ARM Instruction Set
Quick Reference Card

Key to Tables	
{cond}	Refer to Table **Condition Field {cond}**
<Oprnd2>	Refer to Table **Oprnd2**
{field}	Refer to Table **Field**
S	Sets condition codes (optional)
B	Byte operation (optional)
H	Halfword operation (optional)
T	Forces address translation. Cannot be used with pre-indexed addresses
<a_mode2>	Refer to Table **Addressing Mode 2**
<a_mode2P>	Refer to Table **Addressing Mode 2 (Privileged)**
<a_mode3>	Refer to Table **Addressing Mode 3**
<a_mode4L>	Refer to Table **Addressing Mode 4 (Load)**
<a_mode4S>	Refer to Table **Addressing Mode 4 (Store)**
<a_mode5>	Refer to Table **Addressing Mode 5**
#32bit_Imm	A 32-bit constant, formed by right-rotating an 8-bit value by an even number of bits
<reglist>	A comma-separated list of registers, enclosed in braces ({ and })

Operation		Assembler	S updates	Action	Notes
Move	Move	MOV{cond}{S} Rd, <Oprnd2>	N Z C	Rd:= <Oprnd2>	
	NOT	MVN{cond}{S} Rd, <Oprnd2>	N Z C	Rd:= 0xFFFFFFFF EOR <Oprnd2>	
	SPSR to register	MRS{cond} Rd, SPSR		Rd:= SPSR	*Architecture 3, 3M and 4 only*
	CPSR to register	MRS{cond} Rd, CPSR		Rd:= CPSR	*Architecture 3, 3M and 4 only*
	register to SPSR	MSR{cond} SPSR{field}, Rm		SPSR:= Rm	*Architecture 3, 3M and 4 only*
	register to CPSR	MSR{cond} CPSR{field}, Rm		CPSR:= Rm	*Architecture 3, 3M and 4 only*
	immediate to SPSR flags	MSR{cond} SPSR_f, #32bit_Imm		SPSR:= #32-bit_Imm	*Architecture 3, 3M and 4 only*
	immediate to CPSR flags	MSR{cond} CPSR_f, #32bit_Imm		CPSR:= #32-bit_Imm	*Architecture 3, 3M and 4 only*
ALU	Arithmetic				
	Add	ADD{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd:= Rn + <Oprnd2>	
	with carry	ADC{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd:= Rn + <Oprnd2> + Carry	
	Subtract	SUB{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd:= Rn - <Oprnd2>	
	with carry	SBC{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd:= Rn - <Oprnd2> - NOT(Carry)	
	reverse subtract	RSB{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd:= <Oprnd2> - Rn	
	reverse subtract with carry	RSC{cond}{S} Rd, Rn, <Oprnd2>	N Z C V	Rd:= <Oprnd2> - Rn - NOT(Carry)	
	Multiply	MUL{cond}{S} Rd, Rm, Rs	N Z	Rd:= Rm * Rs	*Not in Architecture 1*
	accumulate	MLA{cond}{S} Rd, Rm, Rs, Rn	N Z	Rd:= (Rm * Rs) + Rn	*Not in Architecture 1*
	unsigned long	UMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z	RdHi:= (Rm*Rs)[63:32] RdLo:= (Rm*Rs)[31:0]	*Architecture 3M and 4 only*
	unsigned accumulate long	UMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z	RdLo:= (Rm*Rs)[31:0] + RdLo RdHi:=(Rm*Rs)[63:32] + RdHi + CarryFrom ((Rm*Rs)[31:0] +RdLo))	*Architecture 3M and 4 only*
	signed long	SMULL{cond}{S} RdLo, RdHi, Rm, Rs	N Z	RdHi:= signed(Rm*Rs)[63:32] RdLo:= signed(Rm*Rs)[31:0]	*Architecture 3M and 4 only*
	signed accumulate long	SMLAL{cond}{S} RdLo, RdHi, Rm, Rs	N Z	RdLo:= signed(Rm*Rs)[31:0] + RdLo RdHi:= signed(Rm*Rs)[63:32] + RdHi + CarryFrom ((Rm*Rs)[31:0] +RdLo))	*Architecture 3M and 4 only*
	Compare	CMP{cond} Rd, <Oprnd2>	N Z C V	CPSR flags:= Rn - <Oprnd2>	
	negative	CMN{cond} Rd, <Oprnd2>	N Z C V	CPSR flags:= Rn + <Oprnd2>	
	Logical				
	Test	TST{cond} Rn, <Oprnd2>	N Z C	CPSR flags:= Rn AND <Oprnd2>	
	Test equivalence	TEQ{cond} Rn, <Oprnd2>	N Z C	CPSR flags:= Rn EOR <Oprnd2>	
	AND	AND{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd:= Rn AND <Oprnd2>	
	EOR	EOR{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd:= Rn EOR <Oprnd2>	
	ORR	ORR{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd:= Rn OR <Oprnd2>	
	Bit Clear	BIC{cond}{S} Rd, Rn, <Oprnd2>	N Z C	Rd:= Rn AND NOT <Oprnd2>	
	Shift/Rotate				See Table **Oprnd2**

