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
AND<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	0	0	S	Rn				Rd				shift #				shift		0	Rm			
AND<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	0	0	S	Rn				Rd				Rs			0	shift		1	Rm			
MUL<cond><S> Rd, Rm, Rs	cond				0	0	0	0	0	0	0	S	Rd				SBZ				Rs			1	0	0	1	Rm			
STR<cond>H Rd, <address>	cond				0	0	0	P	U	I	W	0	Rn				Rd				addr_mode			1	0	1	1	addr_mode			
LDR<cond>H Rd, <address>	cond				0	0	0	P	U	I	W	1	Rn				Rd				addr_mode			1	0	1	1	addr_mode			
Undefined Instruction	cond				0	0	0	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0	1	x	x	x	x
LDR<cond>SB Rd, <address>	cond				0	0	0	P	U	I	W	1	Rn				Rd				addr_mode			1	1	0	1	addr_mode			
Undefined Instruction	cond				0	0	0	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1	x	x	x	x
LDR<cond>SH Rd, <address>	cond				0	0	0	P	U	I	W	1	Rn				Rd				addr_mode			1	1	1	1	addr_mode			
EOR<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	0	1	S	Rn				Rd				shift #				shift		0	Rm			
EOR<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	0	1	S	Rn				Rd				Rs			0	shift		1	Rm			
MLA<cond><S> Rd, Rm, Rs, Rn	cond				0	0	0	0	0	0	1	S	Rd				Rn				Rs			1	0	0	1	Rm			
SUB<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	1	0	S	Rn				Rd				shift #				shift		0	Rm			
SUB<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	1	0	S	Rn				Rd				Rs			0	shift		1	Rm			
RSB<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	1	1	S	Rn				Rd				shift #				shift		0	Rm			
RSB<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	1	1	S	Rn				Rd				Rs			0	shift		1	Rm			
ADD<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	0	0	S	Rn				Rd				shift #				shift		0	Rm			
ADD<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	0	0	S	Rn				Rd				Rs			0	shift		1	Rm			
UMULL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	0	0	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
ADC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	0	1	S	Rn				Rd				shift #				shift		0	Rm			
ADC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	0	1	S	Rn				Rd				Rs			0	shift		1	Rm			
UMLAL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	0	1	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
SBC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	1	0	S	Rn				Rd				shift #				shift		0	Rm			
SBC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	1	0	S	Rn				Rd				Rs			0	shift		1	Rm			
SMULL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	1	0	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
RSC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	1	1	S	Rn				Rd				shift #				shift		0	Rm			
RSC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	1	1	S	Rn				Rd				Rs			0	shift		1	Rm			
SMLAL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	1	1	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
