

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 & I<s_op>
BL<cond> <target_addr>	LR = PC+4; PC = PC + <offset>
BX<cond> Rm	PC = Rm Mode=THUMB
CMP<cond> p<cpl>, 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<cpl num>, CRd, #	load coprocessor register with #
LDM<cond><adm> Rm<!, <reg list>	special, see doc
LDM<cond><adm> Rm<!, <reg list>*	for each in <reglist> = [Rn+4]
LDR<cond> Rd, Rn, #	Rd = [Rn + #]
LDR<cond> B Rd, Rn, #	Rd = [Rn + #]
LDR<cond> BT Rd, Rn, #	Rd = [Rn + #]
LDR<cond> SB Rd, <address>	Rd = [address]
LDR<cond> SH Rd, <address>	Rd = [address]
LDR<cond> T Rd, Rn, #	Rd = [Rn + #]
MCR<cond> p<cpl#>, o1, Rd, CRn, CRm, o2	move from co-cpu reg to ARM reg
MLA<cond><S> Rd, Rm, Rs, Rn	Rd = Rm * Rs + Rn
MOV<cond><S> Rd, <sh_op>	Rd = <s_op>
MRC<cond> p<cpl#>, o1, 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_f = # (fields pick bytes to copy)
MSR<cond> CPSR <fields>, Rm	SPSR = Rm (fields pick bytes to copy)
MSR<cond> SPSR_f, #	SPSR_f = # (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
SMLAL<cond><S> RdLo, RdHi, Rm, Rs	RdHi..RdLo = Rm*(RdHi..RdLo)
SMLML<cond><S> RdLo, RdHi, Rm, Rs	RdHi..RdLo = Rm*Rs
STC<cond> p<cpl num>, CRd, #	Store coprocessor Reg with #
STM<cond><adm> Rm, <reg list>^	special, see doc
STM<cond><adm> Rm<!, <reg list>^	[Rm+4] = for each in <reglist>
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>	Rd = [address]
STR<cond> T Rd, Rn, #	Rd = [Rn + #]
SUB<cond><S> Rd, Rn, <sh_op>	Rd = Rn - <s_op>
SWI <swi_number>	call software interrupt
SWP<cond> Rd, Rm, [Rn]	Rd = [Rn]; Rm = Rm
SWP<cond> B Rd, Rm, [Rn]	Rd = [Rn]; Rm = Rm
TEQ<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

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	S	Rn	Rd	rotate		#																					
ADC<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	1	0	1	0	1	S	Rn	Rd	shift #	0	shift	0	Rm																		
ADC<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	1	0	1	0	S	Rn	Rd	rotate		#																				
ADD<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	1	0	0	0	S	Rn	Rd	shift #	0	shift	0	Rm																		
ADD<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	1	0	0	S	Rn	Rd	shift #	0	shift	1	Rm																		
AND<cond><S> Rd, Rn, #	cond	0	0	1	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	S	Rn	Rd	shift #	0	shift	0	Rm																		
AND<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	0	0	0	S	Rn	Rd	rotate		#																				
B<cond> <target_addr>	cond	1	0	1	0								24 bit offset																						
BIC<cond><S> Rd, Rn, #	cond	0	0	1	1	1	0	0	1	S	Rn	Rd	rotate		#																				
BIC<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	1	1	1	0	0	S	Rn	Rd	shift #	0	shift	0	Rm																		
BIC<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	1	1	1	0	S	Rn	Rd	shift #	0	shift	1	Rm																		
BL<cond> <target_addr>	cond	1	0	1	1								24 bit offset																						
BX<cond> Rm	cond	0	0	0	0	0	0	0	1	0	SBO	SBO	SBO	0	0	0	1	Rm																	
CDP<cond> p<cpl#>, , CRd, CRn, CRm, <o2>	cond	1	1	0	0	1	0	0	1	0	CRn	CRd	cp_num	op2	0	CRm																			
CMN<cond> Rn, #	cond	0	0	1	0	1	1	1	1	S	Rn	SBZ	rotate		#																				
CMN<cond> Rn, Rm OP #	cond	0	0	0	1	0	1	1	1	S	Rn	SBZ	shift #	0	shift	0	Rm																		
CMN<cond> Rn, Rm OP Rs	cond	0	0	0	0	1	0	1	1	S	Rn	SBZ	rotate		#																				
CMP<cond> Rd, Rn, #	cond	0	0	1	0	1	0	1	0	1	Rn	SBZ	rotate		#																				
CMP<cond> Rd, Rn, Rm OP #	cond	0	0	0	1	0	1	0	1	0	Rn	SBZ	shift #	0	shift	1	Rm																		
CMP<cond> Rd, Rn, Rm OP Rs	cond	0	0	0	0	1	0	1	0	1	Rn	SBZ	rotate		#																				
EOR<cond><S> Rd, Rn, <sh_op>	cond	0	0	1	0	0	0	1	0	S	Rn	Rd	rotate		#																				
EOR<cond><S> Rd, Rn, Rm OP #	cond	0	0	0	0	0	0	1	0	S	Rn	Rd	shift #	0	shift	0	Rm																		
EOR<cond><S> Rd, Rn, Rm OP Rs	cond	0	0	0	0	0	0	0	1	S	Rn	Rd	shift #	0	shift	1	Rm																		
LDC<cond> p<cpl num>, CRd, #	cond	1	1	0	0	P	U	N	W	0	Rn	CRd	cp_num	#																					
LDM<cond><adm> Rm, <reg list>^	cond	1	0	0	P	U	1	0	1	S	Rn	0	register list																						
LDM<cond><adm> Rm<!, <reg list>^	cond	1	0	0	P	U	0	1	1	S	Rn	1	register list																						
LDM<cond><adm> Rm<!, <reg list>^	cond	1	0	0	P	U	1	1	1	S	Rn	1	register list																						
LDR<cond> Rd, Rn, #	cond	0	1	0	P	U	0	1	0	1	Rn	Rd	shift #	0	shift	0	Rm																		
LDR<cond> B Rd, Rn, #	cond	0	1	0	P	U	1	0	1	0	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> BT Rd, Rn, #	cond	0	1	0	P	U	1	1	1	0	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> SB Rd, Rn, #	cond	0	1	0	P	U	1	1	0	0	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> SH Rd, Rn, #	cond	0	1	0	P	U	1	1	0	1	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> T Rd, Rn, #	cond	0	1	0	P	U	0	1	0	1	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> H Rd, <address>	cond	0	1	0	P	U	1	0	1	0	Rn	Rd	addr mode	1	0	1	1	addr mode																	
LDR<cond> BT Rd, Rn, #	cond	0	1	0	P	U	0	1	0	1	Rn	Rd	shift #	0	shift	0	Rm																		
LDR<cond> SB Rd, Rn, #	cond	0	1	0	P	U	1	0	0	1	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> SH Rd, Rn, #	cond	0	1	0	P	U	1	0	0	0	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> T Rd, Rn, #	cond	0	1	0	P	U	0	1	0	0	Rn	Rd	shift #	0	shift	1	Rm																		
LDR<cond> H Rd, <address>	cond	0	1	0	P	U	1	0	0	0	Rn	Rd	register list																						
LDR<cond> BT Rd, Rn, #	cond	0	1	0	P	U	0	1	0	0	Rn	Rd	register list																						
LDR<cond> SB Rd, Rn, #	cond	0	1	0	P	U	1																												