3.3 Instruction set summary

The processor implements the ARMv6-M Thumb instruction set, including a number of 32-bit instructions that use Thumb-2 technology. The ARMv6-M instruction set comprises:

- all of the 16-bit Thumb instructions from ARMv7-M excluding CBZ, CBNZ and IT
- the 32-bit Thumb instructions BL, DMB, DSB, ISB, MRS and MSR.

Table 3-1 shows the Cortex-M0 instructions and their cycle counts. The cycle counts are based on a system with zero wait-states.

Table 3-1 Cortex-M0 instruction summary

Operation	Description	Assembler	Cycles
Move	8-bit immediate	`MOVS Rd, #<imm>`	1
	Lo to Lo	`MOVS Rd, Rm`	1
	Any to Any	`MOV Rd, Rm`	1
	Any to PC	`MOV PC, Rm`	3
Add	3-bit immediate	`ADDS Rd, Rn, #<imm>`	1
	All registers Lo	`ADDS Rd, Rn, Rm`	1
	Any to Any	`ADD Rd, Rd, Rm`	1
	Any to PC	`ADD PC, PC, Rm`	3
	8-bit immediate	`ADDS Rd, Rd, #<imm>`	1
	With carry	`ADCS Rd, Rd, Rm`	1
	Immediate to SP	`ADD SP, SP, #<imm>`	1
	Form address from SP	`ADD Rd, SP, #<imm>`	1
	Form address from PC	`ADR Rd, <label>`	1
Subtract	Lo and Lo	`SUBS Rd, Rn, Rm`	1
	3-bit immediate	`SUBS Rd, Rn, #<imm>`	1
	8-bit immediate	`SUBS Rd, Rd, #<imm>`	1
	With carry	`SBCS Rd, Rd, Rm`	1
	Immediate from SP	`SUB SP, SP, #<imm>`	1

Table 3-1 Cortex-M0 instruction summary (continued)

Operation	Description	Assembler	Cycles
Subtract	Negate	`RSBS Rd, Rn, #0`	1
Multiply	Multiply	`MULS Rd, Rm, Rd`	1 or 32[a]
Compare	Compare	`CMP Rn, Rm`	1
	Negative	`CMN Rn, Rm`	1
	Immediate	`CMP Rn, #<imm>`	1
Logical	AND	`ANDS Rd, Rd, Rm`	1
	Exclusive OR	`EORS Rd, Rd, Rm`	1
	OR	`ORRS Rd, Rd, Rm`	1
	Bit clear	`BICS Rd, Rd, Rm`	1
	Move NOT	`MVNS Rd, Rm`	1
	AND test	`TST Rn, Rm`	1
Shift	Logical shift left by immediate	`LSLS Rd, Rm, #<shift>`	1
	Logical shift left by register	`LSLS Rd, Rd, Rs`	1
	Logical shift right by immediate	`LSRS Rd, Rm, #<shift>`	1
	Logical shift right by register	`LSRS Rd, Rd, Rs`	1
	Arithmetic shift right	`ASRS Rd, Rm, #<shift>`	1
	Arithmetic shift right by register	`ASRS Rd, Rd, Rs`	1
Rotate	Rotate right by register	`RORS Rd, Rd, Rs`	1
Load	Word, immediate offset	`LDR Rd, [Rn, #<imm>]`	2
	Halfword, immediate offset	`LDRH Rd, [Rn, #<imm>]`	2
	Byte, immediate offset	`LDRB Rd, [Rn, #<imm>]`	2
	Word, register offset	`LDR Rd, [Rn, Rm]`	2
	Halfword, register offset	`LDRH Rd, [Rn, Rm]`	2
	Signed halfword, register offset	`LDRSH Rd, [Rn, Rm]`	2
	Byte, register offset	`LDRB Rd, [Rn, Rm]`	2

Table 3-1 Cortex-M0 instruction summary (continued)

Operation	Description	Assembler	Cycles
Load	Signed byte, register offset	`LDRSB Rd, [Rn, Rm]`	2
	PC-relative	`LDR Rd, <label>`	2
	SP-relative	`LDR Rd, [SP, #<imm>]`	2
	Multiple, excluding base	`LDM Rn!, {<loreglist>}`	1+N[b]
	Multiple, including base	`LDM Rn, {<loreglist>}`	1+N[b]
Store	Word, immediate offset	`STR Rd, [Rn, #<imm>]`	2
	Halfword, immediate offset	`STRH Rd, [Rn, #<imm>]`	2
	Byte, immediate offset	`STRB Rd, [Rn, #<imm>]`	2
	Word, register offset	`STR Rd, [Rn, Rm]`	2
	Halfword, register offset	`STRH Rd, [Rn, Rm]`	2
	Byte, register offset	`STRB Rd, [Rn, Rm]`	2
	SP-relative	`STR Rd, [SP, #<imm>]`	2
	Multiple	`STM Rn!, {<loreglist>}`	1+N[b]
Push	Push	`PUSH {<loreglist>}`	1+N[b]
	Push with link register	`PUSH {<loreglist>, LR}`	1+N[b]
Pop	Pop	`POP {<loreglist>}`	1+N[b]
	Pop and return	`POP {<loreglist>, PC}`	4+N[c]
Branch	Conditional	`B<cc> <label>`	1 or 3[d]
	Unconditional	`B <label>`	3
	With link	`BL <label>`	4
	With exchange	`BX Rm`	3
	With link and exchange	`BLX Rm`	3
Extend	Signed halfword to word	`SXTH Rd, Rm`	1
	Signed byte to word	`SXTB Rd, Rm`	1
	Unsigned halfword	`UXTH Rd, Rm`	1

Table 3-1 Cortex-M0 instruction summary (continued)

Operation	Description	Assembler	Cycles
Extend	Unsigned byte	`UXTB Rd, Rm`	1
Reverse	Bytes in word	`REV Rd, Rm`	1
	Bytes in both halfwords	`REV16 Rd, Rm`	1
	Signed bottom half word	`REVSH Rd, Rm`	1
State change	Supervisor Call	`SVC #<imm>`	- [e]
	Disable interrupts	`CPSID i`	1
	Enable interrupts	`CPSIE i`	1
	Read special register	`MRS Rd, <specreg>`	4
	Write special register	`MSR <specreg>, Rn`	4
	Breakpoint	`BKPT #<imm>`	- [e]
Hint	Send event	`SEV`	1
	Wait for event	`WFE`	2[f]
	Wait for interrupt	`WFI`	2[f]
	Yield	`YIELD`[g]	1
	No operation	`NOP`	1
Barriers	Instruction synchronization	`ISB`	4
	Data memory	`DMB`	4
	Data synchronization	`DSB`	4

[a] Depends on multiplier implementation.

[b] N is the number of elements.

[c] N is the number of elements in the stack-pop list including PC and assumes load or store does not generate a HardFault exception.

[d] 3 if taken, 1 if not-taken.

[e] Cycle count depends on core and debug configuration.

[f] Excludes time spent waiting for an interrupt or event.

[g] Executes as NOP.

See the *ARMv6-M ARM* for more information about the ARMv6-M Thumb instructions