ARM Instruction Set
Quick Reference Card

Single data item loads and stores		§	Assembler	Action if <op> is LDR	Action if <op> is STR	Notes
Load or store word, byte or halfword	Immediate offset		`<op>{size}{T} Rd, [Rn {, #<offset>}]{!}`	Rd := [address, size]	[address, size] := Rd	1, N
	Post-indexed, immediate		`<op>{size}{T} Rd, [Rn], #<offset>`	Rd := [address, size]	[address, size] := Rd	2
	Register offset		`<op>{size} Rd, [Rn, +/-Rm {, <opsh>}]{!}`	Rd := [address, size]	[address, size] := Rd	3, N
	Post-indexed, register		`<op>{size}{T} Rd, [Rn], +/-Rm {, <opsh>}`	Rd := [address, size]	[address, size] := Rd	4
	PC-relative		`<op>{size} Rd, <label>`	Rd := [label, size]	Not available	5, N
Load or store doubleword	Immediate offset	5E*	`<op>D Rd1, Rd2, [Rn {, #<offset>}]{!}`	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	6, 9
	Post-indexed, immediate	5E*	`<op>D Rd1, Rd2, [Rn], #<offset>`	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	6, 9
	Register offset	5E*	`<op>D Rd1, Rd2, [Rn, +/-Rm {, <opsh>}]{!}`	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	7, 9
	Post-indexed, register	5E*	`<op>D Rd1, Rd2, [Rn], +/-Rm {, <opsh>}`	Rd1 := [address], Rd2 := [address + 4]	[address] := Rd1, [address + 4] := Rd2	7, 9
	PC-relative	5E*	`<op>D Rd1, Rd2, <label>`	Rd1 := [label], Rd2 := [label + 4]	Not available	8, 9

Other memory operations		§	Assembler	Action	Notes
Load multiple	Block data load		`LDM{IA|IB|DA|DB} Rn{!}, <reglist-PC>`	Load list of registers from [Rn]	N, I
	return (and exchange)		`LDM{IA|IB|DA|DB} Rn{!}, <reglist+PC>`	Load registers, PC := [address][31:1] (§ 5T: Change to Thumb if [address][0] is 1)	I
	and restore CPSR		`LDM{IA|IB|DA|DB} Rn{!}, <reglist+PC>^`	Load registers, branch (§ 5T: and exchange), CPSR := SPSR. Exception modes only.	I
	User mode registers		`LDM{IA|IB|DA|DB} Rn, <reglist-PC>^`	Load list of User mode registers from [Rn]. Privileged modes only.	I
Pop			`POP <reglist>`	Canonical form of `LDM SP!, <reglist>`	N
Load exclusive	Semaphore operation	6	`LDREX Rd, [Rn]`	Rd := [Rn], tag address as exclusive access. Outstanding tag set if not shared address. Rd, Rn not PC.	
	Halfword or Byte	T2	`LDREX{H|B} Rd, [Rn]`	Rd[15:0] := [Rn] or Rd[7:0] := [Rn], tag address as exclusive access. Outstanding tag set if not shared address. Rd, Rn not PC.	
	Doubleword	T2	`LDREXD Rd1, Rd2, [Rn]`	Rd1 := [Rn], Rd2 := [Rn+4], tag addresses as exclusive access Outstanding tags set if not shared addresses. Rd1, Rd2, Rn not PC.	9
Store multiple	Push, or Block data store		`STM{IA|IB|DA|DB} Rn{!}, <reglist>`	Store list of registers to [Rn]	N, I
	User mode registers		`STM{IA|IB|DA|DB} Rn{!}, <reglist>^`	Store list of User mode registers to [Rn]. Privileged modes only.	I
Push			`PUSH <reglist>`	Canonical form of `STMDB SP!, <reglist>`	N
Store exclusive	Semaphore operation	6	`STREX Rd, Rm, [Rn]`	If allowed, [Rn] := Rm, clear exclusive tag, Rd := 0. Else Rd := 1. Rd, Rm, Rn not PC.	
	Halfword or Byte	T2	`STREX{H|B} Rd, Rm, [Rn]`	If allowed, [Rn] := Rm[15:0] or [Rn] := Rm[7:0], clear exclusive tag, Rd := 0. Else Rd := 1 Rd, Rm, Rn not PC.	
	Doubleword	T2	`STREXD Rd, Rm1, Rm2, [Rn]`	If allowed, [Rn] := Rm1, [Rn+4] := Rm2, clear exclusive tags, Rd := 0. Else Rd := 1 Rd, Rm1, Rm2, Rn not PC.	9
Clear exclusive		T2	`CLREX`	Clear local processor exclusive tag	C
Preload word, byte or halfword	Immediate offset	5TE	`PLD [Rn {, #<offset>}]`	Preload [address, 32]	1, C
	Register offset	5TE	`PLD [Rn, +/-Rm {, <opsh>}]`	Preload [address, 32]	3, C
	PC-relative	5TE	`PLD <label>`	Preload [label, 32]	5, C

