

Vector Floating Point Instruction Set

Quick Reference Card

Key to Tables	
{cond}	See Table Condition Field (on ARM side).
<S/D>	S (single precision) or D (double precision).
<S/D/X>	As above, or X (unspecified precision).
Fd, Fn, Fm	Sd, Sn, Sm (single precision), or Dd, Dn, Dm (double precision).

{E}	E : raise exception on any NaN. Without E : raise exception only on signaling NaNs.
{Z}	Round towards zero. Overrides FPSCR rounding mode.
<VFPregs>	A comma separated list of <i>consecutive</i> VFP registers, enclosed in braces ({ and }).
<VFPsysreg>	FPSCR, or FPSID.

Operation	Assembler	Exceptions	Action	Notes
Vector arithmetic	Multiply and negate and accumulate negate and accumulate and subtract negate and subtract	FMUL<S/D>{cond} Fd, Fn, Fm FNMUL<S/D>{cond} Fd, Fn, Fm FMAC<S/D>{cond} Fd, Fn, Fm FNMAC<S/D>{cond} Fd, Fn, Fm FMSC<S/D>{cond} Fd, Fn, Fm FNMSC<S/D>{cond} Fd, Fn, Fm	IO, OF, UF, IX IO, OF, UF, IX IO, OF, UF, IX IO, OF, UF, IX IO, OF, UF, IX IO, OF, UF, IX	Fd := Fn * Fm Fd := - (Fn * Fm) Fd := Fd + (Fn * Fm) Fd := Fd - (Fn * Fm) Fd := - Fd + (Fn * Fm) 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	FCMP{E}<S/D>{cond} Fd, Fm	IO	Set FPSCR flags on Fd - Fm
	Scalar convert	FCMP{E}Z<S/D>{cond} Fd	IO	Set FPSCR flags on Fd - 0
	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)
	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)
Save VFP registers	Multiple, unindexed increment after decrement before	FST<S/D>{cond} Fd, [Rn{, #<immed_8*4>}] FSTMIA<S/D/X>{cond} Rn, <VFPregs> FSTMIA<S/D/X>{cond} Rn!, <VFPregs>		[address] := Fd Saves list of VFP registers, starting at address in Rn. synonym: FSTMEA (empty ascending) synonym: FSTMFD (full descending)
Load VFP registers	Multiple, unindexed increment after decrement before	FLD<S/D>{cond} Fd, [Rn{, #<immed_8*4>}] FLDMIA<S/D/X>{cond} Rn, <VFPregs> FLDMIA<S/D/X>{cond} Rn!, <VFPregs> FLDMDB<S/D/X>{cond} Rn!, <VFPregs>		Fd := [address] Loads list of VFP registers, starting at address in Rn. synonym: FLDMFD (full descending) synonym: FLDMEA (empty ascending)
Transfer registers	ARM to single Single to ARM ARM to lower half of double Lower half of double to ARM ARM to upper half of double Upper half of double to ARM ARM to VFP system register VFP system register to ARM FPSCR flags to CPSR	FMSR{cond} Sn, Rd FMRs{cond} Rd, Sn FMDLR{cond} Dn, Rd FMRDL{cond} Rd, Dn FMDHR{cond} Dn, Rd FMRDH{cond} Rd, Dn FMRX{cond} <VFPsysreg>, Rd FMRX{cond} Rd, <VFPsysreg> FMSTAT{cond}		Sn := Rd Rd := Sn Dn[31:0] := Rd Rd := Dn[31:0] Dn[63:32] := Rd Rd := Dn[63:32] VFPsysreg := Rd Rd := VFPsysreg CPSR flags := FPSCR flags

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

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.														

If Fd is S0-S7 or D0-D3, operation is Scalar (regardless of vector length).	If Fd is S8-S31 or D4-D15, and Fm is S0-S7 or D0-D3, operation is Mixed (Fm scalar, others vector).
If Fd is S8-S31 or D4-D15, and Fm is S8-S31 or D4-D15, operation is Vector.	S0-S7 (or D0-D3), S8-S15 (D4-D7), S16-S23 (D8-D11), S24-S31 (D12-D15) each form a circulating bank of registers.

Thumb® Instruction Set

Quick Reference Card

All Thumb registers are Lo (R0-R7) except where specified. Hi registers are R8-R15.

