

Opcode	Operation
ADC<cond><S> Rd, Rn, <sh_op>	Rd = Rn + <s_op> + C
ADD<cond><S> Rd, Rn, <sh_op>	Rd = Rn + <s_op>
AND<cond><S> Rd, Rn, <sh_op>	Rd = Rn & <s_op>
B<cond> <target_addr>	PC = PC + <offset>
BIC<cond><S> Rd, Rn, <sh_op>	Rd = Rn & !<s_op>
BL<cond> <target_addr>	LR = PC+4; PC = PC + <offset>
BLX<cond> Rm	PC = Rm; Mode=THUMB
CDP<cond> <p_cp#>, , CRd, CRn, CRm, <o2>	execute coprocessor opcode
CMN<cond> Rn, <sh_op>	<flags> = Rn + <s_op>
CMP<cond> Rn, <sh_op>	<flags> = Rn - <s_op>
EOR<cond><S> Rd, Rn, <sh_op>	Rd = Rn ^ <s_op>
LDC<cond> <p_cp num>, CRd, #	load coprocessor register with #
LDM<cond><admm> Rm, {reg list}^	special, see doc
LDM<cond><admm> Rm<!,>, {reg list}	for each in <reglist> = [Rn+4]
LDM<cond><admm> Rm<!,>, {reg list}^	special, see doc
LDR<cond> Rd, Rn, #	Rd = [Rn + #]
LDR<cond>B Rd, Rn, #	Rd = [Rn + #]
LDR<cond>BT Rd, Rn, #	Rd = [Rn + #]
LDR<cond>H Rd, <address>	Rd = [address]
LDR<cond>SB Rd, <address>	Rd = [address]
LDR<cond>SH Rd, <address>	Rd = [address]
LDR<cond>T Rd, Rn, #	Rd = [Rn + #]
MCR<cond> <p_cp#>, , Rd, CRn, CRm, <o2>	move from co-cpu reg to ARM reg
MLA<cond><S> Rd, Rn, Rm, Rs, Rn	Rd = Rm * Rs + Rn
MOV<cond><S> Rd, <sh_op>	Rd = <s_op>
MRC<cond> <p_cp#>, , Rd, CRn, CRm, <o2>	move from ARM reg to co-cpu reg
MRS<cond> Rd, CPSR	Rd = CPSR
MRS<cond> Rd, SPSR	Rd = SPSR
MSR<cond> CPSR, <fields>, Rm	CPSR = Rm (fields pick bytes to copy)
MSR<cond> CPSR, f, #	CPSR = # (fields pick bytes to copy)
MSR<cond> SPSR, <fields>, Rm	SPSR = Rm (fields pick bytes to copy)
MSR<cond> SPSR, f, #	SPSR = # (fields pick bytes to copy)
MUL<cond><S> Rd, Rm, Rs	Rd = Rm * Rs
MVN<cond><S> Rd, <sh_op>	Rd = ~<s_op>
ORR<cond><S> Rd, Rn, <sh_op>	Rd = Rn <s_op>
RSB<cond><S> Rd, Rn, <sh_op>	Rd = <s_op> - Rn
RSC<cond><S> Rd, Rn, <sh_op>	Rd = <s_op> - Rn + C
SBC<cond><S> Rd, Rn, <sh_op>	Rd = Rn - <s_op> + C
SMMLAL<cond><S> RdLo, RdHi, Rm, Rs	RdHi..RdLo = Rm*Rs+(RdHi..RdLo)
SMULL<cond><S> RdLo, RdHi, Rm, Rs	RdHi..RdLo = Rm*Rs
STC<cond> <p_cp num>, CRd, #	Store coprocessor Reg with #
STM<cond><admm> Rm, {reg list}^	special, see doc
STM<cond><admm> Rm<!,>, {reg list}	[Rm+4] = for each in <reglist>
STM<cond><admm> Rm<!,>, {reg list}^	special, see doc
STR<cond> Rd, Rn, #	[Rn + #] = Rd
STR<cond>B Rd, Rn, #	[Rn + #] = Rd
STR<cond>BT Rd, Rn, #	[Rn + #] = Rd
STR<cond>H Rd, <address>	[address] = Rd
STR<cond>T Rd, Rn, #	[Rn + #] = Rd
SUB<cond><S> Rd, Rn, <sh_op>	Rd = Rn - <s_op>
SWI <swi_number>	call software interrupt
SWP<cond> Rd, Rm, [Rn]	Rd = [Rn]; [Rn] = Rm