TST<cond> Rn, Rm OP #	cond				0	0	0	1	0	0	0	1	Rn				SBZ				shift #				shift		0	Rm			
TST<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	0	0	1	Rn				SBZ				Rs			0	shift		1	Rm			
MRS<cond> Rd, CPSR	cond				0	0	0	1	0	0	0	0	SBO				Rd				SBZ										
SWP<cond> Rd, Rm, [Rn]	cond				0	0	0	1	0	0	0	SBZ	Rn				Rd				SBZ			1	0	0	1	Rm			
TEQ<cond> Rn, Rm OP #	cond				0	0	0	1	0	0	1	1	Rn				SBZ				shift #				shift		0	Rm			
TEQ<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	0	1	1	Rn				SBZ				Rs			0	shift		1	Rm			
MSR<cond> CPSR_<fields>, Rm	cond				0	0	0	1	0	0	1	0	field_mask				SBO				SBZ						0	Rm			
BX<cond> Rm	cond				0	0	0	1	0	0	1	0	SBO				SBO				SBO			0	0	0	1	Rm			
CMP<cond> Rn, Rm OP #	cond				0	0	0	1	0	1	0	1	Rn				SBZ				shift #				shift		0	Rm			
CMP<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	1	0	1	Rn				SBZ				Rs			0	shift		1	Rm			
MRS<cond> Rd, SPSR	cond				0	0	0	1	0	1	0	0	SBO				Rd				SBZ										
SWP<cond>B Rd, Rm, [Rn]	cond				0	0	0	1	0	1	SBZ		Rn				Rd				SBZ			1	0	0	1	Rm			
CMN<cond> Rn, Rm OP #	cond				0	0	0	1	0	1	1	1	Rn				SBZ				shift #				shift		0	Rm			
CMN<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	1	1	1	Rn				SBZ				Rs			0	shift		1	Rm			
MSR<cond> SPSR_<fields>, Rm	cond				0	0	0	1	0	1	1	0	field_mask				SBO				SBZ						0	Rm			
ORR<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	1	1	0	0	S	Rn				Rd				shift #				shift		0	Rm			
ORR<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	1	1	0	0	S	Rn				Rd				Rs			0	shift		1	Rm			
MOV<cond><S> Rd, Rm OP #	cond				0	0	0	1	1	0	1	S	SBZ				Rd				shift #				shift		0	Rm			
MOV<cond><S> Rd, Rm OP Rs	cond				0	0	0	1	1	0	1	S	SBZ				Rd				Rs			0	shift		1	Rm			
BIC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	1	1	1	0	S	Rn				Rd				shift #				shift		0	Rm			
BIC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	1	1	1	0	S	Rn				Rd				Rs			0	shift		1	Rm			
MVN<cond><S> Rd, Rm OP #	cond				0	0	0	1	1	1	1	S	SBZ				Rd				shift #				shift		0	Rm			
MVN<cond><S> Rd, Rm OP Rs	cond				0	0	0	1	1	1	1	S	SBZ				Rd				Rs			0	shift		1	Rm			
AND<cond><S> Rd, Rn, #	cond				0	0	1	0	0	0	0	S	Rn				Rd				rotate				#						
EOR<cond><S> Rd, Rn, #	cond				0	0	1	0	0	0	1	S	Rn				Rd				rotate				#						
SUB<cond><S> Rd, Rn, #	cond				0	0	1	0	0	1	0	S	Rn				Rd				rotate				#						
RSB<cond><S> Rd, Rn, #	cond				0	0	1	0	0	1	1	S	Rn				Rd				rotate				#						
ADD<cond><S> Rd, Rn, #	cond				0	0	1	0	1	0	0	S	Rn				Rd				rotate				#						
ADC<cond><S> Rd, Rn, #	cond				0	0	1	0	1	0	1	S	Rn				Rd				rotate				#						
SBC<cond><S> Rd, Rn, #	cond				0	0	1	0	1	1	0	S	Rn				Rd				rotate				#						
RSC<cond><S> Rd, Rn, #	cond				0	0	1	0	1	1	1	S	Rn				Rd				rotate				#						
TST<cond> Rn, #	cond				0	0	1	1	0	0	0	1	Rn				SBZ				rotate				#						
TEQ<cond> Rn, #	cond				0	0	1	1	0	0	1	1	Rn				SBZ				rotate				#						
MSR<cond> CPSR_f, #	cond				0	0	1	1	0	0	1	0	field_mask				SBO				rotate				#						