Operation		\$	Assembler	Updates	Action	Notes
Move	Immediate		MOV Rd, #<immed_8>	N Z	Rd := immed_8	8-bit immediate value.
	Lo to Lo		MOV Rd, Rm	N Z * *	Rd := Rm	* Clears C and V flags.
	Hi to Lo, Lo to Hi, Hi to Hi		MOV Rd, Rm		Rd := Rm	Not Lo to Lo. Flags not affected.
Arithmetic	Add		ADD Rd, Rn, #<immed_3>	N Z C V	Rd := Rn + immed_3	3-bit immediate value.
	Lo and Lo		ADD Rd, Rn, Rm	N Z C V	Rd := Rn + Rm	
	Hi to Lo, Lo to Hi, Hi to Hi		ADD Rd, Rm		Rd := Rd + Rm	Not Lo to Lo. Flags not affected.
	immediate		ADD Rd, #<immed_8>	N Z C V	Rd := Rd + immed_8	8-bit immediate value.
	with carry		ADC Rd, Rm	N Z C V	Rd := Rd + Rm + C-bit	
	value to SP		ADD SP, #<immed_7*4>		R13 := R13 + immed_7 * 4	9-bit immediate value (word-aligned). Flags not affected.
	form address from SP		ADD Rd, SP, #<immed_8*4>		Rd := R13 + immed_8 * 4	10-bit immediate value (word-aligned). Flags not affected.
	form address from PC		ADD Rd, PC, #<immed_8*4>		Rd := (R15 AND 0xFFFFF) + immed_8 * 4	10-bit immediate value (word-aligned). Flags not affected.
	Subtract		SUB Rd, Rn, Rm	N Z C V	Rd := Rn - Rm	
	immediate 3		SUB Rd, Rn, #<immed_3>	N Z C V	Rd := Rn - immed_3	3-bit immediate value.
	immediate 8		SUB Rd, #<immed_8>	N Z C V	Rd := Rd - immed_8	8-bit immediate value.
	with carry		SBC Rd, Rm	N Z C V	Rd := Rd - Rm - NOT C-bit	
	value from SP		SUB SP, #<immed_7*4>		R13 := R13 - immed_7 * 4	9-bit immediate value (word-aligned). Flags not affected.
	Negate		NEG Rd, Rm	N Z C V	Rd := - Rm	
	Multiply		MUL Rd, Rm	N Z * *	Rd := Rm * Rd	* C and V flags unpredictable in §4T, unchanged in §5T.
	Compare		CMP Rn, Rm	N Z C V	update CPSR flags on Rn - Rm	Can be Lo to Lo, Lo to Hi, Hi to Lo, or Hi to Hi.
	negative		CMN Rn, Rm	N Z C V	update CPSR flags on Rn + Rm	
	immediate		CMP Rn, #<immed_8>	N Z C V	update CPSR flags on Rn - immed_8	8-bit immediate value.
	No operation		NOP		R8 := R8	Flags not affected.
Logical	AND		AND Rd, Rm	N Z	Rd := Rd AND Rm	
	Exclusive OR		EOR Rd, Rm	N Z	Rd := Rd EOR Rm	
	OR		ORR Rd, Rm	N Z	Rd := Rd OR Rm	
	Bit clear		BIC Rd, Rm	N Z	Rd := Rd AND NOT Rm	
	Move NOT		MVN Rd, Rm	N Z	Rd := NOT Rm	
	Test bits		TST Rn, Rm	N Z	update CPSR flags on Rn AND Rm	
Shift/rotate	Logical shift left		LSL Rd, Rm, #<immed_5>	N Z C*	Rd := Rm << immed_5	Allowed shifts 0-31. * C flag unaffected if shift is 0.
			LSL Rd, Rs	N Z C*	Rd := Rd << Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
	Logical shift right		LSR Rd, Rm, #<immed_5>	N Z C V	Rd := Rm >> immed_5	Allowed shifts 1-32.
			LSR Rd, Rs	N Z C	Rd := Rd >> Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
	Arithmetic shift right		ASR Rd, Rm, #<immed_5>	N Z C	Rd := Rm ASR immed_5	Allowed shifts 1-32.
			ASR Rd, Rs	N Z C*	Rd := Rd ASR Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
Branch	Rotate right		ROR Rd, Rs	N Z C*	Rd := Rd ROR Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
	Conditional branch		B{cond} label		R15 := label	label must be within - 252 to + 258 bytes of current instruction. See Table Condition Field (ARM side). AL not allowed.
	Unconditional branch		B label		R15 := label	label must be within ±2Kb of current instruction.
	Long branch with link		BL label		R14 := R15 - 2, R15 := label	Encoded as two Thumb instructions.
	Branch and exchange		BX Rm		R15 := Rm AND 0xFFFFF	label must be within ±4Mb of current instruction.
	Branch with link and exchange	5T	BLX label		R14 := R15 - 2, R15 := label Change to ARM	Change to ARM state if Rm[0] = 0.
Software Interrupt	Branch with link and exchange	5T	BLX Rm		R14 := R15 - 2, R15 := Rm AND 0xFFFFF Change to ARM if Rm[0] = 0	Encoded as two Thumb instructions. label must be within ±4Mb of current instruction.
			SWI <immed_8>		Software interrupt processor exception	8-bit immediate value encoded in instruction.
Breakpoint		5T	BKPT <immed_8>		Prefetch abort <i>or</i> enter debug state	