ARM Instruction Set
Quick Reference Card

Operation		Assembler	Action	Notes
Branch	Branch	`B{cond} label`	R15:= address of label	Address calculated pc-relative
	with link	`BL{cond} label`	R14:=R15-4, R15:= address of label	
	and exchange instruction set	`BX{cond} Rn`	R15:=Rn, T bit:= Rn[0]	*Architecture 4 with Thumb only* to Thumb state; Rn[0] = 1 to ARM state; Rn[0] =0
Load	Word	`LDR{cond} Rd, <a_mode2>`	Rd:= [address]	
	with user-mode privilege	`LDR{cond}T Rd, <a_mode2P>`		
	Byte	`LDR{cond}B Rd, <a_mode2>`	Rd:= [byte value from address] Loads bits 0 to 7 and sets bits 8-31 to 0	
	with user-mode privilege	`LDR{cond}BT Rd, <a_mode2P>`		
	signed	`LDR{cond}SB Rd, <a_mode3>`	Rd:= [signed byte value from address] Loads bits 0 to 7 and sets bits 8-31 to bit 7	*Architecture 4 only*
	Halfword	`LDR{cond}H Rd, <a_mode3>`	Rd:= [halfword value from address] Loads bits 0 to 15 and sets bits 16-31 to 0	*Architecture 4 only*
	signed	`LDR{cond}SH Rd, <a_mode3>`	Rd:= [signed halfword value from address] Loads bits 0 to 15 and sets bits 16-31 to bit 15	*Architecture 4 only*
	Multiple			
	Block data operations			
	Increment Before	`LDM{cond}IB Rd{!}, <reglist>{^}`		! sets the W bit (updates the base register after the transfer) ^ sets the S bit
	Increment After	`LDM{cond}IA Rd{!}, <reglist>{^}`		
	Decrement Before	`LDM{cond}DB Rd{!}, <reglist>{^}`		
	Decrement After	`LDM{cond}DA Rd{!}, <reglist>{^}`		
	Stack operations	`LDM{cond}<a_mode4L> Rd{!}, <reglist>`	Stack manipulation (pop)	! sets the W bit (updates the base register after the transfer)
	and restore CPSR	`LDM{cond}<a_mode4L> Rd{!}, <reglist+pc>^`		
	User registers	`LDM{cond}<a_mode4L> Rd, <reglist>^`		
Store	Word	`STR{cond} Rd, <a_mode2>`	[address]:= Rd	
	with user-mode privilege	`STR{cond}T Rd, <a_mode2P>`		
	Byte	`STR{cond}B Rd, <a_mode2>`	[address]:= byte value from Rd	
	with user-mode privilege	`STR{cond}BT Rd, <a_mode2P>`		
	Halfword	`STR{cond}H Rd, <a_mode3>`	[address]:= halfword value from Rd	*Architecture 4 only*
	Multiple			
	Block data operations			
	Increment Before	`STM{cond}IB Rd{!}, <reglist>{^}`		! sets the W bit (updates the base register after the transfer) ^ sets the S bit
	Increment After	`STM{cond}IA Rd{!}, <reglist>{^}`		
	Decrement Before	`STM{cond}DB Rd{!}, <reglist>{^}`		
	Decrement After	`STM{cond}DA Rd{!}, <reglist>{^}`		
	Stack operations	`STM{cond}<a_mode4S> Rd{!}, <reglist>`	Stack manipulation (push)	
	User registers	`STM{cond}<a_mode4S> Rd{!}, <reglist>^`		
Swap	Word	`SWP{cond} Rd, Rm, [Rn]`		*Not in Architecture 1 or 2*
	Byte	`SWP{cond}B Rd, Rm, [Rn]`		*Not in Architecture 1 or 2*
Coprocessors	Data operations	`CDP{cond} p<cpnum>, <op1>, CRd, CRn, CRm, <op2>`		*Not in Architecture 1*
	Move to ARM reg from coproc	`MRC{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>`		
	Move to coproc from ARM reg	`MCR{cond} p<cpnum>, <op1>, Rd, CRn, CRm, <op2>`		
	Load	`LDC{cond} p<cpnum>, CRd, <a_mode5>`		
	Store	`STC{cond} p<cpnum>, CRd, <a_mode5>`		
Software Interrupt		`SWI 24bit_Imm`	Causes a software interrupt processor exception	24-bit immediate value encoded within the instruction.

