Opcode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSL Rd, Rm, #	0	0	0	0	0	#					Rm			Rd		
LSR Rd, Rm, #	0	0	0	0	1	#					Rm			Rd		
ASR Rd, Rm, #	0	0	0	1	0	#					Rm			Rd		
ADD Rd, Rn, Rm	0	0	0	1	1	0	0	Rm			Rn			Rd		
SUB Rd, Rn, Rm	0	0	0	1	1	0	1	Rm			Rn			Rd		
ADD Rd, Rn, #	0	0	0	1	1	1	0	#			Rn			Rd		
SUB Rd, Rn, #	0	0	0	1	1	1	1	#			Rn			Rd		
MOV Rd, #	0	0	1	0	0	Rd			#							
CMP Rn, #	0	0	1	0	1	Rn			#							
ADD Rd, #	0	0	1	1	0	Rd			#							
SUB Rd, #	0	0	1	1	1	Rd			#							
AND Rd, Rm	0	1	0	0	0	0	0	0	0	0	Rm			Rd		
EOR Rd, Rm	0	1	0	0	0	0	0	0	0	1	Rm			Rd		
LSL Rd, Rs	0	1	0	0	0	0	0	0	1	0	Rs			Rd		
LSR Rd, Rs	0	1	0	0	0	0	0	0	1	1	Rs			Rd		
ASR Rd, Rs	0	1	0	0	0	0	0	1	0	0	Rs			Rd		
ADC Rd, Rm	0	1	0	0	0	0	0	1	0	1	Rm			Rd		
SBC Rd, Rm	0	1	0	0	0	0	0	1	1	0	Rm			Rd		
ROR Rd, Rs	0	1	0	0	0	0	0	1	1	1	Rs			Rd		
TST Rm, Rn	0	1	0	0	0	0	1	0	0	0	Rn			Rm		
NEG Rd, Rm	0	1	0	0	0	0	1	0	0	1	Rm			Rd		
CMP Rm, Rn	0	1	0	0	0	0	1	0	1	0	Rn			Rm		
CMN Rm, Rn	0	1	0	0	0	0	1	0	1	1	Rn			Rm		
ORR Rd, Rm	0	1	0	0	0	0	1	1	0	0	Rm			Rd		
MUL Rd, Rm	0	1	0	0	0	0	1	1	0	1	Rm			Rd		
BIC Rm, Rd	0	1	0	0	0	0	1	1	1	0	Rn			Rm		
MVN Rd, Rm	0	1	0	0	0	0	1	1	1	1	Rm			Rd		
Unpredictable	0	1	0	0	0	1	0	0	0	0	x	x	x	x	x	x
ADD Rd, Rm	0	1	0	0	0	1	0	0	H1	H2	Rm			Rd		
Unpredictable	0	1	0	0	0	1	0	1	0	0	x	x	x	x	x	x
CMP Rm, Rn	0	1	0	0	0	1	0	1	H1	H2	Rn			Rm		
Unpredictable	0	1	0	0	0	1	1	0	0	0	x	x	x	x	x	x
MOV Rd, Rm	0	1	0	0	0	1	1	0	H1	H2	Rm			Rd		
BX Rm	0	1	0	0	0	1	1	1	0	H2	Rm			0	0	0
BLX Rm	0	1	0	0	0	1	1	1	1	H2	Rm			0	0	0
Unpredictable	0	1	0	0	0	1	1	1	x	x	x	x	x	1	x	x
LDR Rd, [PC, #]	0	1	0	0	1	Rd			PC Relative Offset							
STR Rd, [Rn, Rm]	0	1	0	1	0	0	0	Rm			Rn			Rd		
STRH Rd, [Rn, Rm]	0	1	0	1	0	0	1	Rm			Rn			Rd		
STRB Rd, [Rn, Rm]	0	1	0	1	0	1	0	Rm			Rn			Rd		
LDRSB Rd, [Rn, Rm]	0	1	0	1	0	1	1	Rm			Rn			Rd		
LDR Rd, [Rn, Rm]	0	1	0	1	1	0	0	Rm			Rn			Rd		
LDRH Rd, [Rn, Rm]	0	1	0	1	1	0	1	Rm			Rn			Rd		
LDRB Rd, [Rn, Rm]	0	1	0	1	1	1	0	Rm			Rn			Rd		
LDRSH Rd, [Rn, Rm]	0	1	0	1	1	1	1	Rm			Rn			Rd		
STR Rd, [Rn, #OFF]	0	1	1	0	0	# Offset					Rn			Rd		
LDR Rd, [Rn, #OFF]	0	1	1	0	1	# Offset					Rn			Rd		
STRB Rd, [Rn, #OFF]	0	1	1	1	0	# Offset					Rn			Rd		
LDRB Rd, [Rn, #OFF]	0	1	1	1	1	# Offset					Rn			Rd		
STRH Rd, [Rn, #OFF]	1	0	0	0	0	# Offset					Rn			Rd		
LDRH Rd, [Rn, #OFF]	1	0	0	0	1	# Offset					Rn			Rd		
STR Rd, [SP, #OFF]	1	0	0	1	0	Rd			SP Relative Offset							
LDR Rd, [SP, #OFF]	1	0	0	1	1	Rd			SP Relative Offset							