Availability and range of options for Load, Store, and Preload operations

Note	ARM Word, B, D	ARM SB, H, SH	ARM T, BT	Thumb-2 Word, B, SB, H, SH, D	Thumb-2 T, BT, SBT, HT, SHT
1	offset: – 4095 to +4095	offset: –255 to +255	Not available	offset: –255 to +255 if writeback, –255 to +4095 otherwise	offset: 0 to +255, writeback not allowed
2	offset: – 4095 to +4095	offset: –255 to +255	offset: – 4095 to +4095	offset: –255 to +255	Not available
3	Full range of `{, <opsh>}`	`{, <opsh>}` not allowed	Not available	`<opsh>` restricted to `LSL #<sh>`, `<sh>` range 0 to 3	Not available
4	Full range of `{, <opsh>}`	`{, <opsh>}` not allowed	Full range of `{, <opsh>}`	Not available	Not available
5	label within +/– 4092 of current instruction	Not available	Not available	label within +/– 4092 of current instruction	Not available
6	offset: –255 to +255	-	-	offset: –1020 to +1020, must be multiple of 4.	-
7	`{, <opsh>}` not allowed	-	-	Not available	-
8	label within +/– 252 of current instruction	-	-	Not available	-
9	Rd1 even, and not r14, Rd2 == Rd1 + 1.	-	-	Rd1 != PC, Rd2 != PC	-

ARM Instruction Set
Quick Reference Card

Coprocessor operations	§	Assembler		Action	Notes
Data operations		`CDP <copr>, <op1>, CRd, CRn, CRm{, <op2>}`		Coprocessor defined	
Alternative data operations	5	`CDP2 <copr>, <op1>, CRd, CRn, CRm{, <op2>}`		Coprocessor defined	C
Move to ARM register from coprocessor		`MRC <copr>, <op1>, Rd, CRn, CRm{, <op2>}`		Coprocessor defined	
Alternative move	5	`MRC2 <copr>, <op1>, Rd, CRn, CRm{, <op2>}`		Coprocessor defined	C
Two ARM register move	5E*	`MRRC <copr>, <op1>, Rd, Rn, CRm`		Coprocessor defined	
Alternative two ARM register move	6	`MRRC2 <copr>, <op1>, Rd, Rn, CRm`		Coprocessor defined	C
Move to coproc from ARM reg		`MCR <copr>, <op1>, Rd, CRn, CRm{, <op2>}`		Coprocessor defined	
Alternative move	5	`MCR2 <copr>, <op1>, Rd, CRn, CRm{, <op2>}`		Coprocessor defined	C
Two ARM register move	5E*	`MCRR <copr>, <op1>, Rd, Rn, CRm`		Coprocessor defined	
Alternative two ARM register move	6	`MCRR2 <copr>, <op1>, Rd, Rn, CRm`		Coprocessor defined	C
Loads and stores, pre-indexed		`<op> <copr>, CRd, [Rn, #+/-<offset8*4>]{!}`	op: `LDC` or `STC`. offset: multiple of 4 in range 0 to 1020.	Coprocessor defined	
Alternative loads and stores, pre-indexed	5	`<op>2 <copr>, CRd, [Rn, #+/-<offset8*4>]{!}`	op: `LDC` or `STC`. offset: multiple of 4 in range 0 to 1020.	Coprocessor defined	C
Loads and stores, zero offset		`<op> <copr>, CRd, [Rn] {, 8-bit copro. option}`	op: `LDC` or `STC`.	Coprocessor defined	
Alternative loads and stores, zero offset	5	`<op>2 <copr>, CRd, [Rn] {, 8-bit copro. option}`	op: `LDC` or `STC`.	Coprocessor defined	C
Loads and stores, post-indexed		`<op> <copr>, CRd, [Rn], #+/-<offset8*4>`	op: `LDC` or `STC`. offset: multiple of 4 in range 0 to 1020.	Coprocessor defined	
Alternative loads and stores, post-indexed	5	`<op>2 <copr>, CRd, [Rn], #+/-<offset8*4>`	op: `LDC` or `STC`. offset: multiple of 4 in range 0 to 1020.	Coprocessor defined	C

