ARM® Instruction Set
Quick Reference Card

Key to Tables	
{cond}	Refer to Table **Condition Field {cond}**. Omit for unconditional execution.
<Operand2>	Refer to Table **Flexible Operand 2**. Shift and rotate are only available as part of Operand2.
<fields>	Refer to Table **PSR fields**.
<PSR>	Either CPSR (Current Processor Status Register) or SPSR (Saved Processor Status Register)
{S}	Updates condition flags if S present.
C*, V*	Flag is unpredictable in Architecture v4 and earlier, unchanged in Architecture v5 and later.
Q	Sticky flag. Always updates on overflow (no S option). Read and reset using MRS and MSR.
x,y	B meaning half-register [15:0], or T meaning [31:16].
<immed_8r>	A 32-bit constant, formed by right-rotating an 8-bit value by an even number of bits.
<immed_8*4>	A 10-bit constant, formed by left-shifting an 8-bit value by two bits.

<a_mode2>	Refer to Table **Addressing Mode 2**.
<a_mode2P>	Refer to Table **Addressing Mode 2 (Post-indexed only)**.
<a_mode3>	Refer to Table **Addressing Mode 3**.
<a_mode4L>	Refer to Table **Addressing Mode 4 (Block load or Stack pop)**.
<a_mode4S>	Refer to Table **Addressing Mode 4 (Block store or Stack push)**.
<a_mode5>	Refer to Table **Addressing Mode 5**.
<reglist>	A comma-separated list of registers, enclosed in braces, { and }.
{!}	Updates base register after data transfer if ! present.
+/-	+ or –. (+ may be omitted.)
§	Refer to Table **ARM architecture versions**.

Operation		§	Assembler	S updates	Q	Action
Move	Move		MOV{cond}{S} Rd, <Operand2>	N Z C		Rd := Operand2
	NOT		MVN{cond}{S} Rd, <Operand2>	N Z C		Rd := 0xFFFFFFFF EOR Operand2
	PSR to register	3	MRS{cond} Rd, <PSR>			Rd := PSR
	register to PSR	3	MSR{cond} <PSR>_<fields>, Rm			PSR := Rm (selected bytes only)
	immediate to PSR	3	MSR{cond} <PSR>_<fields>, #<immed_8r>			PSR := immed_8r (selected bytes only)
	40-bit accumulator to register	XS	MRA{cond} RdLo, RdHi, Ac			RdLo := Ac[31:0], RdHi := Ac[39:32]
	register to 40-bit accumulator	XS	MAR{cond} Ac, RdLo, RdHi			Ac[31:0] := RdLo, Ac[39:32] := RdHi
Arithmetic	Add		ADD{cond}{S} Rd, Rn, <Operand2>	N Z C V		Rd := Rn + Operand2
	with carry		ADC{cond}{S} Rd, Rn, <Operand2>	N Z C V		Rd := Rn + Operand2 + Carry
	saturating	5E	QADD{cond} Rd, Rm, Rn		Q	Rd := SAT(Rm + Rn)
	double saturating	5E	QDADD{cond} Rd, Rm, Rn		Q	Rd := SAT(Rm + SAT(Rn * 2))
	Subtract		SUB{cond}{S} Rd, Rn, <Operand2>	N Z C V		Rd := Rn – Operand2
	with carry		SBC{cond}{S} Rd, Rn, <Operand2>	N Z C V		Rd := Rn – Operand2 – NOT(Carry)
	reverse subtract		RSB{cond}{S} Rd, Rn, <Operand2>	N Z C V		Rd := Operand2 – Rn
	reverse subtract with carry		RSC{cond}{S} Rd, Rn, <Operand2>	N Z C V		Rd := Operand2 – Rn – NOT(Carry)
	saturating	5E	QSUB{cond} Rd, Rm, Rn		Q	Rd := SAT(Rm – Rn)
	double saturating	5E	QDSUB{cond} Rd, Rm, Rn		Q	Rd := SAT(Rm – SAT(Rn * 2))
	Multiply	2	MUL{cond}{S} Rd, Rm, Rs	N Z C*		Rd := (Rm * Rs)[31:0]
	accumulate	2	MLA{cond}{S} Rd, Rm, Rs, Rn	N Z C*		Rd := ((Rm * Rs) + Rn)[31:0]
	unsigned long	M	UMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C* V*		RdHi,RdLo := unsigned(Rm * Rs)
	unsigned accumulate long	M	UMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C* V*		RdHi,RdLo := unsigned(RdHi,RdLo + Rm * Rs)
	signed long	M	SMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C* V*		RdHi,RdLo := signed(Rm * Rs)
	signed accumulate long	M	SMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z C* V*		RdHi,RdLo := signed(RdHi,RdLo + Rm * Rs)
	signed 16 * 16 bit	5E	SMULxy{cond} Rd, Rm, Rs			Rd := Rm[x] * Rs[y]
	signed 32 * 16 bit	5E	SMULWy{cond} Rd, Rm, Rs			Rd := (Rm * Rs[y])[47:16]
	signed accumulate 16 * 16 bit	5E	SMLAxy{cond} Rd, Rm, Rs, Rn		Q	Rd := Rn + Rm[x] * Rs[y]
	signed accumulate 32 * 16 bit	5E	SMLAWy{cond} Rd, Rm, Rs, Rn		Q	Rd := Rn + (Rm * Rs[y])[47:16]
	signed accumulate long 16 * 16 bit	5E	SMLALxy{cond} RdLo, RdHi, Rm, Rs			RdHi,RdLo := RdHi,RdLo + Rm[x] * Rs[y]
	Multiply with internal 40-bit accumulate	XS	MIA{cond} Ac, Rm, Rs			Ac := Ac + Rm * Rs
	packed halfword	XS	MIAPH{cond} Ac, Rm, Rs			Ac := Ac + Rm[15:0] * Rs[15:0] + Rm[31:16] * Rs[31:16]
	halfword	XS	MIAxy{cond} Ac, Rm, Rs			Ac := Ac + Rm[x] * Rs[y]
	Count leading zeroes	5	CLZ{cond} Rd, Rm			Rd := number of leading zeroes in Rm
Logical	Test		TST{cond} Rn, <Operand2>	N Z C		Update CPSR flags on Rn AND Operand2
	Test equivalence		TEQ{cond} Rn, <Operand2>	N Z C		Update CPSR flags on Rn EOR Operand2
	AND		AND{cond}{S} Rd, Rn, <Operand2>	N Z C		Rd := Rn AND Operand2
	EOR		EOR{cond}{S} Rd, Rn, <Operand2>	N Z C		Rd := Rn EOR Operand2
	ORR		ORR{cond}{S} Rd, Rn, <Operand2>	N Z C		Rd := Rn OR Operand2
	Bit Clear		BIC{cond}{S} Rd, Rn, <Operand2>	N Z C		Rd := Rn AND NOT Operand2
Compare	Compare		CMP{cond} Rn, <Operand2>	N Z C V		Update CPSR flags on Rn – Operand2
	negative		CMN{cond} Rn, <Operand2>	N Z C V		Update CPSR flags on Rn + Operand2
No Op	No operation		NOP			None