CMP<cond> Rn, #	cond				0	0	1	1	0	1	0	1	Rn				SBZ				rotate				#						
CMN<cond> Rn, #	cond				0	0	1	1	0	1	1	1	Rn				SBZ				rotate				#						
MSR<cond> SPSR_f, #	cond				0	0	1	1	0	1	1	0	field_mask				SBO				rotate				#						
ORR<cond><S> Rd, Rn, #	cond				0	0	1	1	1	0	0	S	Rn				Rd				rotate				#						
MOV<cond><S> Rd, #	cond				0	0	1	1	1	0	1	S	SBZ				Rd				rotate				#						
BIC<cond><S> Rd, Rn, #	cond				0	0	1	1	1	1	0	S	Rn				Rd				rotate				#						
MVN<cond><S> Rd, #	cond				0	0	1	1	1	1	1	S	SBZ				Rd				rotate				#						
STR<cond> Rd, Rn, #	cond				0	1	0	P	U	0	W	0	Rn				Rd				#										
LDR<cond> Rd, Rn, #	cond				0	1	0	P	U	0	W	1	Rn				Rd				#										
STR<cond>B Rd, Rn, #	cond				0	1	0	P	U	1	W	0	Rn				Rd				#										
LDR<cond>B Rd, Rn, #	cond				0	1	0	P	U	1	W	1	Rn				Rd				#										
STR<cond>T Rd, Rn, #	cond				0	1	0	0	U	0	1	0	Rn				Rd				#										
LDR<cond>T Rd, Rn, #	cond				0	1	0	0	U	0	1	1	Rn				Rd				#										
STR<cond>BT Rd, Rn, #	cond				0	1	0	0	U	1	1	0	Rn				Rd				#										
LDR<cond>BT Rd, Rn, #	cond				0	1	0	0	U	1	1	1	Rn				Rd				#										
STR<cond> Rd, Rn, #	cond				0	1	1	P	U	0	W	0	Rn				Rd				shift #				shift		0	Rm			
LDR<cond> Rd, Rn, #	cond				0	1	1	P	U	0	W	1	Rn				Rd				shift #				shift		0	Rm			
STR<cond>B Rd, Rn, #	cond				0	1	1	P	U	1	W	0	Rn				Rd				shift #				shift		0	Rm			
LDR<cond>B Rd, Rn, #	cond				0	1	1	P	U	1	W	1	Rn				Rd				shift #				shift		0	Rm			
STR<cond>T Rd, Rn, #	cond				0	1	1	0	U	0	1	0	Rn				Rd				shift #				shift		0	Rm			
LDR<cond>T Rd, Rn, #	cond				0	1	1	0	U	0	1	1	Rn				Rd				shift #				shift		0	Rm			
STR<cond>BT Rd, Rn, #	cond				0	1	1	0	U	1	1	0	Rn				Rd				shift #				shift		0	Rm			
LDR<cond>BT Rd, Rn, #	cond				0	1	1	0	U	1	1	1	Rn				Rd				shift #				shift		0	Rm			
Undefined Instruction	cond				0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x			
STM<cond><addrmode> Rm<!>, reg list	cond				1	0	0	P	U	0	W	0	Rn				register list														
LDM<cond><addrmode> Rm<!>, reg list	cond				1	0	0	P	U	0	W	1	Rn				register list														
STM<cond><addrmode> Rm, reg list^	cond				1	0	0	P	U	1	0	0	Rn				0	register list													
UNPREDICTABLE	cond				1	0	0	x	x	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
LDM<cond><addrmode> Rm, reg list^	cond				1	0	0	P	U	1	0	1	Rn				0	register list													
UNPREDICTABLE	cond				1	0	0	x	x	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
STM<cond><addrmode> Rm<!>, reg list^	cond				1	0	0	P	U	1	W	0	Rn				1	register list													
LDM<cond><addrmode> Rm<!>, reg list^	cond				1	0	0	P	U	1	W	1	Rn				1	register list													
B<cond> <target addr>	cond				1	0	1	0	24_bit_offset																							
BL<cond> <target addr>	cond				1	0	1	1	24_bit_offset																							
STC<cond> p<cp_num>, CRd, #	cond				1	1	0	P	U	N	W	0	Rn				CRd				cp_num				#						
