Abbreviated ARM Instruction Set Summary

Mnemonics	Operands	Description	Operation
Data Movement Instructions			
MOV{S}	Rd,Rm	Copy value from Rm to Rd	Rd ← Rm
MOVW	Rd,#K	Copy 16-bit constant K to Rd (Zero Ext)	Rd ← #K
MOVT	Rd,#K	Copy 16-bit constant K to Rd[31:16]	Rd[31:16] ← #K
STR{B|H}	Rt,[Rn,#$^{+/-}$K]	Store: Regular Immediate Offset Addressing (8-bit K)	[Rn + #$^{+/-}$K] ← Rt
STR{B|H}	Rt,[Rn,#$^{+/-}$K]!	Store: Pre-index Immediate Offset Addressing (8-bit K)	[Rn + #$^{+/-}$K] ← Rt Rn ← Rn + #$^{+/-}$K
STR{B|H}	Rt,[Rn],#$^{+/-}$K	Store: Post-index Immediate Offset Addressing (8-bit K)	[Rn] ← Rt Rn ← Rn + #$^{+/-}$K
STR{B|H}	Rt,[Rn,Rm {,LSL #s}]	Store: Register Offset Addressing (2-bit **s**: i.e. max shift of 3)	[Rn+ (Rm<< #s)]←Rt
LDR{B|H|SB|SH}	Rt,[Rn,#$^{+/-}$K]	Load: Regular Immediate Offset Addressing (8-bit K)	Rt ← [Rn + #$^{+/-}$K]
LDR{B|H|SB|SH}	Rt,[Rn,#$^{+/-}$K]!	Load: Pre-index Immediate Offset Addressing (8-bit K)	Rt ← [Rn + #$^{+/-}$K] Rn ← Rn + #$^{+/-}$K
LDR{B|H|SB|SH}	Rt,[Rn],#$^{+/-}$K	Load: Post-index Immediate Offset Addressing (8-bit K)	Rt ← [Rn] Rn ← Rn + #$^{+/-}$K
LDR{B|H|SB|SH}	Rt,[Rn,Rm {,LSL #s}]	Load: Register Offset Addressing (2-bit **s**: i.e. max shift of 3))	Rt←[Rn+ (Rm<< #s)]
PUSH	Rn	Place Rn on top of the Stack	[SP] ← Rn SP ← SP – 4
POP	Rn	Place the top of Stack into Rn	SP ← SP + 4 Rn ← [SP]

Branch Instructions				
Mnemonics	**Operands**	**Description**	**Operation**	**Branch Condition**
B	label	Unconditional branch	PC ← label	-
BEQ	label	Branch if EQual	if condition, PC←label	Z=1
BNE	label	Branch if Not Equal	if condition, PC←label	Z=0
BLT	label	Branch if Less Than (Signed)	if condition, PC←label	N!=V
BLE	label	Branch if Less than Equal (Signed)	if condition, PC←label	N!=V | Z=1
BGT	label	Branch if Greater Than (Signed)	if condition, PC←label	N=V & Z=0
BGE	label	Branch if Greater than Equal (Signed)	if condition, PC←label	N=V
BLO	label	Branch if LOwer (Unsigned)	if condition, PC←label	C=0
BLS	label	Branch if LOwer or Same (Unsigned)	if condition, PC←label	C=0 | Z=1
BHI	label	Branch if HIgher (Unsigned)	if condition, PC←label	C=1 & Z=0
BHS	label	Branch if Higher or Same (Unsigned)	if condition, PC←label	C=1
BL	label	Update Link Register and Branch	LR←PC+4; PC←label	-
BX	Rn	Branch to address in Rn	PC ←Rn	-

