Thumb® Instruction Set
Quick Reference Card

Key to Tables				
<loreglist>	A comma-separated list of Lo registers, enclosed in braces, { and }.		<loreglist+LR>	A comma-separated list of Lo registers. plus the LR, enclosed in braces, { and }.
			<loreglist+PC>	A comma-separated list of Lo registers. plus the PC, enclosed in braces, { and }.

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>	N Z	Rd := immed	Immediate range 0-255.
	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.
	Copy Any to Any	6	CPY Rd, Rm		Rd := Rm	Any register to any register. Flags not affected.
Arithmetic	Add		ADD Rd, Rn, #<immed>	N Z C V	Rd := Rn + immed	Immediate range 0-7.
	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>	N Z C V	Rd := Rd + immed	Immediate range 0-255.
	with carry		ADC Rd, Rm	N Z C V	Rd := Rd + Rm + C-bit	
	value to SP		ADD SP, #<immed>		R13 := R13 + immed	Immediate range 0-508 (word-aligned). Flags not affected.
	form address from SP		ADD Rd, SP, #<immed>		Rd := R13 + immed	Immediate range 0-1020 (word-aligned). Flags not affected.
	form address from PC		ADD Rd, PC, #<immed>		Rd := (R15 AND 0xFFFFFFFC) + immed	Immediate range 0-1020 (word-aligned). Flags not affected.
	Subtract		SUB Rd, Rn, Rm	N Z C V	Rd := Rn – Rm	
	immediate 3		SUB Rd, Rn, #<immed>	N Z C V	Rd := Rn – immed	Immediate range 0-7.
	immediate 8		SUB Rd, #<immed>	N Z C V	Rd := Rd – immed	Immediate range 0-255.
	with carry		SBC Rd, Rm	N Z C V	Rd := Rd – Rm – NOT C-bit	
	value from SP		SUB SP, #<immed>		R13 := R13 – immed	Immediate range 0-508 (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 and above
	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>	N Z C V	update CPSR flags on Rn – immed	Immediate range 0-255.
	No operation		NOP		None	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, #<shift>	N Z C*	Rd := Rm << shift	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, #<shift>	N Z C	Rd := Rm >> shift	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, #<shift>	N Z C	Rd := Rm ASR shift	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.
	Rotate right		ROR Rd, Rs	N Z C*	Rd := Rd ROR Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
Reverse	Bytes in word	6	REV Rd, Rm		Rd[31:24] := Rm[7:0], Rd[23:16] := Rm[15:8], Rd[15:8] := Rm[23:16], Rd[7:0] := Rm[31:24]	
	Bytes in both halfwords	6	REV16 Rd, Rm		Rd[15:8] := Rm[7:0], Rd[7:0] := Rm[15:8], Rd[31:24] := Rm[23:16], Rd[23:16] := Rm[31:24]	
	Bytes in low halfword, sign extend	6	REVSH Rd, Rm		Rd[15:8] := Rm[7:0], Rd[7:0] := Rm[15:8], Rd[31:16] := Rm[7] * &FFFF	

Operation		§	Assembler	Action	Notes
Load	with immediate offset, word		`LDR Rd, [Rn, #<immed>]`	Rd := [Rn + immed]	Immediate range 0-124, multiple of 4.
	halfword		`LDRH Rd, [Rn, #<immed>]`	Rd := ZeroExtend([Rn + immed][15:0])	Clears bits 31:16. Immediate range 0-62, even.
	byte		`LDRB Rd, [Rn, #<immed>]`	Rd := ZeroExtend([Rn + immed][7:0])	Clears bits 31:8. Immediate range 0-31.
	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>]`	Rd := [(R15 AND 0xFFFFFFFC) + immed]	Immediate range 0-1020, multiple of 4.
	SP-relative		`LDR Rd, [SP, #<immed>]`	Rd := [R13 + immed]	Immediate range 0-1020, multiple of 4.
	Multiple		`LDMIA Rn!, <reglist>`	Loads list of registers	Always updates base register.
Store	with immediate offset, word		`STR Rd, [Rn, #<immed>]`	[Rn + immed] := Rd	Immediate range 0-124, multiple of 4.
	halfword		`STRH Rd, [Rn, #<immed>]`	[Rn + immed][15:0] := Rd[15:0]	Ignores Rd[31:16]. Immediate range 0-62, even.
	byte		`STRB Rd, [Rn, #<immed>]`	[Rn + immed][7:0] := Rd[7:0]	Ignores Rd[31:8]. Immediate range 0-31.
	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>]`	[R13 + immed] := Rd	Immediate range 0-1020, multiple of 4.
	Multiple		`STMIA Rn!, <reglist>`	Stores list of registers	Always updates base register.
Push/ Pop	Push		`PUSH <loreglist>`	Push registers onto stack	Full descending stack.
	Push with link		`PUSH <loreglist+LR>`	Push LR and registers onto stack	
	Pop		`POP <loreglist>`	Pop registers from stack	
	Pop and return	4T	`POP <loreglist+PC>`	Pop registers, branch to address loaded to PC	
	Pop and return with exchange	5T	`POP <loreglist+PC>`	Pop, branch, and change to ARM state if address[0] = 0	
Branch	Conditional branch		`B{cond} label`	R15 := label	label must be within – 252 to + 258 bytes of current instruction. See Table **Condition Field** on reverse.
	Unconditional branch		`B label`	R15 := label	`label` must be within ±2Kb of current instruction.
	Long branch with link		`BL label`	R14 := address of next instruction, R15 := label	Encoded as two Thumb instructions. label must be within ±4Mb of current instruction.
	Branch and exchange		`BX Rm`	R15 := Rm AND 0xFFFFFFFE	Change to ARM state if Rm[0] = 0.
	Branch with link and exchange	5T	`BLX label`	R14 := address of next instruction, R15 := label Change to ARM	Encoded as two Thumb instructions. label must be within ±4Mb of current instruction.
	Branch with link and exchange	5T	`BLX Rm`	R14 := address of next instruction, R15 := Rm AND 0xFFFFFFFE	Change to ARM state if Rm[0] = 0
Extend	Signed extend halfword to word	6	`SXTH Rd, Rm`	Rd[31:0] := SignExtend(Rm[15:0])	
	Signed extend byte to word	6	`SXTB Rd, Rm`	Rd[31:0] := SignExtend(Rm[7:0])	
	Unsigned extend halfword to word	6	`UXTH Rd, Rm`	Rd[31:0] := ZeroExtend(Rm[15:0])	
	Unsigned extend byte to word	6	`UXTB Rd, Rm`	Rd[31:0] := ZeroExtend(Rm[7:0])	
Processor state change	Software interrupt		`SWI <immed_8>`	Software interrupt processor exception	8-bit immediate value encoded in instruction.
	Change processor state	6	`CPSID <iflags>`	Disable specified interrups	
		6	`CPSIE <iflags>`	Enable specified interrups	
	Set endianness	6	`SETEND <endianness>`	Sets endianness for loads and saves.	`<endianness>` can be `BE` (Big Endian) or `LE` (Little Endian).
	Breakpoint	5T	`BKPT <immed_8>`	Prefetch abort *or* enter debug state	8-bit immediate value encoded in instruction.