LDC<cond> p<cp_num>, CRd, #	cond				1	1	0	P	U	N	W	1	Rn				CRd				cp_num				#						
CDP<cond> p<cp#>,<o1>,CRd,CRn,CRm,<o2>	cond				1	1	1	0	op1				CRn				CRd				cp_num			op2			0	CRm		
MCR<cond> p<cp#>,<o1>,Rd,CRn,CRm,<o2>	cond				1	1	1	0	op1			0	CRn				Rd				cp_num			op2			1	CRm		
MRC<cond> p<cp#>,<o1>,Rd,CRn,CRm,<o2>	cond				1	1	1	0	op1			1	CRn				Rd				cp_num			op2			1	CRm		
SWI <swi_number>	cond				1	1	1	1	swi_number																							

Meaning	Mnemonic	Opcode	Status Flags
Equal	EQ	0 0 0 0	Z = 1
Not Equal	NE	0 0 0 1	Z = 0
Carry Set	CS	0 0 1 0	C = 1
Carry Clear	CC	0 0 1 1	C = 0
Unsigned Higher or Same	HS	0 0 1 0	C = 1
Unsigned Lower	LO	0 0 1 1	C = 0
Minus/Negative	MI	0 1 0 0	N = 1
Plus/Positive or Zero	PL	0 1 0 1	N = 0
Overflow	VS	0 1 1 0	V = 1
No Overflow	VC	0 1 1 1	V = 0
Unsigned Higher	HI	1 0 0 0	C = 1, Z = 0
Unsigned Lower or Same	LS	1 0 0 1	C = 0, Z = 1
Signed Greater than or Equal	GE	1 0 1 0	N = V
Signed Less than	LT	1 0 1 1	N != V
Signed Greater than	GT	1 1 0 0	Z = 0, N = V
Signed Less than or Equal	LE	1 1 0 1	Z = 1, N != V
Always	AL	1 1 1 0	-
Never	NE	1 1 1 1	-

rE Ejected

http://www.agbdev.net/re-eject/

ARM Reference

Opcode	Notes	Z	C	N	V
ADC<cond><S> Rd, Rn, <sh_op>	-	s	s	s	s
ADD<cond><S> Rd, Rn, <sh_op>	-	s	s	s	s
AND<cond><S> Rd, Rn, <sh_op>	-	s	s	s	
B<cond> <target_addr>	-				
BIC<cond><S> Rd, Rn, <sh_op>	-	s	s	s	
BL<cond> <target_addr>	-				
BX<cond> Rm	-				
CDP<cond> p<cp#>,o1,CRd,CRn,CRm,o2	-				
CMN<cond> Rn, <sh_op>	-	x	x	x	x
CMP<cond> Rn, <sh_op>	-	x	x	x	x
EOR<cond><S> Rd, Rn, <sh_op>	-	s	s	s	
LDC<cond> p<cp_num>, CRd, #	-				
LDM<cond><adm> Rm, {reg list}^	-				
LDM<cond><adm> Rm<!>, {reg list}	"!" Keeps last Rm Value				
LDM<cond><adm> Rm<!>, {reg list}^	-				
LDR<cond> Rd, Rn, #	word				
LDR<cond>B Rd, Rn, #	byte				
LDR<cond>BT Rd, Rn, #	translate byte				
LDR<cond>H Rd, <address>	half word				
LDR<cond>SB Rd, <address>	signed byte				
LDR<cond>SH Rd, <address>	signed half word				
LDR<cond>T Rd, Rn, #	translate word				
MCR<cond> p<cp#>,o1,Rd,CRn,CRm,o2	-				
MLA<cond><S> Rd, Rm, Rs, Rn	-	s	s	s	
MOV<cond><S> Rd, <sh_op>	-	s	s	s	
MRC<cond> p<cp#>,o1,Rd,CRn,CRm,o2	if Rd = r15 then flags affected	*	*	*	*
MRS<cond> Rd, CPSR	-				
MRS<cond> Rd, SPSR	-				
MSR<cond> CPSR_<fields>, Rm	-				
MSR<cond> CPSR_f, #	-				
MSR<cond> SPSR_<fields>, Rm	-				
MSR<cond> SPSR_f, #	-				
MUL<cond><S> Rd, Rm, Rs	-	s	s	s	
MVN<cond><S> Rd, <sh_op>	-	s	s	s	
ORR<cond><S> Rd, Rn, <sh_op>	-	s	s	s	
RSB<cond><S> Rd, Rn, <sh_op>	-	s	s	s	s
RSC<cond><S> Rd, Rn, <sh_op>	-	s	s	s	s
SBC<cond><S> Rd, Rn, <sh_op>	-	s	s	s	s
SMLAL<cond><S> RdLo, RdHi, Rm, Rs	64bit target	s	s	s	s
SMULL<cond><S> RdLo, RdHi, Rm, Rs	64bit target	s	s	s	s
STC<cond> p<cp_num>, CRd, #	-				
STM<cond><adm> Rm, {reg list}^	-				
STM<cond><adm> Rm<!>, {reg list}	"!" Keeps last Rm Value				
STM<cond><adm> Rm<!>, {reg list}^	-				
STR<cond> Rd, Rn, #	word				
