Arm 1-4

Command	Detail	Example	OP	Cycles	Opcode	Arch
ADCcc**S** R0, R1, *OP2*	Add with Carry	ADC R0,R1,R2	R0 = R1+R2+C	1	0101	
ADDcc**S** R0, R1, *OP2*	Add	ADD R0,R1,R2	R0 = R1+R2	1	0100	
ANDcc**S** R0, R1, *OP2*	Bitwise AND	AND R0,R1,R2	R0 = R1 and R2	1	0000	
Bcc addr	Branch (JP)	B label	R15=addr	3		
BICcc**S** R0, R1, *OP2*	Bit Clear		R0 = R1 and (CPL R2)	1	1110	
BLcc addr	Branch and Link (CALL)	BL label	R14=R15… R15=addr	3		
CDPcc #,e,Crd,Crn,Crm,e2 **CDO2**	Return From Exception					2,5
CMNcc**P** R1, *OP2*	Compare Negative (Set flags like ADD)		flags=R1+R2	1	1001	
CMPcc**P** R1, *OP2*	Compare (Set flags like SUB)		flags=R1-R2	1	1010	
EORcc**S** R0, R1, *OP2*	Exclusive OR (XOR)		R0 = R1 xor R2	1	0001	
LDCcc**L**TN #,Crd,addr,L **LDC2**	Load Coprocessor					2,5
LDMccmm R0**!**,{R1,R2...R3}	Load Multiple (POP)		Move R1,R2...R3→(R0)	1+		
LDRcc**BT** R0,addr,*shft*	LoaD Register (B=8 bit / T=access in user mode)		R0=(addr)	3+	psuedo	
LDRcc**H** R0,addr,*shft*	LoaD Register (16 bit)		R0=(addr)	3+		4T+
LDRcc**SB** R0,addr,*shft*	LoaD Register (8 bit signed)		R0=(addr)	3+		4
LDRcc**SH** R0,addr,*shft*	LoaD Register (16 bit signed)		R0=(addr)	3+		4
MCRcc #,e,Rd,Crn,Crm,e2 **MCR2**,MCRR	Move from registers to coprocessor					2,5,5Ed
MLAcc**S** R0,R1,R2,R3	Multiply with Accumulate		R0=(R1*R2)+R3	16		2
MOVcc**S** R0, *R2 ,shft*	Move		R0 = R2	1	1101	
MRCcc **#,e**,Rd,Crn,Crm,e2 **,MRC2**	Coprocessor Register transfer					2,5
MRScc R0,flags	Move from CPSR/SPSR to register… MSR R0,CPSR	MRS R4, CPSR	=PSR			3
MSRcc fields,#n/R0	Move from register to CPSR	MSR CPSR, R4	PSR=Rm			3
MULcc**S** R0, R1, R2	Multiply		R0=R1*R2	16		2
MVNcc**S** R0, R2 ,*shft*	Move Not (Flip bits of R2)		R0 = -R2	1	1111	
ORRcc**S** R0, R1, R2 ,*shft*	Inclusive Or		R0 = R1 or R2	1	1100	
RSBcc**S** R0, R1, R2 ,*shft*	Reverse SuBtract		R0 = R2-R1	1	0011	
RSCcc**S** R0, R1, R2 ,*shft*	Reverse Subtract with Carry		R0 = R2-R1+C-1	1	0111	
SBCcc**S** R0, R1, R2 ,*shft*	Subtract with carry		R0 = R1-R2+C-1	1	0110	
STCcc**L**TN #,Crd,addr,L **STC2**	Store to Coprosessor					2,5ExP
STMccmm R0,{R1,R2...R3}**!**	Store Multiple (PUSH)		Restore (R0)-> R1,R2...R3	2+		
STRcc**BT** R0,(addr),*shft*	Store Register (32 Bit)		(addr)=R0	2+		
STRcc**H** R0,(addr)	Store Register (16 bit)		(addr)=R0	2+ (H=4+)		4T+
SUBcc**S** R0, R1, R2 ,*shft*	Subtract		R0 = R1-R2	1	0010	
SWIcc #n	Software Interrupt (RST)			3		
SWPcc**B** r0,r1,[base]	Load r0 from [base],store r1 in [base]		Rd=Rn… Rn=Rd			3
TEQcc**P** R1, R2 ,*shft*	Test Inverted (EOR) (P=set flags)	teqp R4,#0	flags=R1 xor R2	1	1001	
TSTcc**P** R1, R2 ,*shft*	Test Masked (AND) (P=set flags)		flags=R1 AND R2	1	1000	
ADRcc Rn,addr	Load relative address into register		R0=addr		psuedo	
ADRcc**L** Rn,label	Load Long relative address into register				psuedo	
