

ARM® Instruction Set

Quick Reference Card

Key to Tables			
{cond}	Refer to Table Condition Field {cond} . Omit for unconditional execution.	<a_mode2>	Refer to Table Addressing Mode 2 .
<Operand2>	Refer to Table Flexible Operand 2 . Shift and rotate are only available as part of Operand2.	<a_mode2P>	Refer to Table Addressing Mode 2 (Post-indexed only) .
<fields>	Refer to Table PSR fields .	<a_mode3>	Refer to Table Addressing Mode 3 .
<PSR>	Either CPSR (Current Processor Status Register) or SPSR (Saved Processor Status Register)	<a_mode4L>	Refer to Table Addressing Mode 4 (Block load or Stack pop) .
{S}	Updates condition flags if S present.	<a_mode4S>	Refer to Table Addressing Mode 4 (Block store or Stack push) .
C*, V*	Flag is unpredictable in Architecture v4 and earlier, unchanged in Architecture v5 and later.	<a_mode5>	Refer to Table Addressing Mode 5 .
Q	Sticky flag. Always updates on overflow (no S option). Read and reset using MRS and MSR.	<reglist>	A comma-separated list of registers, enclosed in braces, { and }.
x, y	B meaning half-register [15:0], or T meaning [31:16].	{!}	Updates base register after data transfer if ! present.
<immed_8r>	A 32-bit constant, formed by right-rotating an 8-bit value by an even number of bits.	+/-	+ or -. (+ may be omitted.)
<immed_8*4>	A 10-bit constant, formed by left-shifting an 8-bit value by two bits.	\$	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