STR<cond>B Rd, Rn, #	byte				
STR<cond>BT Rd, Rn, #	translate byte				
STR<cond>H Rd, <address>	half word				
STR<cond>T Rd, Rn, #	translate word				
SUB<cond><S> Rd, Rn, <sh_op>	-	s	s	s	s
SWI <swi_number>	-				
SWP<cond> Rd, Rm, [Rn]	-				
SWP<cond>B Rd, Rm, [Rn]	-				
TEQ<cond> Rn, <sh_op>	-	x	x	x	
TST<cond> Rn, <sh_op>	-	x	x	x	
UMLAL<cond><S> RdLo, RdHi, Rm, Rs	64bit target	s	s	s	s
UMULL<cond><S> RdLo, RdHi, Rm, Rs	64bit target	s	s	s	s

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

Flag Settings
s - if flag set
x - always
* - special

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>
BX<cond> Rm	PC = Rm; Mode=THUMB
CDP<cond> p<cp#>,o1,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><adm> Rm, {reg list}^	*special, see doc*
LDM<cond><adm> Rm<!>, {reg list}	for each in <reglist> = [Rn+=4]
LDM<cond><adm> 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#>,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<cp#>,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 = # (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
SMLAL<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><adm> Rm, {reg list}^	*special, see doc*
STM<cond><adm> Rm<!>, {reg list}	[Rm+=4] = for each in <reglist>
STM<cond><adm> 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
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	**Z**ero Flag
C	**C**arry Flag
N	**N**egative 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	0	1	0	1	S	Rn				Rd				shift #				shift			0	Rm			
ADC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	0	1	S	Rn				Rd				Rs			0	shift			1	Rm			
ADD<cond><S> Rd, Rn, #	cond				0	0	1	0	1	0	0	S	Rn				Rd				rotate				#							
ADD<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	0	0	S	Rn				Rd				shift #				shift			0	Rm			
ADD<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	0	0	S	Rn				Rd				Rs			0	shift			1	Rm			
AND<cond><S> Rd, Rn, #	cond				0	0	1	0	0	0	0	S	Rn				Rd				rotate				#							
AND<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	0	0	S	Rn				Rd				shift #				shift			0	Rm			
AND<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	0	0	S	Rn				Rd				Rs			0	shift			1	Rm			
B<cond> <target addr>	cond				1	0	1	0	24_bit_offset																							
BIC<cond><S> Rd, Rn, #	cond				0	0	1	1	1	1	0	S	Rn				Rd				rotate				#							
BIC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	1	1	1	0	S	Rn				Rd				shift #				shift			0	Rm			
BIC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	1	1	1	0	S	Rn				Rd				Rs			0	shift			1	Rm			
BL<cond> <target addr>	cond				1	0	1	1	24_bit_offset																							
BX<cond> Rm	cond				0	0	0	1	0	0	1	0	SBO				SBO				SBO				0	0	0	1	Rm			
CDP<cond> p<cp#>,<o1>,CRd,CRn,CRm,<o2>	cond				1	1	1	0	op1				CRn				CRd				cp_num				op2			0	CRm			
CMN<cond> Rn, #	cond				0	0	1	1	0	1	1	1	Rn				SBZ				rotate				#							