ADD Rd, PC, #OFF	1	0	1	0	0	Rd			PC Relative Offset							
ADD Rd, SP, #OFF	1	0	1	0	1	Rd			SP Relative Offset							
SUB SP, SP, #OFF	1	0	1	1	0	0	0	0	1	SP Relative Offset						
Unpredictable	1	0	1	1	0	0	0	1	x	x	x	x	x	x	x	x
PUSH {<reg list>, <LR>}	1	0	1	1	0	1	0	LR	Register List							
POP {<reg list>, <PC>}	1	0	1	1	1	1	0	PC	Register List							
Unpredictable	1	0	1	1	0	x	1	x	x	x	x	x	x	x	x	x
Unpredictable	1	0	1	1	1	0	x	x	x	x	x	x	x	x	x	x
BKPT #	1	0	1	1	1	1	1	0	#							
Unpredictable	1	0	1	1	1	1	1	1	x	x	x	x	x	x	x	x
STMIA Rn!, {<reg list>}	1	1	0	0	0	Rn			Register List							
LDMIA Rn!, {<reg list>}	1	1	0	0	1	Rn			Register List							
B{<cond>} <Target Addr>	1	1	0	1	cond				# Offset							
Unused Opcode	1	1	0	1	1	1	1	0	x	x	x	x	x	x	x	x
SWI #	1	1	0	1	1	1	1	1	#							
B <Target Addr>	1	1	1	0	0	# Offset										
BLX <Target Addr>	1	1	1	0	1	# Offset (lower half)										
BL{X} <Target Addr> (+)	1	1	1	1	0	# Offset (upper half)										
BL <Target Addr>	1	1	1	1	1	# Offset (lower half)										

rE Ejected
re-eject.gbadev.org
ARM Thumb Reference V2

rev	Mnemonic	Definition	Alternate Description
	ADC	Add with Carry	ADD numbers and Carry bit
	ADD	Add	ADD numbers
	AND	Logical AND	AND together numbers
	ASR	Arithmetic Shift Right	Signed Right Shift (>>)
	B	Branch	Jump to an address
	BIC	Bit Clear	AND's the compliment of a number
v5	BKPT	Breakpoint	Software Breakpoint
	BL	Branch with Link	Jump to an address, and set LR to return address
v5	BLX	Branch with Link and Exchange	Jump to an address, set LR to return address, switch operating modes
	BX	Branch and Exchange	Jump to an address, and switch operating modes
	CMN	Compare Negative	Compare numbers by addition
	CMP	Compare	Compare numbers by subtraction
	EOR	Logical Exclusive OR (XOR)	Exclusive OR together numbers
	LDMIA	Load Multiple, Increment After	Load Multiple registers at once
	LDR	Load Register (word)	Load an unsigned 32bit number into a register
	LDRB	Load Register (byte)	Load an unsigned 8bit number into a register
	LDRH	Load Register (halfword)	Load an unsigned 16bit number into a register
	LDRSB	Load Register (byte)	Load a signed 8bit number into a register
	LDRSH	Load Register (halfword)	Load a signed 16bit number into a register
	LSL	Logical Shift Left	Unsigned Left Shift (<<)
	LSR	Logical Shift Right	Unsigned Right Shift (>>)
	MOV	Move	Move a number
	MUL	Multiply	Multiply numbers
	MVN	Move Not	Compliment a number
	NEG	Negate	Negate a number
