

Opcode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSL Rd, Rn, #	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LSR Rd, Rn, #	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ASR Rd, Rn, #	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ADD Rd, Rn, Rm	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
SUB Rd, Rn, Rm	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
ADD Rd, Rn, #	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
SUB Rd, Rn, #	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
MOV Rd, #	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CMP Rn, #	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ADD Rd, #	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
SUB Rd, #	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0
AND Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EOR Rd, Rm	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
ORR Rd, Rm	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
LSL Rd, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LSR Rd, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ASR Rd, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADC Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SBC Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROR Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TST Rm, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NEG Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ORR Rm, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CMP Rm, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CMN Rm, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ORR Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MUL Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BIC Rm, Rd	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MVN Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADD Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CMP Rm, Rn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MOV Rd, Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLX Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLX Rm	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDR Rd, [PC, #]	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
STR Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STRH Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STRB Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDRSB Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDR Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDRH Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDRB Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDRSB Rd, [Rn, Rm]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STR Rd, [Rn, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDR Rd, [Rn, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STRB Rd, [Rn, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDRB Rd, [Rn, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STRH Rd, [Rn, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDRH Rd, [Rn, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STR Rd, [SP, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDR Rd, [SP, #OFF]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADD Rd, PC, #OFF	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADD Rd, SP, #OFF	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SUB SP, SP, #OFF	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PUSH {<reg list>, <LR>}	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
POP {<reg list>, <PC>}	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
BKPT #	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Unpredictable	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STIA Rn1, {<reg list>}	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDIA Rn1, {<reg list>}	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B{<cond>} <Target Addr>	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Unused Opcode	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
SWI #	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
B{<cond>} <Target Addr>	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLX <Target Addr>	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BL{X} <Target Addr> (+)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BL <Target Addr>	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0



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
v5	BIC	Bit Clear	AND's the complement of a number
	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
	LDIA	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	Complement 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	Push 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
ADD Rd, #	Rd = Rd + #	-	-	X	X	X
ADD Rd, PC, #OFF	Rd = Rd + (PC + (#OFF << 2))	-	-	X	X	X
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)	-	-	X	X	X
AND Rd, Rm	Rd = Rd & Rm	-	X	X	X	X
ASR Rd, Rm, #	Rd = Rm >> #	signed	X	X	X	X
ASR Rd, Rn, Rm	Rd = Rm >> Rn	signed	X	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	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; Tail[0]	-	-	-	-	-
BLX 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
LDIA Rn1, {<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 Half word				
LDRH Rd, [Rn, Rm]	Rd = [Rn + Rm]	Unsigned Half word				
LDRSB Rd, [Rn, Rm]	Rd = [Rn + Rm]	Signed Byte				
LDRSH Rd, [Rn, Rm]	Rd = [Rn + Rm]	Signed Half word				
LSL Rd, Rm, #	Rd = Rm << #	Unsigned/Signed	X	X	X	X
LSL Rd, Rn, Rm	Rd = Rm << Rn	Unsigned/Signed	X	X	X	X
LSR Rd, Rm, #	Rd = Rm >> #	Unsigned	X	X	X	X
LSR Rd, Rn, Rm	Rd = Rm >> Rn	Unsigned	X	X	X	X
MOV Rd, #	R					