SWP<cond>B Rd, Rm, [Rn]	Rd = [Rn]; [Rn] = Rm
TBQ<cond> Rn, <sh_op>	<flags> = Rn ^ <s_op>
TST<cond> Rn, <sh_op>	<flags> = Rn & <s_op>
UMLAL<cond><S> RdLo, RdHi, Rm, Rs	RdHi..RdLo = Rm*Rs+(RdHi..RdLo)
UMULL<cond><S> RdLo, RdHi, Rm, Rs	RdHi..RdLo = Rm*Rs

Data Processing Opcode
 Load/Store Opcode
 Branching Opcode
 Multiplication Opcode
 Other Opcodes
 Coprocessor Opcodes

Flag	Description
Z	Zero Flag
C	Carry Flag
N	Negative Flag
V	Overflow Flag

<http://www.agbdev.net/re-eject/> ARM Reference

Mnemonic	Description	Mnemonic	Description
ADC	Add with Carry	MSR	Move to Status Register
ADD	Add	MUL	Multiply
AND	Logical AND	MVN	Move Negative
B	Branch	ORR	Logical OR
BIC	Bit Clear	RSB	Reverse Subtract
BL	Branch and Link	RSC	Reverse Subtract with Carry
BX	Branch and Exchange	SBC	Subtract with Carry
CDP	Coprocessor Data Processing	SMMLAL	Signed Long Multiply Accumulate
CMN	Compare Negative	SMULL	Signed Long Multiply
CMP	Compare	STC	Store Coprocessor
EOR	Logical Exclusive OR (XOR)	STM	Store Multiple
LDC	Load Coprocessor	STR	Store Register
LDM	Load Multiple	STRB	Store Register Byte
LDR	Load Register	STRBT	Store Register Byte Translate
LDRB	Load Register Byte	STRH	Store Register Half Word
LDRBT	Load Register Byte Translate	STRT	Store Register Translate
LDRH	Load Register Half Word	SUB	Subtract
LDRSB	Load Register Signed Byte	SWI	Software Interrupt
LDRSH	Load Register Signed Half Word	SWP	Swap
LDRT	Load Register Translate	SWPB	Swap Byte
MCR	Move to Coprocessor from ARM reg	TEQ	Test Equivalence
MLA	Multiply Accumulate	TST	Test
MOV	Move	UMLAL	Unsigned Long Multiply Accumulate
MRC	Move to ARM reg from Coprocessor	UMULL	Unsigned Long Multiply
MRS	Move from Status Register		

Opcode	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC<cond><S> Rd, Rn, #	cond	0	0	1	0	1	0	1	0	1	0	S	Rn	Rd	rotate																	
ADC<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	1	0	1	0	1	0	S	Rn	Rd	shift #																	
ADC<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	1	0	1	0	1	S	Rn	Rd	Rs	0																
ADD<cond><S> Rd, Rn, #	cond	0	0	1	0	1	0	1	0	1	0	S	Rn	Rd	rotate																	
ADD<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	0	1	0	1	0	1	S	Rn	Rd	shift #																	
ADD<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	1	0	1	0	1	S	Rn	Rd	Rs	0																
AND<cond><S> Rd, Rn, #	cond	0	0	1	0	0	0	0	0	0	0	S	Rn	Rd	rotate																	
AND<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	0	0	0	0	0	0	S	Rn	Rd	shift #																	