CMN<cond> Rn, Rm OP #	cond				0	0	0	1	0	1	1	1	Rn				SBZ				shift #				shift			0	Rm			
CMN<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	1	1	1	Rn				SBZ				Rs			0	shift			1	Rm			
CMP<cond> Rn, #	cond				0	0	1	1	0	1	0	1	Rn				SBZ				rotate				#							
CMP<cond> Rn, Rm OP #	cond				0	0	0	1	0	1	0	1	Rn				SBZ				shift #				shift			0	Rm			
CMP<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	1	0	1	Rn				SBZ				Rs			0	shift			1	Rm			
EOR<cond><S> Rd, Rn, #	cond				0	0	1	0	0	0	1	S	Rn				Rd				rotate				#							
EOR<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	0	1	S	Rn				Rd				shift #				shift			0	Rm			
EOR<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	0	1	S	Rn				Rd				Rs			0	shift			1	Rm			
LDC<cond> p<cp_num>, CRd, #	cond				1	1	0	P	U	N	W	1	Rn				CRd				cp_num				#							
LDM<cond><addrmode> Rm, reg list^	cond				1	0	0	P	U	1	0	1	Rn				0	register list														
LDM<cond><addrmode> Rm<!>, reg list	cond				1	0	0	P	U	0	W	1	Rn				register list															
LDM<cond><addrmode> Rm<!>, reg list^	cond				1	0	0	P	U	1	W	1	Rn				1	register list														
LDR<cond> Rd, Rn, #	cond				0	1	0	P	U	0	W	1	Rn				Rd				#											
LDR<cond> Rd, Rn, #	cond				0	1	1	P	U	0	W	1	Rn				Rd				shift #				shift			0	Rm			
LDR<cond>B Rd, Rn, #	cond				0	1	0	P	U	1	W	1	Rn				Rd				#											
LDR<cond>B Rd, Rn, #	cond				0	1	1	P	U	1	W	1	Rn				Rd				shift #				shift			0	Rm			
LDR<cond>BT Rd, Rn, #	cond				0	1	0	0	U	1	1	1	Rn				Rd				#											
LDR<cond>BT Rd, Rn, #	cond				0	1	1	0	U	1	1	1	Rn				Rd				shift #				shift			0	Rm			
LDR<cond>H Rd, <address>	cond				0	0	0	P	U	I	W	1	Rn				Rd				addr_mode			1	0	1	1	addr_mode			
LDR<cond>SB Rd, <address>	cond				0	0	0	P	U	I	W	1	Rn				Rd				addr_mode			1	1	0	1	addr_mode			
LDR<cond>SH Rd, <address>	cond				0	0	0	P	U	I	W	1	Rn				Rd				addr_mode			1	1	1	1	addr_mode			
LDR<cond>T Rd, Rn, #	cond				0	1	0	0	U	0	1	1	Rn				Rd				#											
LDR<cond>T Rd, Rn, #	cond				0	1	1	0	U	0	1	1	Rn				Rd				shift #				shift			0	Rm			
MCR<cond> p<cp#>,<o1>,Rd,CRn,CRm,<o2>	cond				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	1	S	Rd				Rn				Rs			1	0	0	1	Rm			
MOV<cond><S> Rd, #	cond				0	0	1	1	1	0	1	S	SBZ				Rd				rotate				#							
MOV<cond><S> Rd, Rm OP #	cond				0	0	0	1	1	0	1	S	SBZ				Rd				shift #				shift			0	Rm			
MOV<cond><S> Rd, Rm OP Rs	cond				0	0	0	1	1	0	1	S	SBZ				Rd				Rs			0	shift			1	Rm			
MRC<cond> p<cp#>,<o1>,Rd,CRn,CRm,<o2>	cond				1	1	1	0	op1			1	CRn				Rd				cp_num				op2			1	CRm			
MRS<cond> Rd, CPSR	cond				0	0	0	1	0	0	0	0	SBO				Rd				SBZ											
MRS<cond> Rd, SPSR	cond				0	0	0	1	0	1	0	0	SBO				Rd				SBZ											
MSR<cond> CPSR_<fields>, Rm	cond				0	0	0	1	0	0	1	0	field_mask				SBO				SBZ							0	Rm			
MSR<cond> CPSR_f, #	cond				0	0	1	1	0	0	1	0	field_mask				SBO				rotate				#							
MSR<cond> SPSR_<fields>, Rm	cond				0	0	0	1	0	1	1	0	field_mask				SBO				SBZ							0	Rm			