	ORR	Logical OR	OR together numbers
	POP	Pop multiple registers	Takes numbers off the stack
	PUSH	Push multiple registers	Puts numbers on to the stack
	ROR	Rotate Right Register	Shifts right (>>), and numbers shifted off are appended to top
	SBC	Subtract with Carry	Subtract numbers and ADD Carry bit
	STMIA	Store Multiple, Increment After	Store Multiple registers at once
	STR	Store Register (word)	Store a 32bit number into an address
	STRB	Store Register (byte)	Store an 8bit number into an address
	STRH	Store Register (halfword)	Store a 16bit number into an address
	SUB	Subtract	Subtract numbers
	SWI	Software Interrupt	Execute code/"bios" calls
	TST	Test	Checks if one of more bits are set
	Unused	Unused Opcode	Future revisions of the Architecture will not use this space

Opcode	Work	Notes	Z	C	N	V
ADC Rd, Rm	Rd = Rd + Rm + C	-	x	x	x	x
ADD Rd, #	Rd = Rd + #	-	x	x	x	x
ADD Rd, PC, #OFF	Rd = Rd + (PC + (#OFF << 2))	-				
ADD Rd, Rm	Rd = Rd + Rm	Rd or Rm must be a *high register*				
ADD Rd, Rn, #	Rd = Rn + #	-	x	x	x	x
ADD Rd, Rn, Rm	Rd = Rm + Rn	-	x	x	x	x
ADD Rd, SP, #OFF	Rd = SP + (#OFF << 2)	-				
AND Rd, Rm	Rd = Rd & Rm	-	x	x	x	x
ASR Rd, Rm, #	Rd = Rm >> #	signed	x	x	x	
ASR Rd, Rs	Rd = Rm >> Rs	signed	x	x	x	
B <Target Addr>	PC = PC + (#OFF << 1)	-				
B{<cond>} <Target Addr>	PC = PC + (#OFF << 1)	If <cond> is true				
BIC Rm, Rd	Rd = Rd & !(Rm)	-	x	x	x	
BKPT #	CALL Breakpoint with #	v5 only. v4 it does nothing				
BL <Target Addr>	See Branching Description	-				
BLX <Target Addr>	See Branching Description	-				
BLX Rm	LR = (PC + 2) | 1; PC = Rm[31..1] << 1; T=Rm[0]	-				
BX Rm	PC = Rm[31..1] << 1; T = Rm[0]	-				
CMN Rm, Rn	<flags> = Rm + Rn	-	x	x	x	x
CMP Rm, Rn	<flags> = Rm - Rn	-	x	x	x	x
CMP Rm, Rn	<flags> = Rm - Rn	Rm or Rn must be a *high register*	x	x	x	x
CMP Rn, #	<flags> = Rm - #	-	x	x	x	x
EOR Rd, Rm	Rd = Rd ^ Rm	-	x	x	x	x
LDMIA Rn!, {<reg list>}	for each in <reg list> = [Rn+=4]	-				
LDR Rd, [PC, #OFF]	Rd = [PC + (#OFF << 2)]	Word				
LDR Rd, [Rn, #OFF]	Rd = [Rn + (#OFF << 2)]	Word				
LDR Rd, [Rn, Rm]	Rd = [Rn + Rm]	Word				
LDR Rd, [SP, #OFF]	Rd = [SP + (#OFF << 2)]	Word				
LDRB Rd, [Rn, #OFF]	Rd = [Rn + (#OFF << 2)]	Unsigned Byte				
LDRB Rd, [Rn, Rm]	Rd = [Rn + Rm]	Unsigned Byte				
LDRH Rd, [Rn, #OFF]	Rd = [Rn + (#OFF << 2)]	Unsigned Halfword				
LDRH Rd, [Rn, Rm]	Rd = [Rn + Rm]	Unsigned Halfword				
LDRSB Rd, [Rn, Rm]	Rd = [Rn + Rm]	Signed Byte				
