1.8 Instruction set summary

This section provides:

- a summary of the processor 16-bit instructions
- a summary of the processor 32-bit instructions.

Table 1-1 lists the 16-bit Cortex-M3 instructions.

Table 1-1 16-bit Cortex-M3 instruction summary

Operation	Assembler
Add register value and C flag to register value	`ADC <Rd>, <Rm>`
Add immediate 3-bit value to register	`ADD <Rd>, <Rn>, #<immed_3>`
Add immediate 8-bit value to register	`ADD <Rd>, #<immed_8>`
Add low register value to low register value	`ADD <Rd>, <Rn>, <Rm>`
Add high register value to low or high register value	`ADD <Rd>, <Rm>`
Add 4* (immediate 8-bit value) with PC to register	`ADD <Rd>, PC, #<immed_8> * 4`
Add 4* (immediate 8-bit value) with SP to register	`ADD <Rd>, SP, #<immed_8> * 4`
Add 4* (immediate 7-bit value) to SP	`ADD SP, #<immed_7> * 4`
Bitwise AND register values	`AND <Rd>, <Rm>`
Arithmetic shift right by immediate number	`ASR <Rd>, <Rm>, #<immed_5>`
Arithmetic shift right by number in register	`ASR <Rd>, <Rs>`
Branch conditional	`B<cond> <target address>`
Branch unconditional	`B <target_address>`
Bit clear	`BIC <Rd>, <Rm>`
Software breakpoint	`BKPT <immed_8>`
Branch with link	`BL <Rm>`
Branch with link and exchange	`BLX <Rm>`
Compare not zero and branch	`CBNZ <Rn>,<label>`
Compare zero and branch	`CBZ <Rn>,<label>`
Compare negation of register value with another register value	`CMN <Rn>, <Rm>`

Table 1-1 16-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Compare immediate 8-bit value	`CMP <Rn>, #<immed_8>`
Compare registers	`CMP <Rn>, <Rm>`
Compare high register to low or high register	`CMP <Rn>, <Rm>`
Change processor state	`CPS <effect>, <iflags>`
Copy high or low register value to another high or low register	`CPY <Rd> <Rm>`
Bitwise exclusive OR register values	`EOR <Rd>, <Rm>`

Condition the following instruction, Condition the following two instructions, Condition the following three instructions, Condition the following four instructions	`IT <cond>`
	`IT<x> <cond>`
	`IT<x><y> <cond>`
	`IT<x><y><z> <cond>`

Multiple sequential memory word loads	`LDMIA <Rn>!, <registers>`
Load memory word from base register address + 5-bit immediate offset	`LDR <Rd>, [<Rn>, #<immed_5> * 4]`
Load memory word from base register address + register offset	`LDR <Rd>, [<Rn>, <Rm>]`
Load memory word from PC address + 8-bit immediate offset	`LDR <Rd>, [PC, #<immed_8> * 4]`
Load memory word from SP address + 8-bit immediate offset	`LDR, <Rd>, [SP, #<immed_8> * 4]`
Load memory byte [7:0] from register address + 5-bit immediate offset	`LDRB <Rd>, [<Rn>, #<immed_5>]`
Load memory byte [7:0] from register address + register offset	`LDRB <Rd>, [<Rn>, <Rm>]`
Load memory halfword [15:0] from register address + 5-bit immediate offset	`LDRH <Rd>, [<Rn>, #<immed_5> * 2]`
Load halfword [15:0] from register address + register offset	`LDRH <Rd>, [<Rn>, <Rm>]`
Load signed byte [7:0] from register address + register offset	`LDRSB <Rd>, [<Rn>, <Rm>]`
Load signed halfword [15:0] from register address + register offset	`LDRSH <Rd>, [<Rn>, <Rm>]`
Logical shift left by immediate number	`LSL <Rd>, <Rm>, #<immed_5>`
Logical shift left by number in register	`LSL <Rd>, <Rs>`
Logical shift right by immediate number	`LSR <Rd>, <Rm>, #<immed_5>`
Logical shift right by number in register	`LSR <Rd>, <Rs>`
Move immediate 8-bit value to register	`MOV <Rd>, #<immed_8>`

