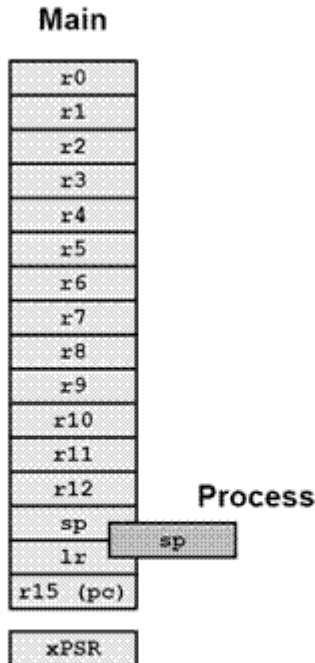


Cortex-M3 Instruction Set Summary



The processor implements the ARMv7-M Thumb instruction set. Table 1 shows the Cortex-M3 instructions and their cycle counts. The cycle counts are based on a system with zero wait states. Within the assembler syntax, depending on the operation, the `<op2>` field can be replaced with one of the following options:

- a simple register specifier, for example `Rm`
- an immediate shifted register, for example `Rm, LSL #4`
- a register shifted register, for example `Rm, LSL Rs`
- an immediate value, for example `#0xE000E000`.

For brevity, not all load and store addressing modes are shown. See the *ARMv7-M Architecture Reference Manual* for more information. Table 1 uses the following abbreviations in the Cycles column:

P - The number of cycles required for a pipeline refill. This ranges from 1 to 3 depending on the alignment and width of the target instruction, and whether the processor manages to speculate the address early.

B - The number of cycles required to perform the barrier operation. For `DSB` and `DMB`, the minimum

number of cycles is zero. For `ISB`, the minimum number of cycles is equivalent to the number required for a pipeline refill.

N - The number of registers in the register list to be loaded or stored, including PC or LR.

W - The number of cycles spent waiting for an appropriate event.

Table 1 Cortex-M3 instruction set summary

Operation	Description	Assembler	Cycles
Move	Register	<code>MOV Rd, <op2></code>	1
	16-bit immediate	<code>MOVW Rd, #<imm></code>	1
	Immediate into top	<code>MOVT Rd, #<imm></code>	1
	To PC	<code>MOV PC, Rm</code>	1 + P
Add	Add	<code>ADD Rd, Rn, <op2></code>	1

Operation	Description	Assembler	Cycles
	Add to PC	ADD PC, PC, Rm	1 + P
	Add with carry	ADC Rd, Rn, <op2>	1
	Form address	ADR Rd, <label>	1
Subtract	Subtract	SUB Rd, Rn, <op2>	1
	Subtract with borrow	SBC Rd, Rn, <op2>	1
	Reverse	RSB Rd, Rn, <op2>	1
Multiply	Multiply	MUL Rd, Rn, Rm	1
	Multiply accumulate	MLA Rd, Rn, Rm	2
	Multiply subtract	MLS Rd, Rn, Rm	2
	Long signed	SMULL RdLo, RdHi, Rn, Rm	3 to 5 ^[a]
	Long unsigned	UMULL RdLo, RdHi, Rn, Rm	3 to 5 ^[a]
	Long signed accumulate	SMLAL RdLo, RdHi, Rn, Rm	4 to 7 ^[a]
	Long unsigned accumulate	UMLAL RdLo, RdHi, Rn, Rm	4 to 7 ^[a]
Divide	Signed	SDIV Rd, Rn, Rm	2 to 12 ^[b]
	Unsigned	UDIV Rd, Rn, Rm	2 to 12 ^[b]
Saturate	Signed	SSAT Rd, #<imm>, <op2>	1
	Unsigned	USAT Rd, #<imm>, <op2>	1
Compare	Compare	CMP Rn, <op2>	1
	Negative	CMN Rn, <op2>	1
Logical	AND	AND Rd, Rn, <op2>	1
	Exclusive OR	EOR Rd, Rn, <op2>	1
	OR	ORR Rd, Rn, <op2>	1

Operation	Description	Assembler	Cycles
	OR NOT	ORN Rd, Rn, <op2>	1
	Bit clear	BIC Rd, Rn, <op2>	1
	Move NOT	MVN Rd, <op2>	1
	AND test	TST Rn, <op2>	1
	Exclusive OR test	TEQ Rn, <op1>	
Shift	Logical shift left	LSL Rd, Rn, #<imm>	1
	Logical shift left	LSL Rd, Rn, Rs	1
	Logical shift right	LSR Rd, Rn, #<imm>	1
	Logical shift right	LSR Rd, Rn, Rs	1
	Arithmetic shift right	ASR Rd, Rn, #<imm>	1
	Arithmetic shift right	ASR Rd, Rn, Rs	1
Rotate	Rotate right	ROR Rd, Rn, #<imm>	1
	Rotate right	ROR Rd, Rn, Rs	1
	With extension	RRX Rd, Rn	1
Count	Leading zeroes	CLZ Rd, Rn	1
Load	Word	LDR Rd, [Rn, <op2>]	2 ^[C]
	To PC	LDR PC, [Rn, <op2>]	2 ^[C] + P
	Halfword	LDRH Rd, [Rn, <op2>]	2 ^[C]
	Byte	LDRB Rd, [Rn, <op2>]	2 ^[C]
	Signed halfword	LDRSH Rd, [Rn, <op2>]	2 ^[C]
	Signed byte	LDRSB Rd, [Rn, <op2>]	2 ^[C]
	User word	LDRT Rd, [Rn, #<imm>]	2 ^[C]
	User halfword	LDRHT Rd, [Rn, #<imm>]	2 ^[C]
	User byte	LDRBT Rd, [Rn, #<imm>]	2 ^[C]