NOP	no operation				psuedo	
P - Alter Processor Flags	**S – Set condition codes**	**cc – Condition Code**		**B – Byte**		
H - 16 Bit **D** - 64 bit	**T- Translation (User Privilages in Super mode)**					

Arm Thumb

Command	Detail	Example	OP	N Z C V	ValidRegs
LDR Rd,[Rn,#]	LoaD Register (32 bit)	LDR r3,[r5,#0]		- - - -	R0-R7,#= 0 to 124 (Multiples of 4)
LDRB Rd,[Rn,#]	LoaD Register (8 bit)	LDRB r3,[r5,#2]		- - - -	R0-R7,#= 0 to 31 (Multiples of 1)
LDSB Rd,[Rn,#]	LoaD Register (Signed 8 bit)	LDRSB r3,[r5,#2]		- - - -	R0-R7,#= 0 to 31 (Multiples of 1)
LDRH Rd,[Rn,#]	LoaD Register (16 bit)	LDRH r3,[r5,#4]		- - - -	R0-R7,#= 0 to 62 (Multiples of 2)
LDSH Rd,[Rn,#]	LoaD Register (Signed 16 bit)	LDRSH r3,[r5,#4]		- - - -	R0-R7,#= 0 to 62 (Multiples of 2)
STR Rd,[Rn,#]	Store Register (32 Bit)	STR r3,[r5,#0]		- - - -	R0-R7,#= 0 to 124 (Multiples of 4)
STRB Rd,[Rn,#]	Store Register (8 Bit)	STRB r3,[r5,#0]		- - - -	R0-R7,#= 0 to 31 (Multiples of 1)
STRH Rd,[Rn,#]	Store Register (16 Bit)	STRH r3,[r5,#0]		- - - -	R0-R7,#= 0 to 62 (Multiples of 2)
POP {reglist}	Pop registers from the stack	POP {r1-r3,r5}		- - - -	R0-R7,LR
PUSH {reglist}	Push registers on to the stack	PUSH {r1-r3,r5}		- - - -	R0-R7,PC
LDMIA Rn!,{reglist}	Load Multiple and increment after	LDMIA R0!,{r1-r3,r5}		- - - -	R0-R7
STMIA Rn!,{reglist}	Store Multiple and increment after	STMIA R0!,{r1-r3,r5}		- - - -	R0-R7
ADD Rd,Rn,Rm	Add		Rd=Rn+Rm	N Z C V	R0-R7
ADD Rd,Rn,#	Add		Rd=Rn+#	N Z C V	R0-R7 #=0 to 7
ADD Rd,#	Add		Rd=Rd+#	N Z C V	R0-R7 #=0 to 255
SUB Rd,Rn,Rm	Subtract		Rd=Rn-Rm	N Z C V	R0-R7
SUB Rd,Rn,#	Subtract		Rd=Rn-#	N Z C V	R0-R7 #=0 to 7
SUB Rd,#	Subtract		Rd=Rd-#	N Z C V	R0-R7 #=0 to 255
ADD Rd,Rm	Add Low/High Regs (Can't both be low)		Rd=Rd+Rm	N Z C V	R0-R15,SP
ADD SP,#	Add to Stack Pointer		SP=SP+#	N Z C V	#0 to 508 (Multiple of 4)
SUB SP,#	Add to Stack Pointer		SP=SP+#	N Z C V	#0 to 508 (Multiple of 4)
ADD Rd,PC/SP,#	Add immediate to SP/PC		Rd=PC/SP+#	N Z C V	R0-R7, Rp=PC/SP #=0 to 1020 (Multiples of 4)
ADC Rd,Rm	Add with carry		Rd=Rd+Rm+C	N Z C V	R0-R7
SBC Rd,Rm	Subtract with carry		Rd=Rd-(Rm+C)	N Z C V	R0-R7
MUL Rd,Rm	Multiply		Rd=Rd*Rm	N Z - -	R0-R7
AND Rd,Rm	Logical AND		Rd=Rd AND Rm	N Z - -	R0-R7
ORR Rd,Rm	Logical OR		Rd=Rd OR Rm	N Z - -	R0-R7
EOR Rd,Rm	Logical Exclusive OR (XOR)		Rd=Rd EOR Rm	N Z - -	R0-R7
BIC Rd,Rm	Logical Bit Clear		Rd=Rd AND (NOT Rm)	N Z - -	R0-R7
ASR Rd,Rs	Arithmetic Shift Right Rs bits		Rd=Rd ASR Rs	N Z C -	R0-R7
ASR Rd,#	Arithmetic Shift Right # bits		Rd=Rd ASR #	N Z C -	R0-R7, #1 to 32
LSR Rd,Rs	Logical Shift Right Rs bits		Rd=Rd LSR Rs	N Z C -	R0-R7
LSR Rd,#	Logical Shift Right # bits		Rd=Rd LSR #	N Z C -	R0-R7, #1 to 32
LSL Rd,Rs	Logical Shift Left Rs bits		Rd=Rd LSL Rs	N Z C -	R0-R7
LSL Rd,#	Logical Shift Left # bits		Rd=Rd LSL #	N Z C -	R0-R7, #0 to 31