LDRSH Rd, [Rn, Rm]	Rd = [Rn + Rm]	Signed Halfword				
LSL Rd, Rm, #	Rd = Rm << #	Unsigned/Signed	x	x	x	
LSL Rd, Rs	Rd = Rm << Rs	Unsigned/Signed	x	x	x	
LSR Rd, Rm, #	Rd = Rm >> #	Unsigned	x	x	x	
LSR Rd, Rs	Rd = Rm >> Rs	Unsigned	x	x	x	
MOV Rd, #	Rd = #	-	x		x	
MOV Rd, Rm	Rd = Rm	Rd or Rm must be a *high register*				
MUL Rd, Rm	Rd = Rd * Rm	-	x	x	x	
MVN Rd, Rm	Rd = !(Rm)	-	x		x	
NEG Rd, Rm	Rd = -(Rm)	-	x	x	x	x
ORR Rd, Rm	Rd = Rd | Rm	-	x	x	x	
POP {<reg list>, <PC>}	get <reg list> and/or <PC> from stack	-				
PUSH {<reg list>, <LR>}	put <reg list> and/or <LR> on stack	-				
ROR Rd, Rs	Rd = Rd >|> Rs	-	x	x	x	
SBC Rd, Rm	Rd = (Rd - Rm) + C	-	x	x	x	x
STMIA Rn!, {<reg list>}	[Rn+=4] = for each in <reg list>	-				
STR Rd, [Rn, #OFF]	[Rn + (#OFF << 2)] = Rd	word				
STR Rd, [Rn, Rm]	[Rn + Rm] = Rd	word				
STR Rd, [SP, #OFF]	[SP + (#OFF << 2)] = Rd	word				
STRB Rd, [Rn, #OFF]	[Rn + (#OFF << 2)] = Rd	byte				
STRB Rd, [Rn, Rm]	[Rn + Rm] = Rd	byte				
STRH Rd, [Rn, #OFF]	[Rn + (#OFF << 2)] = Rd	halfword				
STRH Rd, [Rn, Rm]	[Rn + Rm] = Rd	halfword				
SUB Rd, #	Rd = Rd - #	-	x	x	x	x
SUB Rd, Rn, #	Rd = Rn - #	-	x	x	x	x
SUB Rd, Rn, Rm	Rd = Rn - Rm	-	x	x	x	x
SUB SP, #OFF	SP = SP - (#OFF << 2)	-				
SWI #	Run "bios" function	-				
TST Rm, Rn	<flags> = Rn & Rm	-	x		x	
Unused Opcode	none	Free for software use. Minimal risk of future CPU revisions turning this into an opcode.				

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	-

Opcode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC Rd, Rm	0	1	0	0	0	0	0	1	0	1	Rm			Rd		
ADD Rd, #	0	0	1	1	0	Rd			#							
ADD Rd, PC, #OFF	1	0	1	0	0	Rd			PC Relative Offset							
ADD Rd, Rm	0	1	0	0	0	1	0	0	H1	H2	Rm			Rd		
ADD Rd, Rn, #	0	0	0	1	1	1	0	#			Rn			Rd		
ADD Rd, Rn, Rm	0	0	0	1	1	0	0	Rm			Rn			Rd		
ADD Rd, SP, #OFF	1	0	1	0	1	Rd			SP Relative Offset							
AND Rd, Rm	0	1	0	0	0	0	0	0	0	0	Rm			Rd		
ASR Rd, Rm, #	0	0	0	1	0	#					Rm			Rd		
ASR Rd, Rs	0	1	0	0	0	0	0	1	0	0	Rs			Rd		
B <Target Addr>	1	1	1	0	0	# Offset										
B{<cond>} <Target Addr>	1	1	0	1	cond				# Offset							
BIC Rm, Rd	0	1	0	0	0	0	1	1	1	0	Rn			Rm		
BKPT #	1	0	1	1	1	1	1	0	#							
BL <Target Addr>	1	1	1	1	1	# Offset (lower half)										
BL{X} <Target Addr> (+)	1	1	1	1	0	# Offset (upper half)										
BLX <Target Addr>	1	1	1	0	1	# Offset (lower half)										
BLX Rm	0	1	0	0	0	1	1	1	1	H2	Rm			0	0	0
BX Rm	0	1	0	0	0	1	1	1	0	H2	Rm			0	0	0
CMN Rm, Rn	0	1	0	0	0	0	1	0	1	1	Rn			Rm		