ARM Instruction Set

Quick Reference Card

Operation		\$	Assembler	Action	Notes
Branch	Branch		B{cond} label	R15 := label	label must be within ±32Mb of current instruction.
	with link		BL{cond} label	R14 := R15 - 4, R15 := label	label must be within ±32Mb of current instruction.
	and exchange	4T,5	BX{cond} Rm	R15 := Rm, Change to Thumb if Rm[0] is 1	Cannot be conditional.
	with link and exchange (1)	5T	BLX label	R14 := R15 - 4, R15 := label, Change to Thumb	label must be within ±32Mb of current instruction.
	with link and exchange (2)	5	BLX{cond} Rm	R14 := R15 - 4, R15 := Rm[31:1] Change to Thumb if Rm[0] is 1	
Load	Word		LDR{cond} Rd, <a_mode2>	Rd := [address]	Rd must not be R15.
	User mode privilege branch (§ 5T: and exchange)		LDR{cond}T Rd, <a_mode2P>		Rd must not be R15.
			LDR{cond} R15, <a_mode2>	R15 := [address][31:1] (§ 5T: Change to Thumb if [address][0] is 1)	
	Byte		LDR{cond}B Rd, <a_mode2>	Rd := ZeroExtend[byte from address]	Rd must not be R15.
	User mode privilege signed		LDR{cond}BT Rd, <a_mode2P>		Rd must not be R15.
	Halfword	4	LDR{cond}SB Rd, <a_mode3>	Rd := SignExtend[byte from address]	Rd must not be R15.
	signed	4	LDR{cond}H Rd, <a_mode3>	Rd := ZeroExtend[halfword from address]	Rd must not be R15.
	Doubleword	4	LDR{cond}SH Rd, <a_mode3>	Rd := SignExtend[halfword from address]	Rd must not be R15.
	Pop, or Block data load return (and exchange)	5E*	LDR{cond}D Rd, <a_mode3>	Rd := [address], R(d+1) := [address + 4]	Rd must be even, and not R14.
	and restore CPSR		LDM{cond}<a_mode4L> Rn{!}, <reglist-pc>	Load list of registers from [Rn]	
Load multiple			LDM{cond}<a_mode4L> Rn{!}, <reglist+pc>	Load registers, R15 := [address][31:1] (§ 5T: Change to Thumb if [address][0] is 1)	
	User mode registers		LDM{cond}<a_mode4L> Rn{!}, <reglist+pc>^	Load registers, branch (§ 5T: and exchange), CPSR := SPSR	Use from exception modes only.
	Memory system hint	5E*	PLD <a_mode2>	Load list of User mode registers from [Rn] Memory may prepare to load from address	Use from privileged modes only.
Soft preload					Cannot be conditional.
Store	Word		STR{cond} Rd, <a_mode2>	[address] := Rd	
	User mode privilege		STR{cond}T Rd, <a_mode2P>	[address] := Rd	
	Byte		STR{cond}B Rd, <a_mode2>	[address][7:0] := Rd[7:0]	
	User mode privilege		STR{cond}BT Rd, <a_mode2P>	[address][7:0] := Rd[7:0]	
	Halfword	4	STR{cond}H Rd, <a_mode3>	[address][15:0] := Rd[15:0]	
	Doubleword	5E*	STR{cond}D Rd, <a_mode3>	[address] := Rd, [address + 4] := R(d+1)	Rd must be even, and not R14.
Store multiple	Push, or Block data store		STM{cond}<a_mode4S> Rn{!}, <reglist>	Store list of registers to [Rn]	
	User mode registers		STM{cond}<a_mode4S> Rn{!}, <reglist>^	Store list of User mode registers to [Rn]	Use from privileged modes only.
Swap	Word	3	SWP{cond} Rd, Rm, [Rn]	temp := [Rn], [Rn] := Rm, Rd := temp	
	Byte	3	SWP{cond}B Rd, Rm, [Rn]	temp := ZeroExtend([Rn][7:0]), [Rn][7:0] := Rm[7:0], Rd := temp	
Coprocessors	Data operations	2	CDP{cond} <copr>, <op1>, CRd, CRn, CRm{, <op2>}	Coprocessor defined	
	Alternative operations	5	CDP2 <copr>, <op1>, CRd, CRn, CRm{, <op2>}		Cannot be conditional.
	Move to ARM reg from coproc	2	MRC{cond} <copr>, <op1>, Rd, CRn, CRm{, <op2>}		
	Alternative moves	5	MRC2 <copr>, <op1>, Rd, CRn, CRm{, <op2>}		Cannot be conditional.
	Two ARM register move	5E*	MRRC{cond} <copr>, <op1>, Rd, Rn, CRm		
	Move to coproc from ARM reg	2	MCR{cond} <copr>, <op1>, Rd, CRn, CRm{, <op2>}		
	Alternative moves	5	MCR2 <copr>, <op1>, Rd, CRn, CRm{, <op2>}		Cannot be conditional.
	Two ARM register move	5E*	MCRR{cond} <copr>, <op1>, Rd, Rn, CRm		
	Load	2	LDC{cond} <copr>, CRd, <a_mode5>		Cannot be conditional.
	Alternative loads	5	LDC2 <copr>, CRd, <a_mode5>		
Store		2	STC{cond} <copr>, CRd, <a_mode5>		
	Alternative stores	5	STC2 <copr>, CRd, <a_mode5>		Cannot be conditional.
Software interrupt			SWI{cond} <immed_24>	Software interrupt processor exception	24-bit value encoded in instruction.
Breakpoint		5	BKPT <immed_16>	Prefetch abort <i>or</i> enter debug state	Cannot be conditional.

ARM Addressing Modes
Quick Reference Card

Addressing Mode 2 - Word and Unsigned Byte Data Transfer			
Pre-indexed	Immediate offset	[Rn, #+/-<immed_12>]{!}	Equivalent to [Rn,#0]
	Zero offset	[Rn]	
	Register offset	[Rn, +/-Rm]{!}	
	Scaled register offset	[Rn, +/-Rm, LSL #<immed_5>]{!}	
		[Rn, +/-Rm, LSR #<immed_5>]{!}	
		[Rn, +/-Rm, ASR #<immed_5>]{!}	
Post-indexed	Immediate offset	[Rn, +/-Rm, ROR #<immed_5>]{!}	Allowed shifts 0-31 Allowed shifts 1-32 Allowed shifts 1-32 Allowed shifts 1-31
	Register offset	[Rn, +/-Rm, RRX]{!}	
	Scaled register offset	[Rn], #+/-<immed_12>	
		[Rn], +/-Rm	
		[Rn], +/-Rm, LSL #<immed_5>	
		[Rn], +/-Rm, LSR #<immed_5>	
		[Rn], +/-Rm, ASR #<immed_5>	Allowed shifts 1-32 Allowed shifts 1-32 Allowed shifts 1-31
		[Rn], +/-Rm, ROR #<immed_5>	
		[Rn], +/-Rm, RRX	