AND<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	0	0	0	0	0	S	Rn	Rd	Rs	0																
B<cond> <target addr>	cond	1	0	1	0																											
BIC<cond><S> Rd, Rn, #	cond	0	0	1	1	1	1	0	1	0	1	S	Rn	Rd	rotate																	
BIC<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	1	1	1	0	1	0	1	S	Rn	Rd	shift #																	
BIC<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	1	1	1	0	1	0	1	S	Rn	Rd	Rs	0																
BL<cond> <target addr>	cond	1	0	1	1																											
BX<cond> Rm	cond	0	0	0	1	0	0	1	0	1	0	SBO	SBO	SBO																		
CDP<cond> <p<cp#>, , CRd, CRn, CRm, <o2>	cond	1	1	1	0							op1	CRn	CRd	cp_num	op2	0															
CMN<cond> Rn, #	cond	0	0	1	1	0	1	1	1	1	1	Rn	SBZ	rotate																		
CMN<cond> Rn, Rm OP #	cond	0	0	0	1	0	1	1	1	1	1	Rn	SBZ	shift #																		
CMN<cond> Rn, Rm OP Rs	cond	0	0	0	1	0	1	1	1	1	1	Rn	SBZ	Rs	0																	
CMP<cond> Rn, #	cond	0	0	1	1	0	1	0	1	0	1	Rn	SBZ	rotate																		
CMP<cond> Rn, Rm OP #	cond	0	0	0	1	0	1	0	1	0	1	Rn	SBZ	shift #																		
CMP<cond> Rn, Rm OP Rs	cond	0	0	0	1	0	1	0	1	0	1	Rn	SBZ	Rs	0																	
EOR<cond><S> Rd, Rn, #	cond	0	0	1	0	0	0	1	0	1	0	S	Rn	Rd	rotate																	
EOR<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	0	0	1	0	1	0	S	Rn	Rd	shift #																	
EOR<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	0	1	0	1	0	S	Rn	Rd	Rs	0																
LDC<cond> <p<cp_num>, CRd, #	cond	1	1	0								P	U	N	W	1	Rn	CRd	cp_num													
LDM<cond><admmode> Rm, reg list^	cond	1	0	0								P	U	1	0	1	Rn	0														
LDM<cond><admmode> Rm<!>, reg list	cond	1	0	0								P	U	0	1	1	Rn															
LDM<cond><admmode> Rm<!>, reg list^	cond	1	0	0								P	U	1	1	1	Rn	1														
LDR<cond> Rd, Rn, #	cond	0	1	0	1	0	0	0	1	1	1	Rn	Rd																			
LDR<cond> Rd, Rn, #	cond	0	1	1	0	1	0	0	1	1	1	Rn	Rd																			
LDR<cond>B Rd, Rn, #	cond	0	1	0	1	0	1	1	1	1	1	Rn	Rd																			
LDR<cond>B Rd, Rn, #	cond	0	1	1	0	1	1	1	1	1	1	Rn	Rd																			
LDR<cond>BT Rd, Rn, #	cond	0	1	0	1	0	1	1	1	1	1	Rn	Rd																			
LDR<cond>BT Rd, Rn, #	cond	0	1	1	0	1	0	1	1	1	1	Rn	Rd																			
LDR<cond>H Rd, <address>	cond	0	0	0	0	1	0	1	1	1	1	Rn	Rd	addr_mode	1	0	1	1	addr_mode													
LDR<cond>SB Rd, <address>	cond	0	0	0	0	1	0	1	1	1	1	Rn	Rd	addr_mode	1	1	0	1	addr_mode													
LDR<cond>SH Rd, <address>	cond	0	0	0	0	1	0	1	1	1	1	Rn	Rd	addr_mode	1	1	1	1	addr_mode													