MSR<cond> SPSR_f, #	cond				0	0	1	1	0	1	1	0	field_mask				SBO				rotate				#							
MUL<cond><S> Rd, Rm, Rs	cond				0	0	0	0	0	0	0	S	Rd				SBZ				Rs			1	0	0	1	Rm			
MVN<cond><S> Rd, #	cond				0	0	1	1	1	1	1	S	SBZ				Rd				rotate				#							
MVN<cond><S> Rd, Rm OP #	cond				0	0	0	1	1	1	1	S	SBZ				Rd				shift #				shift			0	Rm			
MVN<cond><S> Rd, Rm OP Rs	cond				0	0	0	1	1	1	1	S	SBZ				Rd				Rs			0	shift			1	Rm			
ORR<cond><S> Rd, Rn, #	cond				0	0	1	1	1	0	0	S	Rn				Rd				rotate				#							
ORR<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	1	1	0	0	S	Rn				Rd				shift #				shift			0	Rm			
ORR<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	1	1	0	0	S	Rn				Rd				Rs			0	shift			1	Rm			
RSB<cond><S> Rd, Rn, #	cond				0	0	1	0	0	1	1	S	Rn				Rd				rotate				#							
RSB<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	1	1	S	Rn				Rd				shift #				shift			0	Rm			
RSB<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	1	1	S	Rn				Rd				Rs			0	shift			1	Rm			
RSC<cond><S> Rd, Rn, #	cond				0	0	1	0	1	1	1	S	Rn				Rd				rotate				#							
RSC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	1	1	S	Rn				Rd				shift #				shift			0	Rm			
RSC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	1	1	S	Rn				Rd				Rs			0	shift			1	Rm			
SBC<cond><S> Rd, Rn, #	cond				0	0	1	0	1	1	0	S	Rn				Rd				rotate				#							
SBC<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	1	1	0	S	Rn				Rd				shift #				shift			0	Rm			
SBC<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	1	1	0	S	Rn				Rd				Rs			0	shift			1	Rm			
SMLAL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	1	1	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
SMULL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	1	0	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
STC<cond> p<cp_num>, CRd, #	cond				1	1	0	P	U	N	W	0	Rn				CRd				cp_num				#							
STM<cond><addrmode> Rm, reg list^	cond				1	0	0	P	U	1	0	0	Rn				0	register list														
STM<cond><addrmode> Rm<!>, reg list	cond				1	0	0	P	U	0	W	0	Rn				register list															
STM<cond><addrmode> Rm<!>, reg list^	cond				1	0	0	P	U	1	W	0	Rn				1	register list														
STR<cond> Rd, Rn, #	cond				0	1	0	P	U	0	W	0	Rn				Rd				#											
STR<cond> Rd, Rn, #	cond				0	1	1	P	U	0	W	0	Rn				Rd				shift #				shift			0	Rm			
STR<cond>B Rd, Rn, #	cond				0	1	0	P	U	1	W	0	Rn				Rd				#											
STR<cond>B Rd, Rn, #	cond				0	1	1	P	U	1	W	0	Rn				Rd				shift #				shift			0	Rm			
STR<cond>BT Rd, Rn, #	cond				0	1	0	0	U	1	1	0	Rn				Rd				#											
STR<cond>BT Rd, Rn, #	cond				0	1	1	0	U	1	1	0	Rn				Rd				shift #				shift			0	Rm			
STR<cond>H Rd, <address>	cond				0	0	0	P	U	I	W	0	Rn				Rd				addr_mode			1	0	1	1	addr_mode			
STR<cond>T Rd, Rn, #	cond				0	1	0	0	U	0	1	0	Rn				Rd				#											
STR<cond>T Rd, Rn, #	cond				0	1	1	0	U	0	1	0	Rn				Rd				shift #				shift			0	Rm			