Mnemonics	Operands	Description	Operation
Compare and Test Instructions			
CMP	Rn,Rm	Compare registers, Update Status	Rn – Rm
CMP	Rn,#K	Compare 8-bit K, Update Status	Rn – #K
TST	Rn,Rm	Test registers, Update Status	Rn and Rm
TST	Rn,#K	Test with 8-bit K, Update Status	Rn and #K
TEQ	Rn,Rm	Test Equivalent, Update Status	Rn xor Rm
TEQ	Rn,#K	Test Equivalent 8-bit K, Update Status	Rn xor #K
Logic and Arithmetic Instructions			
ADD{S}	{Rd,} Rn,Rm	Add two registers	Rdn ← Rn + Rm
ADD{S}	{Rd,} Rn,#K	Add register with 8-bit constant K	Rdn ← Rn + #K
ADC{S}	{Rd,} Rn,Rm	Add two registers with Carry (C)	Rdn ← Rn + Rm + C
ADC{S}	{Rd,} Rn,#K	Add register with 8-bit K and Carry(C)	Rdn ← Rn + #K + C
SUB{S}	{Rd,} Rn,Rm	Subtract two registers	Rdn ← Rn - Rm
SUB{S}	{Rd,} Rn,#K	Subtract 8-bit constant K from register	Rdn ← Rn – #K
SBC{S}	{Rd,} Rn,Rm	Subtract two registers with Carry (C)	Rdn ← Rn - Rm - C
SBC{S}	{Rd,} Rn,#K	Subtract 8-bit K from reg with Carry(C)	Rdn ← Rn - #K - C
MUL	{Rd,} Rn,Rm	Multiple regs (Signed or Unsigned)	Rdn ← Rn * Rm
UMULL	RdL,RdH,Rn,Rm	Unsigned multiple (64-bit result)	RdH:RdL← Rn * Rm
SMULL	RdL,RdH,Rn,Rm	Signed multiple (64-bit result)	RdH:RdL← Rn * Rm
AND{S}	{Rd,} Rn,(Rm|#K)	AND registers or 8-bit constant K	Rdn←Rn and (Rm|#K)
ORR{S}	{Rd,} Rn,(Rm|#K)	OR registers or 8-bit constant K	Rdn←Rn or (Rm|#K)
EOR{S}	{Rd,} Rn,(Rm|#K)	Exclusive OR reg or 8-bit constant K	Rdn←Rn xor (Rm|#K)
NEG{S}	{Rd,} Rn	Negate a register	Rdn← -Rn
LSL{S}	{Rd,} Rn,(Rm|#K)	Logical Shift Left	Rdn← Rn << (Rm|#K)
LSR{S}	{Rd,} Rn,(Rm|#K)	Logical Shift Right	Rdn← Rn >> (Rm|#K)
ASR{S}	{Rd,} Rn,(Rm|#K)	Arithmetic Shift Right (sign bit shift in)	Rdn← Rn >> (Rm|#K)
ROR{S}	{Rd,} Rn,(Rm|#K)	ROtate Right (least sig bit shift in)	Rdn← Rn >> (Rm|#K)
Signed and Unsigned (Zero) Extension Instructions			
SXTB	Rd, Rm	Sign extend the lower byte	Rd←SignExt(Rm[7:0])
SXTH	Rd, Rm	Sign extend the lower half-word	Rd←SignExt(Rm[15:0])
UXTB	Rd, Rm	Zero extend the lower byte	Rd←ZeroExt(Rm[7:0])
UXTH	Rd, Rm	Zero extend the lower half-word	Rd←ZeroExt(Rm[15:0])

Note 1: { } indicates optional parameters

Note 2: Rdn: means destination register is Rd or Rn, depending on if optional {Rd,} is specified

Note 3: { | } indicates alternative optional parameters

Note 4: (|) indicates alternative required parameters

Note 5: # indicates a constant

Note 6: Suffixes: B (zero extended byte), H (zero extended half-word: 16-bits),
SB (sign extended byte), SH (sign extended half-word: 16-bits)

Note 7: Specifying an optional S indicates version of instruction that updates the Status Register