CMP Rm, Rn	0	1	0	0	0	0	1	0	1	0	Rn			Rm		
CMP Rm, Rn	0	1	0	0	0	1	0	1	H1	H2	Rn			Rm		
CMP Rn, #	0	0	1	0	1	Rn			#							
EOR Rd, Rm	0	1	0	0	0	0	0	0	0	1	Rm			Rd		
LDMIA Rn!, {<reg list>}	1	1	0	0	1	Rn			Register List							
LDR Rd, [PC, #]	0	1	0	0	1	Rd			PC Relative Offset							
LDR Rd, [Rn, #OFF]	0	1	1	0	1	# Offset					Rn			Rd		
LDR Rd, [Rn, Rm]	0	1	0	1	1	0	0	Rm			Rn			Rd		
LDR Rd, [SP, #OFF]	1	0	0	1	1	Rd			SP Relative Offset							
LDRB Rd, [Rn, #OFF]	0	1	1	1	1	# Offset					Rn			Rd		
LDRB Rd, [Rn, Rm]	0	1	0	1	1	1	0	Rm			Rn			Rd		
LDRH Rd, [Rn, #OFF]	1	0	0	0	1	# Offset					Rn			Rd		
LDRH Rd, [Rn, Rm]	0	1	0	1	1	0	1	Rm			Rn			Rd		
LDRSB Rd, [Rn, Rm]	0	1	0	1	0	1	1	Rm			Rn			Rd		
LDRSH Rd, [Rn, Rm]	0	1	0	1	1	1	1	Rm			Rn			Rd		
LSL Rd, Rm, #	0	0	0	0	0	#					Rm			Rd		
LSL Rd, Rs	0	1	0	0	0	0	0	0	1	0	Rs			Rd		
LSR Rd, Rm, #	0	0	0	0	1	#					Rm			Rd		
LSR Rd, Rs	0	1	0	0	0	0	0	0	1	1	Rs			Rd		
MOV Rd, #	0	0	1	0	0	Rd			#							
MOV Rd, Rm	0	1	0	0	0	1	1	0	H1	H2	Rm			Rd		
MUL Rd, Rm	0	1	0	0	0	0	1	1	0	1	Rm			Rd		
MVN Rd, Rm	0	1	0	0	0	0	1	1	1	1	Rm			Rd		
NEG Rd, Rm	0	1	0	0	0	0	1	0	0	1	Rm			Rd		
ORR Rd, Rm	0	1	0	0	0	0	1	1	0	0	Rm			Rd		
POP {<reg list>, <PC>}	1	0	1	1	1	1	0	PC	Register List							
PUSH {<reg list>, <LR>}	1	0	1	1	0	1	0	LR	Register List							
ROR Rd, Rs	0	1	0	0	0	0	0	1	1	1	Rs			Rd		
SBC Rd, Rm	0	1	0	0	0	0	0	1	1	0	Rm			Rd		
STMIA Rn!, {<reg list>}	1	1	0	0	0	Rn			Register List							
STR Rd, [Rn, #OFF]	0	1	1	0	0	# Offset					Rn			Rd		
STR Rd, [Rn, Rm]	0	1	0	1	0	0	0	Rm			Rn			Rd		
STR Rd, [SP, #OFF]	1	0	0	1	0	Rd			SP Relative Offset							
STRB Rd, [Rn, #OFF]	0	1	1	1	0	# Offset					Rn			Rd		
STRB Rd, [Rn, Rm]	0	1	0	1	0	1	0	Rm			Rn			Rd		
STRH Rd, [Rn, #OFF]	1	0	0	0	0	# Offset					Rn			Rd		
STRH Rd, [Rn, Rm]	0	1	0	1	0	0	1	Rm			Rn			Rd		
SUB Rd, #	0	0	1	1	1	Rd			#							
SUB Rd, Rn, #	0	0	0	1	1	1	1	#			Rn			Rd		
SUB Rd, Rn, Rm	0	0	0	1	1	0	1	Rm			Rn			Rd		
SUB SP, SP, #OFF	1	0	1	1	0	0	0	0	1	SP Relative Offset						
SWI #	1	1	0	1	1	1	1	1	#							
TST Rm, Rn	0	1	0	0	0	0	1	0	0	0	Rn			Rm		
Unpredictable	0	1	0	0	0	1	0	0	0	0	x	x	x	x	x	x
Unpredictable	0	1	0	0	0	1	0	1	0	0	x	x	x	x	x	x
Unpredictable	0	1	0	0	0	1	1	0	0	0	x	x	x	x	x	x
Unpredictable	0	1	0	0	0	1	1	1	x	x	x	x	x	1	x	x
Unpredictable	1	0	1	1	0	0	0	1	x	x	x	x	x	x	x	x
Unpredictable	1	0	1	1	0	x	1	x	x	x	x	x	x	x	x	x
Unpredictable	1	0	1	1	1	0	x	x	x	x	x	x	x	x	x	x
Unpredictable	1	0	1	1	1	1	1	1	x	x	x	x	x	x	x	x
Unused Opcode	1	1	0	1	1	1	1	0	x	x	x	x	x	x	x	x