SUB<cond><S> Rd, Rn, #	cond				0	0	1	0	0	1	0	S	Rn				Rd				rotate				#							
SUB<cond><S> Rd, Rn, Rm OP #	cond				0	0	0	0	0	1	0	S	Rn				Rd				shift #				shift			0	Rm			
SUB<cond><S> Rd, Rn, Rm OP Rs	cond				0	0	0	0	0	1	0	S	Rn				Rd				Rs			0	shift			1	Rm			
SWI <swi_number>	cond				1	1	1	1	swi_number																							
SWP<cond> Rd, Rm, [Rn]	cond				0	0	0	1	0	0	SBZ		Rn				Rd				SBZ			1	0	0	1	Rm			
SWP<cond>B Rd, Rm, [Rn]	cond				0	0	0	1	0	1	SBZ		Rn				Rd				SBZ			1	0	0	1	Rm			
TEQ<cond> Rn, #	cond				0	0	1	1	0	0	1	1	Rn				SBZ				rotate				#							
TEQ<cond> Rn, Rm OP #	cond				0	0	0	1	0	0	1	1	Rn				SBZ				shift #				shift			0	Rm			
TEQ<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	0	1	1	Rn				SBZ				Rs			0	shift			1	Rm			
TST<cond> Rn, #	cond				0	0	1	1	0	0	0	1	Rn				SBZ				rotate				#							
TST<cond> Rn, Rm OP #	cond				0	0	0	1	0	0	0	1	Rn				SBZ				shift #				shift			0	Rm			
TST<cond> Rn, Rm OP Rs	cond				0	0	0	1	0	0	0	1	Rn				SBZ				Rs			0	shift			1	Rm			
UMLAL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	0	1	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
UMULL<cond><S> RdLo, RdHi, Rm, Rs	cond				0	0	0	0	1	0	0	S	RdHi				RdLo				Rs			1	0	0	1	Rm			
Undefined Instruction	cond				0	0	0	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0	1	x	x	x	x
Undefined Instruction	cond				0	0	0	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1	x	x	x	x
Undefined Instruction	cond				0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x	x
UNPREDICTABLE	cond				1	0	0	x	x	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
UNPREDICTABLE	cond				1	0	0	x	x	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Mnemonic	Description
ADC	Add with Carry
ADD	Add
AND	Logical AND
B	Branch
BIC	Bit Clear
BL	Branch and Link
BX	Branch and Exchange
CDP	Coprocessor Data Processing
CMN	Compare Negative
CMP	Compare
EOR	Logical Exclusive OR (XOR)
LDC	Load Coprocessor
LDM	Load Multiple
LDR	Load Register
LDRB	Load Register Byte
LDRBT	Load Register Byte Translate
LDRH	Load Register Half Word
LDRSB	Load Register Signed Byte
LDRSH	Load Register Signed Half Word
LDRT	Load Register Translate
MCR	Move to Coprocessor from ARM reg
MLA	Multiply Accumulate
MOV	Move
MRC	Move to ARM reg from Coprocessor
MRS	Move from Status Register

Mnemonic	Description
MSR	Move to Status Register
MUL	Multiply
MVN	Move Negative
ORR	Logical OR
RSB	Reverse Subtract
RSC	Reverse Subtract with Carry
SBC	Subtract with Carry
SMLAL	Signed Long Multiply Accumulate
SMULL	Signed Long Multiply
STC	Store Coprocessor
STM	Store Multiple
STR	Store Register
STRB	Store Register Byte
STRBT	Store Register Byte Translate
STRH	Store Register Half Word
STRT	Store Register Translate
SUB	Subtract
SWI	Software Interrupt
SWP	Swap
SWPB	Swap Byte
TEQ	Test Equivalence
TST	Test
UMLAL	Unsigned Long Multiply Accumulate
UMULL	Unsigned Long Multiply