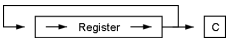
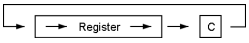
B ARM Instruction Summary

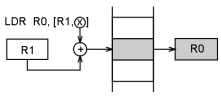
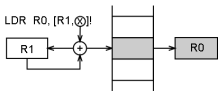
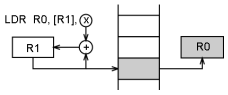
cc: Condition Codes

Generic		*Unsigned*		*Signed*	
CS	Carry Set	HI	Higer Than	GT	Greater Than
CC	Carry Clear	HS	Higer or Same	GE	Greater Than or Equal
EQ	Equal (Zero Set)	LO	Lower Than	LT	Less Than
NE	Not Equal (Zero Clear)	LS	Lower Than or Same	LE	Less Than or Equal
VS	Overflow Set			MI	Minus (Negative)
VC	Overflow Clear			PL	Plus (Positive)

ARM Instructions

Add with Carry	ADC⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* + ⟨*op1*⟩ + CPSR(C)
Add	ADD⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* + ⟨*op1*⟩
Bitwise AND	AND⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* & ⟨*op1*⟩
Branch	B⟨*cc*⟩	⟨*offset*⟩	⟨*cc*⟩: PC	← PC + ⟨*offset*⟩
Branch and Link	BL⟨*cc*⟩	⟨*offset*⟩	⟨*cc*⟩: LR	← PC + 8
			⟨*cc*⟩: PC	← PC + ⟨*offset*⟩
Compare	CMP⟨*cc*⟩	R*n*, ⟨*op1*⟩	⟨*cc*⟩: CSPR	← (R*n* - ⟨*op1*⟩)
Exclusive OR	EOR⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* ⊕ ⟨*op1*⟩
Load Register	LDR⟨*cc*⟩	R*d*, ⟨*op2*⟩	⟨*cc*⟩: R*d*	← M(⟨*op2*⟩)
Load Register Byte	LDR⟨*cc*⟩B	R*d*, ⟨*op2*⟩	⟨*cc*⟩: R*d*(7:0)	← M(⟨*op2*⟩)
			⟨*cc*⟩: R*d*(31:8)	← 0
Move	MOV⟨*cc*⟩⟨*S*⟩	R*d*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← ⟨*op1*⟩
Move Negative	MVN⟨*cc*⟩⟨*S*⟩	R*d*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← $\overline{\langle op1 \rangle}$
Bitwise OR	ORR⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* | ⟨*op1*⟩
Subtract with Carry	SBC⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* - ⟨*op1*⟩ - $\overline{\text{CPSR(C)}}$
Store Register	STR⟨*cc*⟩	R*d*, ⟨*op2*⟩	⟨*cc*⟩: M(⟨*op2*⟩)	← R*d*
Store Register Byte	STR⟨*cc*⟩⟨*S*⟩	R*d*, ⟨*op2*⟩	⟨*cc*⟩: M(⟨*op2*⟩)	← R*d*(7:0)
Subtract	SUB⟨*cc*⟩⟨*S*⟩	R*d*, R*n*, ⟨*op1*⟩	⟨*cc*⟩: R*d*	← R*n* - ⟨*op1*⟩
Software Interrupt	SWI⟨*cc*⟩	⟨*value*⟩		
Swap	SWP⟨*cc*⟩	R*d*, R*m*, [R*n*]	⟨*cc*⟩: R*d*	← M(R*n*)
			⟨*cc*⟩: M(R*n*)	← R*m*
Swap Byte	SWP⟨*cc*⟩B	R*d*, R*m*, [R*n*]	⟨*cc*⟩: R*d*(7:0)	← M(R*n*)(7:0)
			⟨*cc*⟩: M(R*n*)(7:0)	← R*m*(7:0)

op1: Data Access		
Immediate	#⟨*value*⟩	⟨*op1*⟩ ← IR(value)
Register	R*m*	⟨*op1*⟩ ← R*m*
		
Logical Shift Left Immediate	R*m*, LSL #⟨*value*⟩	⟨*op1*⟩ ← R*m* ≪ IR(value)
Logical Shift Left Register	R*m*, LSL R*s*	⟨*op1*⟩ ← R*m* ≪ R*s*(7:0)
		
Logical Shift Right Immediate	R*m*, LSR #⟨*value*⟩	⟨*op1*⟩ ← R*m* ≫ IR(value)
Logical Shift Right Register	R*m*, LSR R*s*	⟨*op1*⟩ ← R*m* ≫ R*s*(7:0)
		
Arithmetic Shift Right Immediate	R*m*, ASR #⟨*value*⟩	⟨*op1*⟩ ← R*m* +≫ IR(value)
Arithmetic Shift Right Register	R*m*, ASR R*s*	⟨*op1*⟩ ← R*m* +≫ R*s*(7:0)
		
Rotate Right Immediate	R*m*, ROR #⟨*value*⟩	⟨*op1*⟩ ← R*m* ⋙ ⟨*value*⟩
Rotate Right Register	R*m*, ROR R*s*	⟨*op1*⟩ ← R*m* ⋙ R*s*(4:0)
		
Rotate Right with Extend	R*m*, RRX	⟨*op1*⟩ ← C ⋙ R*m* ⋙ C

op2: Memory Access		
		
Immediate Offset	[R*n*, #±⟨*value*⟩]	⟨*op2*⟩ ← R*n* + IR(value)
Register Offset	[R*n*, R*m*]	⟨*op2*⟩ ← R*n* + R*m*
Scaled Register Offset	[⟨*Rn*⟩, R*m*, ⟨*shift*⟩ #⟨*value*⟩]	⟨*op2*⟩ ← R*n* + (R*m* *shift* IR(value))
		
Immediate Pre-indexed	[R*n*, #±⟨*value*⟩] !	⟨*op2*⟩ ← R*n* + IR(value) R*n* ← ⟨*op2*⟩
Register Pre-indexed	[R*n*, R*m*] !	⟨*op2*⟩ ← R*n* + R*m* R*n* ← ⟨*op2*⟩
Scaled Register Pre-indexed	[R*n*, R*m*, ⟨*shift*⟩ #⟨*value*⟩] !	⟨*op2*⟩ ← R*n* + (R*m* *shift* IR(value)) R*n* ← ⟨*op2*⟩
		
Immediate Post-indexed	[R*n*], #±⟨*value*⟩	⟨*op2*⟩ ← R*n* R*n* ← R*n* + IR(value)
Register Post-indexed	[R*n*], R*m*	⟨*op2*⟩ ← R*n* R*n* ← R*n* + R*m*
Scaled Register Post-indexed	[R*n*], R*m*, ⟨*shift*⟩ #⟨*value*⟩	⟨*op2*⟩ ← R*n* R*n* ← R*n* + R*m* *shift* IR(value)

Where ⟨*shift*⟩ is one of: LSL, LSR, ASR, ROR or RRX and has the same effect as for ⟨*op1*⟩