Addressing Mode 2 (Post-indexed only)			
Post-indexed	Immediate offset	[Rn], #+/-<immed_12>	Equivalent to [Rn],#0
	Zero offset	[Rn]	
	Register offset	[Rn], +/-Rm	
	Scaled register offset	[Rn], +/-Rm, LSL #<immed_5>	
		[Rn], +/-Rm, LSR #<immed_5>	
		[Rn], +/-Rm, ASR #<immed_5>	
		[Rn], +/-Rm, ROR #<immed_5>	Allowed shifts 0-31 Allowed shifts 1-32 Allowed shifts 1-32 Allowed shifts 1-31
		[Rn], +/-Rm, RRX	

Addressing Mode 3 - Halfword, Signed Byte, and Doubleword Data Transfer			
Pre-indexed	Immediate offset	[Rn, #+/-<immed_8>]{!}	Equivalent to [Rn,#0]
	Zero offset	[Rn]	
	Register	[Rn, +/-Rm]{!}	
Post-indexed	Immediate offset	[Rn], #+/-<immed_8>	
	Register	[Rn], +/-Rm	

Addressing Mode 4 - Multiple Data Transfer			
Block load		Stack pop	
IA	Increment After	FD	Full Descending
IB	Increment Before	ED	Empty Descending
DA	Decrement After	FA	Full Ascending
DB	Decrement Before	EA	Empty Ascending
Block store		Stack push	
EA	Increment After	EA	Empty Ascending
IB	Increment Before	FA	Full Ascending
DA	Decrement After	ED	Empty Descending
DB	Decrement Before	FD	Full Descending

Addressing Mode 5 - Coprocessor Data Transfer			
Pre-indexed	Immediate offset	[Rn, #+/-<immed_8*4>]{!}	Equivalent to [Rn,#0]
	Zero offset	[Rn]	
Post-indexed	Immediate offset	[Rn], #+/-<immed_8*4>	
Unindexed	No offset	[Rn], {8-bit copro. option}	

ARM architecture versions	
<i>n</i>	ARM architecture version <i>n</i> and above.
<i>n</i> T	T variants of ARM architecture version <i>n</i> and above.
M	ARM architecture version 3M, and 4 and above, except xM variants.
<i>n</i> E	All E variants of ARM architecture version <i>n</i> and above.
<i>n</i> E*	E variants of ARM architecture version <i>n</i> and above, except xP variants.
XS	XScale coprocessor instruction

Flexible Operand 2		
Immediate value	#<immed_8r>	
Logical shift left immediate	Rm, LSL #<immed_5>	Allowed shifts 0-31
Logical shift right immediate	Rm, LSR #<immed_5>	Allowed shifts 1-32
Arithmetic shift right immediate	Rm, ASR #<immed_5>	Allowed shifts 1-32
Rotate right immediate	Rm, ROR #<immed_5>	Allowed shifts 1-31
Register	Rm	
Rotate right extended	Rm, RRX	
Logical shift left register	Rm, LSL Rs	
Logical shift right register	Rm, LSR Rs	
Arithmetic shift right register	Rm, ASR Rs	
Rotate right register	Rm, ROR Rs	

PSR fields (use at least one suffix)		
Suffix	Meaning	
c	Control field mask byte	PSR[7:0]
f	Flags field mask byte	PSR[31:24]
s	Status field mask byte	PSR[23:16]
x	Extension field mask byte	PSR[15:8]

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