Thumb Instruction Set
Quick Reference Card

Operation		\$	Assembler	Action	Notes
Load	with immediate offset, word		LDR Rd, [Rn, #<immed_5*4>]	Rd := [Rn + immed_5 * 4]	
	halfword		LDRH Rd, [Rn, #<immed_5*2>]	Rd := ZeroExtend([Rn + immed_5 * 2][15:0])	Clears bits 31:16
	byte		LDRB Rd, [Rn, #<immed_5>]	Rd := ZeroExtend([Rn + immed_5][7:0])	Clears bits 31:8
	with register offset, word		LDR Rd, [Rn, Rm]	Rd := [Rn + Rm]	
	halfword		LDRH Rd, [Rn, Rm]	Rd := ZeroExtend([Rn + Rm][15:0])	Clears bits 31:16
	signed halfword		LDRSH Rd, [Rn, Rm]	Rd := SignExtend([Rn + Rm][15:0])	Sets bits 31:16 to bit 15
	byte		LDRB Rd, [Rn, Rm]	Rd := ZeroExtend([Rn + Rm][7:0])	Clears bits 31:8
	signed byte		LDRSB Rd, [Rn, Rm]	Rd := SignExtend([Rn + Rm][7:0])	Sets bits 31:8 to bit 7
	PC-relative		LDR Rd, [PC, #<immed_8*4>]	Rd := [(R15 AND 0xFFFFFFF0) + immed_8 * 4]	
	SP-relative		LDR Rd, [SP, #<immed_8*4>]	Rd := [R13 + immed_8 * 4]	
	Multiple		LDMIA Rn!, <reglist>	Loads list of registers	Always updates base register.
Store	with immediate offset, word		STR Rd, [Rn, #<immed_5*4>]	[Rn + immed_5 * 4] := Rd	
	halfword		STRH Rd, [Rn, #<immed_5*2>]	[Rn + immed_5 * 2][15:0] := Rd[15:0]	Ignores Rd[31:16]
	byte		STRB Rd, [Rn, #<immed_5>]	[Rn + immed_5][7:0] := Rd[7:0]	Ignores Rd[31:8]
	with register offset, word		STR Rd, [Rn, Rm]	[Rn + Rm] := Rd	
	halfword		STRH Rd, [Rn, Rm]	[Rn + Rm][15:0] := Rd[15:0]	Ignores Rd[31:16]
	byte		STRB Rd, [Rn, Rm]	[Rn + Rm][7:0] := Rd[7:0]	Ignores Rd[31:8]
	SP-relative, word		STR Rd, [SP, #<immed_8*4>]	[R13 + immed_8 * 4] := Rd	
	Multiple		STMIA Rn!, <reglist>	Stores list of registers	Always updates base register.
Push/ Pop	Push		PUSH <reglist>	Push registers onto stack	Full descending stack.
	Push with link		PUSH <reglist, LR>	Push LR and registers onto stack	
	Pop		POP <reglist>	Pop registers from stack	
	Pop and return		POP <reglist, PC>	Pop registers, branch to address loaded to PC	
	Pop and return with exchange	5T	POP <reglist, PC>	Pop, branch, and change to ARM state if address[0] = 0	

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This reference card is intended only to assist the reader in the use of the product. ARM Ltd shall not be liable for any loss or damage arising from the use of any information in this reference card, or any error or omission in such information, or any incorrect use of the product.

Document Number

ARM QRC 0001E

Change Log

Issue	Date	By	Change
A	June 1995	BJH	First Release
B	Sept 1996	BJH	Second Release
C	Nov 1998	BJH	Third Release
D	Oct 1999	CKS	Fourth Release
E	Oct 2000	CKS	Fifth Release