ROR Rd,Rs	Rotate Right Rs bits		Rd=Rd ROR Rs	N Z C -	R0-R7
CMP Rn,Rm	Compare (Set flags like SUB)		Flags=Rn-Rm	N Z C V	R0-R15
CMP Rn,#	Compare (Set flags like SUB)		Flags=Rn-#	N Z C V	R0-R7, #0 to 255
CMN Rn,Rm	Compare Negative (Set flags like ADD)		Flags=Rn+Rm	N Z C V	R0-R7
MOV Rd,#	Move Immediate		Rd=#	N Z C V	R0-R7, #0 to 255
MOV Rd,Rm	Move		Rd=Rm	N Z C V	R0-R15 (Flags unchanged R8+)
MVN Rd,Rm	Move Not (Flip bits of Rm)		Rd=NOT Rm	N Z C V	R0-R7
NEG Rd,Rm	Negate		Rd=-Rm	N Z C V	R0-R7
TST Rn,Rm	Test Masked (AND)		Flags= Rn AND Rm	N Z - -	R0-R7
B label	Branch to label			- - - -	Label= -2048 to +2048
BEQ label	Branch if Equal		Z=1	- - - -	-252 to +258
BNE label	Branch if Not Equal		Z=0	- - - -	-252 to +258
BCS label	Branch Carry Set		C=1	- - - -	-252 to +258
BHS label	Branch if Higher or Same (Unsigned)		C=1	- - - -	-252 to +258
BCC label	Branch if Carry Clear		C=0	- - - -	-252 to +258
BLO label	Branch if Lower or Same (Unsigned)		C=0	- - - -	-252 to +258
BMI label	Branch if Minus		N=1	- - - -	-252 to +258
BPL label	Branch if Plus		N=0	- - - -	-252 to +258
BVS label	Branch if oVerflow Set		V=1	- - - -	-252 to +258
BVC label	Branch if oVerflow Clear		V=0	- - - -	-252 to +258
BHI label	Branch if Higher (Unsigned)		C=1 and Z=0	- - - -	-252 to +258
BLS label	Branch if Lower or Same (Unsigned)		C=0 or Z=1	- - - -	-252 to +258
BGE label	Branch if Greater or Equal (Signed)		N=V	- - - -	-252 to +258
BLT label	Branch if Less than (Signed)		N<>V	- - - -	-252 to +258
BGT label	Branch if Greater than (Signed)		Z=0 N=V	- - - -	-252 to +258
BLE label	Branch if Less than or Equal (Signed)		Z=1 N<>V	- - - -	-252 to +258
BL label	Branch and Link		PC=label R14/LR=Return Address	- - - -	-4mb to +4mb
BX Rm	Branch and Exchange to Rm		PC=Rm	T=Bit0	R0-R15, -4mb to +4mb
SWI #	Software Interrupt			- - - -	#=0 to 255
BKPT #	Breakpoint (enter debug mode)			- - - -	#=0 to 255
ADR Rn,addr	Load address into Rn		ADD Rn,PC,#		#=0 to 1024
NOP	No operation		MOV R8,R8		

Arm 5+						
Command	Detail	Example	OP	Cycles	Opcode	Arch
BXJcc R0	Branch and change to Jazelle state					6
BKPT imm	Breakpoint					5
BLX addr,**BLX**cc R0	Branch , link and exchange					5Tb
BXcc R0	Branch and exchange		R15=Rn… Tbit=Rn[0]			5tb
CLZcc R0, R1	Count Leading Zeros					5
CPSeeff #n	Change Processor state					6
CPYcc R0, R1	Copy one register to another		R0=R1			6
LDRcc**D** R0,addr	LoaD Register (64 bit)		R0=(addr),R1=(addr+4)	3+		5TE
LDREXcc R0,R1	LoaD Register and set memory exclusive		R0=(R1)	3+		6
MAR	Mover from registers to 40 bit acc					Xscale
MCRRcc #,e,Rd,Rn,Crn,Crm,e2 **MCRR2**	Move from 2 registers to coprocessor					5TE,6
MIA,MIAPH,MIAxy	Multiply with internal 40 bit accumulate					Xscale
MRA	Multiply from 40 bit accumulator to registers					Xscale
MRRCcc **#**,e,Rd,Rn,Crm, **MRC2**	Move from coprocessor to 2 regs					5E
