm)	Destination is top 16 bits of Sd
	Single to half-precision	3H	VCVTB{C}.F16.F32 Sd,Sm	ID, IO, OF, UF, IX	Sd:=convertStoH(Sm)	Destination is bottom 16 bits of Sd
	Half to single-precision	3H	VCVTT{C}.F32.F16 Sd,Sm	ID, IO, OF, UF, IX	Sd:=convertHtoS(Sm)	Source is top 16 bits of Sm
	Half to single-precision	3H	VCVTB{C}.F32.F16 Sd,Sm	ID, IO, OF, UF, IX	Sd:=convertHtoS(Sm)	Source is bottom 16 bits of Sm
Insert constant	Insert constant in register	3	VMOV{C}.<P> Fd, #<fpconst>		Fd := <fpconst>	
Transfer registers	Copy VFP register		VMOV{C}.<P> Fd, Fm		Fd := Fm	
	ARM® to single		VMOV{C} Sn, Rd		Sn := Rd	
	Single to ARM		VMOV{C} Rd, Sn		Rd := Sn	
	Two ARM to two singles	2	VMOV{C} Sn, Sm, Rd, Rn		Sn := Rd, Sm := Rn	Sm must be S(n+1)
	Two singles to two ARM	2	VMOV{C} Rd, Rn, Sn, Sm		Rd := Sn, Rn := Sm	Sm must be S(n+1)
	Two ARM to double	2	VMOV{C} Dm, Rd, Rn		Dm[31:0] := Rd, Dm[63:32] := Rn	
	Double to two ARM	2	VMOV{C} Rd, Rn, Dm		Rd := Dm[31:0], Rn := Dm[63:32]	
	ARM to lower half of double		VMOV{C} Dn[0], Rd		Dn[31:0] := Rd	
	Lower half of double to ARM		VMOV{C} Rd, Dn[0]		Rd := Dn[31:0]	

Vector Floating Point Instruction Set
Quick Reference Card

Operation		§	Assembler	Exceptions	Action	Notes
Transfer registers (continued)	ARM to upper half of double		`VMOV{C} Dn[1], Rd`		Dn[63:32] := Rd	
	Upper half of double to ARM		`VMOV{C} Rd, Dn[1]`		Rd := Dn[63:32]	
	ARM to VFP system register		`VMSR{C} <VFPsysreg>, Rd`		VFPsysreg := Rd	
	VFP system register to ARM		`VMRS{C} Rd, <VFPsysreg>`		Rd := VFPsysreg	
	FPSCR flags to APSR		`VMRS{C} APSR_nzcv, FPSCR`		APSR flags := FPSCR flags	

Operation		§	Assembler	Synonyms	Action
Save VFP registers	Single		`VSTR{C} Fd, [Rn{, #<immed>}]`		[address] := Fd. Immediate range 0-1020, multiple of 4.
	Single, PC-relative		`VSTR{C} Fd, <label>`		
	Multiple, unindexed / increment after		`VSTM{C} Rn{!}, <VFPregs>`	`VSTMIA, VSTMEA`	Saves list of VFP registers, starting at address in Rn.
	decrement before		`VSTMDB{C} Rn!, <VFPregs>`	`VSTMFD` (full descending)	
	Push onto stack		`VPUSH{C} <VFPregs>`	`VSTMFD SP!`	
Load VFP registers	Single		`VLDR{C} Fd, [Rn{, #<immed>}]`		Fd := [address]. Immediate range 0-1020, multiple of 4.
	Single, PC-relative		`VLDR{C} Fd, <label>`		
	Multiple, unindexed / increment after		`VLDM{C} Rn{!}, <VFPregs>`	`VLDMIA, VLDMFD`	Loads list of VFP registers, starting at address in Rn.
	decrement before		`VLDMDB{C} Rn!, <VFPregs>`	`VLDMEA` (empty ascending)	
	Pop from stack		`VPOP{C} <VFPregs>`	`VLDM SP!`	

FPSCR format								Rounding		(Stride – 1)*3			Vector length – 1						Exception trap enable bits								Cumulative exception bits				
31	30	29	28	27	26	25	24	23	22	21	20		18	17	16	15			12	11	10	9	8	7			4	3	2	1	0
N	Z	C	V	QC	AHP	DB	FZ	RMODE		STRIDE			LEN			IDE			IXE	UFE	OFE	DZE	IOE	IDC			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. (Deprecated)

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			

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

Proprietary Notice

Document Number

ARM QRC 0007E

Change Log

Issue	Date	Change
A	Nov 2004	First Release
B	May 2005	Release for RVCT 2.2 SP1
C	March 2006	Release for RVCT 3.0
D	March 2007	Release for RVCT 3.1
E	Sept 2008	Release for RVCT 4.0