LDR<cond>T Rd, Rn, #	cond	0	1	0	1	0	0	1	1	1	1	Rn	Rd																			
LDR<cond>T Rd, Rn, #	cond	0	1	1	0	1	0	0	1	1	1	Rn	Rd	shift #																		
MCR<cond> <p<cp#>, , Rd, CRn, CRm, <o2>	cond	1	1	1	1	0						op1	0	CRn	Rd	cp_num	op2	1	CRm													
MLA<cond><S> Rd, Rm, Rs, Rn	cond	0	0	0	0	0	0	0	1	0	1	S	Rd	Rn	Rs	1	0	0	1	Rm												
MOV<cond><S> Rd, #	cond	0	0	1	1	1	0	1	0	1	0	S	SBZ	Rd	rotate																	
MOV<cond><S> Rd, Rm OP #	cond	0	0	0	1	1	0	1	0	1	0	S	SBZ	Rd	shift #																	
MOV<cond><S> Rd, Rm OP Rs	cond	0	0	0	1	1	0	1	0	1	0	S	SBZ	Rd	Rs	0																
MRC<cond> <p<cp#>, , Rd, CRn, CRm, <o2>	cond	1	1	1	1	0						op1	1	CRn	Rd	cp_num	op2	2	1	CRm												
MRS<cond> Rd, CPSR	cond	0	0	0	1	0	0	0	0	0	0	SBO	Rd																			
MRS<cond> Rd, SPSR	cond	0	0	0	1	0	1	0	0	0	0	SBO	Rd																			
MSR<cond> CPSR <fields>, Rm	cond	0	0	0	1	0	0	1	0	1	0	field_mask	SBO																			
MSR<cond> CPSR f, #	cond	0	0	1	1	0	1	0	1	0	1	field_mask	SBO	rotate																		
MSR<cond> CPSR <fields>, Rm	cond	0	0	0	1	0	1	1	0	1	0	field_mask	SBO																			
MSR<cond> SPSCR f, #	cond	0	0	1	1	0	1	1	0	1	0	field_mask	SBO	rotate																		
MUL<cond><S> Rd, Rm, Rs	cond	0	0	0	0	0	0	0	0	1	0	S	Rd	SBZ	Rs	1	0	0	1	Rm												
MVN<cond><S> Rd, #	cond	0	0	1	1	1	1	1	1	1	1	S	SBZ	Rd	rotate																	
MVN<cond><S> Rd, Rm OP #	cond	0	0	0	1	1	1	1	1	1	1	S	SBZ	Rd	shift #																	
MVN<cond><S> Rd, Rm OP Rs	cond	0	0	0	1	1	1	1	1	1	1	S	SBZ	Rd	Rs	0																
ORR<cond><S> Rd, Rn, #	cond	0	0	1	1	1	0	1	0	1	0	S	Rn	Rd	rotate																	
ORR<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	1	1	0	1	0	1	0	S	Rn	Rd	shift #																	
ORR<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	1	1	0	1	0	1	0	S	Rn	Rd	Rs	0																
RSB<cond><S> Rd, Rn, #	cond	0	0	1	0	1	0	1	1	1	1	S	Rn	Rd	rotate																	
RSB<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	1	0	1	1	1	1	S	Rn	Rd	shift #																	
RSB<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	1	0	1	1	1	1	S	Rn	Rd	Rs	0																
RSC<cond><S> Rd, Rn, #	cond	0	0	1	0	1	1	1	1	1	1	S	Rn	Rd	rotate																	
RSC<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	1	1	1	1	1	1	S	Rn	Rd	shift #																	
RSC<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	1	1	1	1	1	1	S	Rn	Rd	Rs	0																
SBCC<cond><S> Rd, Rn, #	cond	0	0	1	1	0	1	0	1	0	1	S	Rn	Rd	rotate																	
SBCC<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	1	0	1																									