Miscellaneous operations		§	Assembler	Action	Notes
Swap word			`SWP Rd, Rm, [Rn]`	temp := [Rn], [Rn] := Rm, Rd := temp.	D
Swap byte			`SWPB Rd, Rm, [Rn]`	temp := ZeroExtend([Rn][7:0]), [Rn][7:0] := Rm[7:0], Rd := temp	D
Store return state		6	`SRS{IA|IB|DA|DB} SP{!}, #<p_mode>`	[SPm] := LR, [SPm + 4] := CPSR	C, I
Return from exception		6	`RFE{IA|IB|DA|DB} Rn{!}`	PC := [Rn], CPSR := [Rn + 4]	C, I
Breakpoint		5	`BKPT <imm16>`	Prefetch abort *or* enter debug state. 16-bit bitfield encoded in instruction.	C, N
Secure Monitor Interrupt		Z	`SMI <imm16>`	Secure Monitor interrupt exception. 16-bit bitfield encoded in instruction.	
Software interrupt			`SWI <imm24>`	Software interrupt exception. 24-bit bitfield encoded in instruction.	N
No operation		6	`NOP`	None, might not even consume any time.	N
Hints	Set event	T2	`SEV`	Signal event in multiprocessor system. NOP if not implemented.	N
	Wait for event	T2	`WFE`	Wait for event, IRQ, FIQ, Imprecise abort, or Debug entry request. NOP if not implemented.	N
	Wait for interrupt	T2	`WFI`	Wait for IRQ, FIQ, Imprecise abort, or Debug entry request. NOP if not implemented.	N
	Yield	T2	`YIELD`	Yield control to alternative thread. NOP if not implemented.	N

Notes			
N	Some or all forms of this instruction are 16-bit (Narrow) instructions in Thumb-2 code. For details see the *Thumb 16-bit Instruction Set (UAL) Quick Reference Card.*		
U	This instruction is not allowed in an IT block. Condition codes are not allowed for this instruction in either ARM or Thumb state.		
Q	This instruction sets the Q flag if saturation (addition or substraction) or overflow (multiplication) occurs. The Q flag is read and reset using `MRS` and `MSR`.		
G	This instruction updates the four GE flags in the CPSR based on the results of the individual operations.		
A	This instruction is not available in Thumb state.	**T**	This instruction is not available in ARM state.
S	The S modifier is not available in the Thumb-2 instruction.	**I**	IA is the default, and is normally omitted.
C	Condition codes are not allowed for this instruction in ARM state.	**B**	This instruction can be conditional in Thumb state without having to be in an IT block.
D	Deprecated. Use `LDREX` and `STREX` instead.	**P**	Rn can be the PC in Thumb state in this instruction.

ARM Instruction Set
Quick Reference Card

ARM architecture versions	
n	ARM architecture version *n* and above
*n*T, *n*J	T or J variants of ARM architecture version *n* and above.
M	ARM v3M, and 4 and above, except xM variants
5E	ARM v5E, and 6 and above
5E*	ARM v6 and above, and 5E except xP variants
T2	All Thumb-2 versions of ARM v6 and above
Z	All Security extension versions of ARMv6 and above
XS	XScale coprocessor instruction

Flexible Operand 2	
Immediate value	`#<imm8m>`
Register, optionally shifted by constant (see below)	`Rm {, <opsh>}`
Register, logical shift left by register	`Rm, LSL Rs`
Register, logical shift right by register	`Rm, LSR Rs`
Register, arithmetic shift right by register	`Rm, ASR Rs`
Register, rotate right by register	`Rm, ROR Rs`

Register, optionally shifted by constant		
(No shift)	`Rm`	Same as `Rm, LSL #0`
Logical shift left	`Rm, LSL #<shift>`	Allowed shifts 0-31
Logical shift right	`Rm, LSR #<shift>`	Allowed shifts 1-32
Arithmetic shift right	`Rm, ASR #<shift>`	Allowed shifts 1-32
Rotate right	`Rm, ROR #<shift>`	Allowed shifts 1-31
Rotate right with extend	`Rm, RRX`	

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		
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)

All ARM instructions (except those with Note C or Note U) can have any one of these condition codes after the instruction mnemonic (that is, before the first space in the instruction as shown on this card). This condition is encoded in the instruction.

All Thumb-2 instructions (except those with Note U) can have any one of these condition codes after the instruction mnemonic. This condition is encoded in a preceding IT instruction (except in the case of conditional Branch instructions). Condition codes in instructions must match those in the preceding IT instruction.

On processors without Thumb-2, the only Thumb instruction that can have a condition code is `B <label>`.

Processor Modes	
16	User
17	FIQ Fast Interrupt
18	IRQ Interrupt
19	Supervisor
23	Abort
27	Undefined
31	System

Prefixes for Parallel Instructions	
`S`	Signed arithmetic modulo 2^8 or 2^{16}, sets CPSR GE bits
`Q`	Signed saturating arithmetic
`SH`	Signed arithmetic, halving results
`U`	Unsigned arithmetic modulo 2^8 or 2^{16}, sets CPSR GE bits
`UQ`	Unsigned saturating arithmetic
`UH`	Unsigned arithmetic, halving results

Document Number

ARM QRC 0001I

Change Log

Issue	Date	Change	Issue	Date	Change
A	June 1995	First Release	B	Sept 1996	Second Release
C	Nov 1998	Third Release	D	Oct 1999	Fourth Release
E	Oct 2000	Fifth Release	F	Sept 2001	Sixth Release
G	Jan 2003	Seventh Release	H	Oct 2003	Eighth Release
I	Dec 2004	Ninth Release	J	May 2005	RVCT 2.2 SP1
K	March 2006	RVCT 3.0			