PKHBTcc R0, R1, R2 ,*shft*	Pack Halfword Bottom/Top (L from R1 / H from R2)		R0=R2H+R1L			6
PKHTBcc R0, R1, R2 ,*shft*	Pack Halfword Top/Bottom (H from R1 / L from R2)		R0=R1H+R2L			6
PLD mode	Cache Preload					5E
QADDcc R0, R1, R2	Saturating Arithmatic					5Exp
QADD16cc R0, R1, R2	Saturating Arithmatic (16 bit)					6
QADD8cc R0, R1, R2	Saturating Arithmatic (8 bit)					6
QADDSUBXcc R0, R1, R2	Saturating Add and Subtract with Exchange					6
QDADDcc R0, R1, R2	Saturating Double and Add					5TE
QDSUBcc R0, R1, R2	Saturating Double and Subtract					5TE
QSUBcc R0, R1, R2	Saturating Subtract					5TE
QSUB16cc R0, R1, R2	Saturating Subtract (16 bit)					6
QSUB8cc R0, R1, R2	Saturating Subtract (8 bit)					6
QSUBADDXcc R0, R1, R2	Saturating Add and Subtract with Exchange					6
REVcc R0, R1	reverses the byte order in a 32-bit register.					6
REV16cc R0, R1	reverses the byte order in a 16-bit register.					6
REVSHcc R0, R1	reverses the byte order in a 16-bit register, and sign extend					6
RFE<mode> R0!	Return From Exception					6
SADD16cc R0, R1, R2	Signed Add two 16 bit numbers					6
SADD8cc R0, R1, R2	Signed Add four 8-bit signed integer additions					6
SADDSUBXcc R0, R1, R2	Signed 16-bit Add and Subtract with Exchange					6
SELcc R0, R1, R2	Select bytes from R1/R2 based on GE flags					6
SETEND <endian>	Set Endian mode					6
SHADD16cc R0, R1, R2	Signed Halving Add (16 bit)					6
SHADD8cc R0, R1, R2	Signed Halving Add (8 bit)					6
SHADDSUBXcc R0, R1, R2	Signed Halving Add and Subtract with Exchange (16 bit)					6
SHSUB16cc R0, R1, R2	Signed Halving Subtract (16 bit)					6
SHSUB8cc R0, R1, R2	Signed Halving Subtract (8 bit)					6
SHSUBADDXcc R0, R1, R2	Signed Halving Subtract and Add with Exchange (16 bit)					6
SMLALxycc R0L, R1H, R2,R3	Signed Multiply-accumulate Long					5TE
SMLAxycc	Signed Multiply-accumulate					5TE
SMLADXcc	Signed Multiply-accumulate Dual					6
SMLALcc**S** R0L, R1H, R2,R3	Signed Multiply-accumulate Long					6
SMLAWycc	Signed Multiply-accumulate Word B and T					5ExP
SMLSDXcc R0, R1, R2,R3	Signed Multiply Subtract accumulate Dual					6
SMLSLDXcc R0, R1, R2,R3	Signed Multiply Subtract accumulate LongDual					6
SMMLARcc R0, R1, R2,R3	Signed Most significant word Multiply Accumulate					6
SMMLSRcc R0, R1, R2,R3	Signed Most significant word Multiply Subtract					6
SMULLRcc R0, R1, R2	Signed Multiply (R=Round)					6
SMUADXcc R0, R1, R2	Signed Dual Multiply Add					6
SMULXYcc R0, R1, R2	Signed Multiply BB , BT , TB , or TT					ARMv5TE
SMULLcc R0L, R1H, R2,R3	Signed Multiply Long					ARMv5TE
SMULWYcc R0, R1, R2	Signed Multiply Word B and T					ARMv5TE
SMUSDXcc R0, R1, R2	Signed Dual Multiply Subtract					6
SRS<Mode> #mode!	Store Return State					6
SSAT16cc R0,#n, R1,*shft*	Signed Saturate (16 bit)					6