ARM Addressing Modes
Quick Reference Card

Addressing Mode 2	
Immediate offset	`[Rn, #+/-12bit_Offset]`
Register offset	`[Rn, +/-Rm]`
Scaled register offset	`[Rn, +/-Rm, LSL #5bit_shift_imm]`
	`[Rn, +/-Rm, LSR #5bit_shift_imm]`
	`[Rn, +/-Rm, ASR #5bit_shift_imm]`
	`[Rn, +/-Rm, ROR #5bit_shift_imm]`
	`[Rn, +/-Rm, RRX]`
Pre-indexed offset	
Immediate	`[Rn, #+/-12bit_Offset]!`
Register	`[Rn, +/-Rm]!`
Scaled register	`[Rn, +/-Rm, LSL #5bit_shift_imm]!`
	`[Rn, +/-Rm, LSR #5bit_shift_imm]!`
	`[Rn, +/-Rm, ASR #5bit_shift_imm]!`
	`[Rn, +/-Rm, ROR #5bit_shift_imm]!`
	`[Rn, +/-Rm, RRX]!`
Post-indexed offset	
Immediate	`[Rn], #+/-12bit_Offset`
Register	`[Rn], +/-Rm`
Scaled register	`[Rn], +/-Rm, LSL #5bit_shift_imm`
	`[Rn], +/-Rm, LSR #5bit_shift_imm`
	`[Rn], +/-Rm, ASR #5bit_shift_imm`
	`[Rn], +/-Rm, ROR #5bit_shift_imm`
	`[Rn, +/-Rm, RRX]`

Addressing Mode 2 (Privileged)	
Immediate offset	`[Rn, #+/-12bit_Offset]`
Register offset	`[Rn, +/-Rm]`
Scaled register offset	`[Rn, +/-Rm, LSL #5bit_shift_imm]`
	`[Rn, +/-Rm, LSR #5bit_shift_imm]`
	`[Rn, +/-Rm, ASR #5bit_shift_imm]`
	`[Rn, +/-Rm, ROR #5bit_shift_imm]`
	`[Rn, +/-Rm, RRX]`
Post-indexed offset	
Immediate	`[Rn], #+/-12bit_Offset`
Register	`[Rn], +/-Rm`
Scaled register	`[Rn], +/-Rm, LSL #5bit_shift_imm`
	`[Rn], +/-Rm, LSR #5bit_shift_imm`
	`[Rn], +/-Rm, ASR #5bit_shift_imm`
	`[Rn], +/-Rm, ROR #5bit_shift_imm`
	`[Rn, +/-Rm, RRX]`

Addressing Mode 3 - Signed Byte and Halfword Data Transfer	
Immediate offset	`[Rn, #+/-8bit_Offset]`
Pre-indexed	`[Rn, #+/-8bit_Offset]!`
Post-indexed	`[Rn], #+/-8bit_Offset`
Register	`[Rn, +/-Rm]`
Pre-indexed	`[Rn, +/-Rm]!`
Post-indexed	`[Rn], +/-Rm`

Addressing Mode 5 - Coprocessor Data Transfer	
Immediate offset	`[Rn, #+/-(8bit_Offset*4)]`
Pre-indexed	`[Rn, #+/-(8bit_Offset*4)]!`
Post-indexed	`[Rn], #+/-(8bit_Offset*4)`

Addressing Mode 4 (Load)			
Addressing Mode		**Stack Type**	
`IA`	Increment After	`FD`	Full Descending
`IB`	Increment Before	`ED`	Empty Descending
`DA`	Decrement After	`FA`	Full Ascending
`DB`	Decrement Before	`EA`	Empty Ascending

Addressing Mode 4 (Store)			
Addressing Mode		**Stack Type**	
`IA`	Increment After	`EA`	Empty Ascending
`IB`	Increment Before	`FA`	Full Ascending
`DA`	Decrement After	`ED`	Empty Descending
`DB`	Decrement Before	`FD`	Full Descending

Oprnd2	
Immediate value	`#32bit_Imm`
Logical shift left	`Rm LSL #5bit_Imm`
Logical shift right	`Rm LSR #5bit_Imm`
Arithmetic shift right	`Rm ASR #5bit_Imm`
Rotate right	`Rm ROR #5bit_Imm`
Register	`Rm`
Logical shift left	`Rm LSL Rs`
Logical shift right	`Rm LSR Rs`
Arithmetic shift right	`Rm ASR Rs`
Rotate right	`Rm ROR Rs`
Rotate right extended	`Rm RRX`

Field		
Suffix	**Sets**	
`_c`	Control field mask bit	(bit 3)
`_f`	Flags field mask bit	(bit 0)
`_s`	Status field mask bit	(bit 1)
`_x`	Extension field mask bit	(bit 2)

Condition Field {cond}	
Suffix	**Description**
`EQ`	Equal
`NE`	Not equal
`CS`	Unsigned higher or same
`CC`	Unsigned lower
`MI`	Negative
`PL`	Positive or zero
`VS`	Overflow
`VC`	No overflow
`HI`	Unsigned higher
`LS`	Unsigned lower or same
`GE`	Greater or equal
`LT`	Less than
`GT`	Greater than
`LE`	Less than or equal
`AL`	Always