ARM2 Instructions

Inspired by 6502 Instructions by Beagle Bros

Any instruction can be conditional, e.g. MOVEQ R0, R1
Some instructions can optionally update the
status flags, e.g. ADDS R0, R1, R2
Use ! to update Rn when pre-indexing, e.g. LDR R0, [R1, #4]!

Status
***** may change
_ no change
X scrambled

Conditions
AL CC CS EQ
GE GT HI LE
LS LT MI NE
NV PL VC VS

Abbreviations
Rd, Rn, Rm, Rs any register
#imm signed expression shiftable
into an 8-bit value
shift any of ASL, LSL, LSR, ASR or ROR
cnt shift count in range of 1..31
addr 26 bit address
mode any of IA, IB, DA, DB (or EA, ED, FA or FD)
reg_list e.g. R2, R4-R6
off offset in range of -4095..4095

Name and description	Addressing modes	Status N Z C V I F
ADC		
Arithmetic add with carry	ADC Rd, Rn, #imm ADC Rd, Rn, Rm ADC Rd, Rn, Rm shift #cnt ADC Rd, Rn, Rm shift Rs ADC Rd, Rn, Rm RRX	* * * * _ _ (if S)
ADD		

Arithmetic add	ADD Rd, Rn, #imm
	ADD Rd, Rn, Rm
	ADD Rd, Rn, Rm shift #cnt
	ADD Rd, Rn, Rm shift Rs
	ADD Rd, Rn, Rm RRX
		* * * * _ _
		(if S)

AND		
Logical AND	AND Rd, Rn, #imm AND Rd, Rn, Rm AND Rd, Rn, Rm shift #cnt AND Rd, Rn, Rm shift Rs AND Rd, Rn, Rm RRX	* * * _ _ _ (if S)
B		
Branch	B addr	_ _ _ _ _ _
BIC		
Bit clear	BIC Rd, Rn, #imm BIC Rd, Rn, Rm BIC Rd, Rn, Rm shift #cnt BIC Rd, Rn, Rm shift Rs BIC Rd, Rn, Rm RRX	* * * _ _ _ (if S)
BL		
Branch with link (R14 ← PC)	BL addr	_ _ _ _ _ _
CMN		
Set negative compare	CMN Rn, #imm CMN Rn, Rm CMN Rn, Rm shift #cnt CMN Rn, Rm shift Rs CMN Rn, Rm RRX	* * * * _ _
CMP		

Arithmetic comparison	CMP Rn, #imm
	CMP Rn, Rm
	CMP Rn, Rm shift #cnt
	CMP Rn, Rm shift Rs
	CMP Rn, Rm RRX
		* * * * _ _

EOR		
Logical exclusive OR	EOR Rd, Rn, #imm EOR Rd, Rn, Rm EOR Rd, Rn, Rm shift #cnt EOR Rd, Rn, Rm shift Rs EOR Rd, Rn, Rm RRX	* * * _ _ _ (if S)
LDM		
Load multiple registers	LDMmode Rn, { reg_list }	_ _ _ _ _ _
LDR		

Load register from memory	LDR Rd, [Rn, #off]
	LDR Rd, [Rn, Rm]
	LDR Rd, [Rn, Rm shift #cnt]
	LDR Rd, [Rd], #off
	LDR Rd, [Rd], Rm
	LDR Rd, [Rd], Rm shift #cnt
		_ _ _ _ _ _

MLA		
Multiply and accumulate	MLA Rd, Rm, Rs, Rn	* * X _ _ _ (if S)
MOV		

Move register or constant	MOV Rd, #imm
	MOV Rd, Rm
	MOV Rd, Rm shift #cnt
	MOV Rd, Rm shift Rs
	MOV Rd, Rm RRX
		* * * _ _ _
		(if S)

MUL		
Multiply	MUL Rd, Rm, Rs	* * X _ _ _ (if S)
MVN		
Move complement of register	MVN Rd, #imm MVN Rd, Rm MVN Rd, Rm shift #cnt MVN Rd, Rm shift Rs MVN Rd, Rm RRX	* * * _ _ _ (if S)
ORR		
Logical OR	ORR Rd, Rn, #imm ORR Rd, Rn, Rm ORR Rd, Rn, Rm shift #cnt ORR Rd, Rn, Rm shift Rs ORR Rd, Rn, Rm RRX	* * * _ _ _ (if S)
RSB		
Reverse-operand subtract	RSB Rd, Rn, #imm RSB Rd, Rn, Rm RSB Rd, Rn, Rm shift #cnt RSB Rd, Rn, Rm shift Rs RSB Rd, Rn, Rm RRX	* * * * _ _ (if S)
RSC		
Reverse-operand subtract with carry	RSC Rd, Rn, #imm RSC Rd, Rn, Rm RSC Rd, Rn, Rm shift #cnt RSC Rd, Rn, Rm shift Rs RSC Rd, Rn, Rm RRX	* * * * _ _ (if S)
SBC		
Subtract with carry	SBC Rd, Rn, #imm SBC Rd, Rn, Rm SBC Rd, Rn, Rm shift #cnt SBC Rd, Rn, Rm shift Rs SBC Rd, Rn, Rm RRX	* * * * _ _ (if S)
STM		
Store multiple registers	STMmode Rn, { reg_list }	_ _ _ _ _ _
STR		
Store register to memory	STR Rd, [Rn, #off] STR Rd, [Rn, Rm] STR Rd, [Rn, Rm shift #cnt] STR Rd, [Rd], #off STR Rd, [Rd], Rm STR Rd, [Rd], Rm shift #cnt	_ _ _ _ _ _
SUB		

Subtract	SUB Rd, Rn, #imm
	SUB Rd, Rn, Rm
	SUB Rd, Rn, Rm shift #cnt
	SUB Rd, Rn, Rm shift Rs
	SUB Rd, Rn, Rm RRX
		* * * * _ _
		(if S)

SWI		
Software interrupt	SWI operand	_ _ _ _ _ _
TEQ		
Set condition codes via XOR	TEQ Rn, #imm TEQ Rn, Rm TEQ Rn, Rm shift #cnt TEQ Rn, Rm shift Rs TEQ Rn, Rm RRX	* * * _ _ _
TST		
Set condition codes via AND	TST Rn, #imm TST Rn, Rm TST Rn, Rm shift #cnt TST Rn, Rm shift Rs TST Rn, Rm RRX	* * * _ _ _