Operation	Description	Assembler	Cycles
	User signed halfword	LDRSHT Rd, [Rn, #<imm>]	2 _[c]
	User signed byte	LDRSBT Rd, [Rn, #<imm>]	2 _[c]
	PC relative	LDR Rd,[PC, #<imm>]	2 _[c]
	Doubleword	LDRD Rd, Rd, [Rn, #<imm>]	1 + N
	Multiple	LDM Rn, {<reglist>}	1 + N
	Multiple including PC	LDM Rn, {<reglist>, PC}	1 + N + P
Store	Word	STR Rd, [Rn, <op2>]	2 _[c]
	Halfword	STRH Rd, [Rn, <op2>]	2 _[c]
	Byte	STRB Rd, [Rn, <op2>]	2 _[c]
	Signed halfword	STRSH Rd, [Rn, <op2>]	2 _[c]
	Signed byte	STRSB Rd, [Rn, <op2>]	2 _[c]
	User word	STRT Rd, [Rn, #<imm>]	2 _[c]
	User halfword	STRHT Rd, [Rn, #<imm>]	2 _[c]
	User byte	STRBT Rd, [Rn, #<imm>]	2 _[c]
	User signed halfword	STRSHT Rd, [Rn, #<imm>]	2 _[c]
	User signed byte	STRSBT Rd, [Rn, #<imm>]	2 _[c]
	Doubleword	STRD Rd, Rd, [Rn, #<imm>]	1 + N
	Multiple	STM Rn, {<reglist>}	1 + N
Push	Push	PUSH {<reglist>}	1 + N
	Push with link register	PUSH {<reglist>, LR}	1 + N
Pop	Pop	POP {<reglist>}	1 + N
	Pop and return	POP {<reglist>, PC}	1 + N + P

Operation	Description	Assembler	Cycles
Semaphore	Load exclusive	LDREX Rd, [Rn, #<imm>]	2
	Load exclusive half	LDREXH Rd, [Rn]	2
	Load exclusive byte	LDREXB Rd, [Rn]	2
	Store exclusive	STREX Rd, Rt, [Rn, #<imm>]	2
	Store exclusive half	STREXH Rd, Rt, [Rn]	2
	Store exclusive byte	STREXB Rd, Rt, [Rn]	2
	Clear exclusive monitor	CLREX	1
Branch	Conditional	B<cc> <label>	1 or 1 + p[d]
	Unconditional	B <label>	1 + P
	With link	BL <label>	1 + P
	With exchange	BX Rm	1 + P
	With link and exchange	BLX Rm	1 + P
	Branch if zero	CBZ Rn, <label>	1 or 1 + p[d]
	Branch if non-zero	CBNZ Rn, <label>	1 or 1 + p[d]
	Byte table branch	TBB [Rn, Rm]	2 + P
	Halfword table branch	TBH [Rn, Rm, LSL#1]	2 + P
State change	Supervisor call	SVC #<imm>	-
	If-then-else	IT... <cond>	1[e]
	Disable interrupts	CPSID <flags>	1 or 2
	Enable interrupts	CPSIE <flags>	1 or 2
	Read special register	MRS Rd, <specreg>	1 or 2
	Write special register	MSR <specreg>, Rn	1 or 2

Operation	Description	Assembler	Cycles
	Breakpoint	BKPT #<imm>	-
Extend	Signed halfword to word	SXTH Rd, <op2>	1
	Signed byte to word	SXTB Rd, <op2>	1
	Unsigned halfword	UXTH Rd, <op2>	1
	Unsigned byte	UXTB Rd, <op2>	1
Bit field	Extract unsigned	UBFX Rd, Rn, #<imm>, #<imm>	1
	Extract signed	SBFX Rd, Rn, #<imm>, #<imm>	1
	Clear	BFC Rd, Rn, #<imm>, #<imm>	1
	Insert	BFI Rd, Rn, #<imm>, #<imm>	1
Reverse	Bytes in word	REV Rd, Rm	1
	Bytes in both halfwords	REV16 Rd, Rm	1
	Signed bottom halfword	REVSH Rd, Rm	1
	Bits in word	RBIT Rd, Rm	1
Hint	Send event	SEV	1
	Wait for event	WFE	1 + W
	Wait for interrupt	WFI	1 + W
	No operation	NOP	1
Barriers	Instruction synchronization	ISB	1 + B
	Data memory	DMB	1 + B
	Data synchronization	DSB <flags>	1 + B
<p>[a] UMULL, SMULL, UMLAL, and SMLAL instructions use early termination depending on the size of the source values. These are interruptible, that is abandoned and restarted, with worst case latency of one cycle.</p>			

Operation	Description	Assembler	Cycles
<p>[b] Division operations use early termination to minimize the number of cycles required based on the number of leading ones and zeroes in the input operands.</p> <p>[c] Neighboring load and store single instructions can pipeline their address and data phases. This enables these instructions to complete in a single execution cycle.</p> <p>[d] Conditional branch completes in a single cycle if the branch is not taken.</p> <p>[e] An <i>IT</i> instruction can be folded onto a preceding 16-bit Thumb instruction, enabling execution in zero cycles.</p>			