SSATcc R0,#n, R1,*shft*	Signed Saturate					6
SSUB16cc R0, R1, R2	Signed Subtract (16 bit)					6
SSUB8cc R0, R1, R2	Signed Subtract (8 bit)					6
SSUBADDXcc R0, R1, R2	Signed Subtract and Add with Exchange (16 bit)					6
STRcc**D** R0,(addr)	Store Register (64 bit)		(addr)=R0,(addr+4)=R1	2+		ARMv5TE
STREXcc R0,R1,R2	Store Register Exclusive					6
SXTABcc R0,R1,R2,*shft*	Extract an 8 bit value, and sign extend					6
SXTAB16cc R0,R1,R2,*shft*	Extract two 8 bit value, and sign extend to 16 bits					6
SXTAHcc R0,R1,R2,*shft*	Extract a 16 bit value, and sign extend					6
SXTBcc R0,R1,*shft*	Take a 8-bit value from a register and sign extends it to 32 bits.					6
SXTB16cc R0,R1,*shft*	Take two 8-bit value from a register and sign extends it to 16 bits.					6
SXTHcc R0,R1,*shft*	Take two 16-bit value from a register and sign extend to 32 bits					6
UADD16cc R0,R1,R2	Unsigned Add (16 bit)					6
UADD8cc R0,R1,R2	Unsigned Add (8 bit)					6
UADDSUBXcc R0,R1,R2	Unsigned Add and Subtract with Exchange					6
UHADD16cc R0,R1,R2	Unsigned Halving Add (16 bit)					6
UHADD8cc R0,R1,R2	Unsigned Halving Add (8 bit)					6
UHSUB16cc R0,R1,R2	Unsigned Halving Subtract (16 bit)					6
UHSUB8cc R0,R1,R2	Unsigned Halving Subtract (8 bit)					6
USUBADDXcc R0,R1,R2	Unsigned Subtract and Add with Exchange					6
UMAALcc**S R0L, R1H, R2,R3**	Unsigned Multiply Accumulate Long					6
UMULLcc**S R0L, R1H, R2,R3**	Unsigned Multiply Long					6
UQADD16cc R0,R1,R2	Unsigned Saturating Add (16 bit)					6
UQADD8cc R0,R1,R2	Unsigned Saturating Add (8 bit)					6
UQADDSUBXcc R0,R1,R2	Unsigned Saturating Add and Subtract with Exchange					6
UQSUB16cc R0,R1,R2	Unsigned Saturating Subtract (16 bit)					6
UQSUB8cc R0,R1,R2	Unsigned Saturating Subtract (8 bit)					6
UQSUBADDXcc R0,R1,R2	Unsigned Saturating Subtract and Add with Exchange					6
USAD8cc R0,R1,R2	Unsigned Sum of Absolute Differences					6
USADA8cc R0,R1,R2,R3	Unsigned Sum of Absolute Differences and Accumulate					6
USATcc R0,#n, R1,*shft*	Unsigned Saturate					6
USAT16cc R0,#n, R1,*shft*	Unsigned Saturate (16 bit)					6
USUB16cc R0,R1,R2	Unsigned Subtract (16 bit)					6
USUB8cc R0,R1,R2	Unsigned Subtract (8 bit)					6
USUBADDXcc R0,R1,R2	Unsigned Subtract and Add with Exchange					6
UXTABcc R0,R1,R2,*shft*	Extract an 8 bit value and Zero extend					6
UXTAB16cc R0,R1,R2,*shft*	Extract two 8 bit values and Zero extend					6
UXTAHcc R0,R1,R2,*shft*	Extract an 16 bit value and Zero extend					6
UXTBcc R0,R1,*shft*	Extract an 8 bit value and Zero extend					6
UXTB16cc R0,R1,*shft*	Extract two 8 bit values and Zero extend					6
UXTHcc R0,R1,*shft*	Extract a 16 bit value and Zero Extend					6
P - Alter Processor Flags	**S – Set condition codes**	**cc – Condition Code**		**B – Byte**		
H - 16 Bit **D** - 64 bit	**T- Translation (User Privilages in Super mode)**					