ARM Instruction Set Summary (ARM7, ARM7TDMI, ARM9, and ARM9E core)

Mnemonic	Operation	Description
ADC	Rd := Rn + Op2 + C	Add with carry
ADD	Rd := Rn + Op2	Add
AND	Rd := Rn AND Op2	AND
B	R15 := address	Branch
BIC	Rd := Rn AND NOT Op2	Bit Clear
BKPT[1]	Enter debug state	Breakpoint
BL	R14 := address of next instruction, R15 :=address	Branch with Link
BLX[1]	R14 := address of next instruction R15 := Rm[31:1], change to Thumb if address bit 0 is 1	Branch with Link and Exchange
BX	R15 := Rn, change to Thumb if address bit 0 is 1	Branch and Exchange
CDP	Coprocessor specific	Coprocessor Data Processing
CDP2[1]	Coprocessor specific	Alternative Data Operations
CLZ[1]	Rd := number of leading zeroes in Rm	Count Leading Zeroes
CMN	CPSR flags := Rn + Op2	Compare Negative
CMP	CPSR flags := Rn - Op2	Compare
EOR	Rd:= Rn EOR Op2	Exclusive OR
LDC	Coprocessor load	Load Coprocessor from Memory
LDC2[1]	Coprocessor specific	Alternative Loads
LDM	Stack manipulation (Pop)	Load multiple Registers
LDR	Rd := [address][31:0]	Load 32-bit word from memory.
LDRB	Rd := ZeroExtend ([address][7:0])	Load register byte value to Memory.
LDRH	Rd := ZeroExtend ([address][15:0])	Load register 16-bit halfword value to Memory.
LDRSB	Rd := SignExtend ([address][7:0])	Load register signed byte value to Memory.
LDRSH	Rd := SignExtend ([address][15:0])	Load register signed halfword from Memory.
LDRD[1]	Rd := [address][31:0] Rd+1 := [address+4][31:0]	Load register pair **Rd** and **Rd+1**
MCR	cRn:=rRn {<op>cRm}	Move CPU register to coprocessor register
MCR2[1]	Coprocessor specific	Alternative move
MCRR[1]	Coprocessor specific	Two ARM register move
MLA	Rd := (Rm * Rs) + Rn	Multiply Accumulate
MOV	Rd := Op2	Move register or constant
MRC	Rn := cRn{<op>cRm}	Move from coprocessor register to CPU register

MRC2[1]	Coprocessor specific	Alternative move
MRRC[1]	Coprocessor specific	Two ARM register move
MRS	Rn := PSR	Move PSR status flags to register
MSR	PSR := Rm	Move register to PSR status flags
MUL	Rd := Rm * Rs	Multiply
MVN	Rd := NOT Rm	Move inverted register or constant
NOP[1]	None	No operation
ORR	Rd:=Rn OR Op2	OR
PLD[1]	Memory may prepare to load from address	Memory system hint
QADD[1]	Rd := SAT (Rm + Rn)	Saturated add
QDADD[1]	Rd := SAT (Rm + SAT (Rn * 2))	Saturated add double
QSUB[1]	Rd := SAT (Rm - Rn)	Saturated subtract
QDSUB[1]	Rd := SAT (Rm - SAT (Rn*2))	Saturated subtract double
RSB	Rd := Op2 - Rn	Reverse Subtract
RSC	Rd := Op2 - Rn - 1+Carry	Reverse Subtract with Carry
SBC	Rd := Rn - Op2 - 1+Carry	Subtract with Carry
SMULxy[1]	Rd := Rm[x] * Rs[y]	Saturating arithmetic
SMULWy[1]	Rd := (Rm * Rs[y])[47:16]	Saturating arithmetic
SMLAxy[1]	Rd := Rn + Rm[x] * Rs[y]	Saturating arithmetic
SMLAWy[1]	Rd := Rn+(Rm * Rs[y])[47:16]	Saturating arithmetic
SMLALxy[1]	RdHi,RdLo := RdHi,RdLo + Rm[x]*Rs[y]	Saturating arithmetic
STC	address:= CRn	Store Coprocessor register to memory
STC2[1]	Coprocessor specific	Alternative stores
STM	stack manipulation (Push)	Store Multiple
STR	<address>:=Rd	Store register to memory
STRB	[address][7:0] := Rd[7:0]	Store register byte value to Memory.
STRH	[address][15:0] :=Rd[15:0]	Store register 16-bit halfword value to Memory
STRD[1]	[address][31:0] := Rd [address+4][31:0] := Rd+1	Store register pair **Rd** and **Rd+1**
SUB	Rd := Rn - Op2	Subtract
SWI	OS call	Software Interrupt
SWP	Rd := [Rn], [Rn] := Rm	Swap register with memory word
SWPB	Rd := ZeroExtended[Rn][7:0], [Rn][7:0] := Rm	Swap register with memory byte
TEQ	CPSR flags:= Rn EOR Op2	Test bitwise equality
TST	CPSR flags:= Rn AND Op2	Test bits

Note: [1] instruction only available in ARM9E devices only.