Table 1-1 16-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Move low register value to low register	`MOV <Rd>, <Rn>`
Move high or low register value to high or low register	`MOV <Rd>, <Rm>`
Multiply register values	`MUL <Rd>, <Rm>`
Move complement of register value to register	`MVN <Rd>, <Rm>`
Negate register value and store in register	`NEG <Rd>, <Rm>`
No operation	`NOP <c>`
Bitwise logical OR register values	`ORR <Rd>, <Rm>`
Pop registers from stack	`POP <registers>`
Pop registers and PC from stack	`POP <registers, PC>`
Push registers onto stack	`PUSH <registers>`
Push LR and registers onto stack	`PUSH <registers, LR>`
Reverse bytes in word and copy to register	`REV <Rd>, <Rn>`
Reverse bytes in two halfwords and copy to register	`REV16 <Rd>, <Rn>`
Reverse bytes in low halfword [15:0], sign-extend, and copy to register	`REVSH <Rd>, <Rn>`
Rotate right by amount in register	`ROR <Rd>, <Rs>`
Subtract register value and C flag from register value	`SBC <Rd>, <Rm>`
Send event	`SEV <c>`
Store multiple register words to sequential memory locations	`STMIA <Rn>!, <registers>`
Store register word to register address + 5-bit immediate offset	`STR <Rd>, [<Rn>, #<immed_5> * 4]`
Store register word to register address	`STR <Rd>, [<Rn>, <Rm>]`
Store register word to SP address + 8-bit immediate offset	`STR <Rd>, [SP, #<immed_8> * 4]`
Store register byte [7:0] to register address + 5-bit immediate offset	`STRB <Rd>, [<Rn>, #<immed_5>]`
Store register byte [7:0] to register address	`STRB <Rd>, [<Rn>, <Rm>]`
Store register halfword [15:0] to register address + 5-bit immediate offset	`STRH <Rd>, [<Rn>, #<immed_5> * 2]`
Store register halfword [15:0] to register address + register offset	`STRH <Rd>, [<Rn>, <Rm>]`

Table 1-1 16-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Subtract immediate 3-bit value from register	`SUB <Rd>, <Rn>, #<immed_3>`
Subtract immediate 8-bit value from register value	`SUB <Rd>, #<immed_8>`
Subtract register values	`SUB <Rd>, <Rn>, <Rm>`
Subtract 4 (immediate 7-bit value) from SP	`SUB SP, #<immed_7> * 4`
Operating system service call with 8-bit immediate call code	`SVC <immed_8>`
Extract byte [7:0] from register, move to register, and sign-extend to 32 bits	`SXTB <Rd>, <Rm>`
Extract halfword [15:0] from register, move to register, and sign-extend to 32 bits	`SXTH <Rd>, <Rm>`
Test register value for set bits by ANDing it with another register value	`TST <Rn>, <Rm>`
Extract byte [7:0] from register, move to register, and zero-extend to 32 bits	`UXTB <Rd>, <Rm>`
Extract halfword [15:0] from register, move to register, and zero-extend to 32 bits	`UXTH <Rd>, <Rm>`
Wait for event	`WFE <c>`
Wait for interrupt	`WFI <c>`

Table 1-2 lists the 32-bit Cortex-M3 instructions.

Table 1-2 32-bit Cortex-M3 instruction summary

Operation	Assembler
Add register value, immediate 12-bit value, and C bit	`ADC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>`
Add register value, shifted register value, and C bit	`ADC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Add register value and immediate 12-bit value	`ADD{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Add register value and shifted register value	`ADD{S}.W <Rd>, <Rm>{, <shift>}`
Add register value and immediate 12-bit value	`ADDW.W <Rd>, <Rn>, #<immed_12>`
Bitwise AND register value with immediate 12-bit value	`AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>`
Bitwise AND register value with shifted register value	`AND{S}.W <Rd>, <Rn>, Rm>{, <shift>}`
Arithmetic shift right by number in register	`ASR{S}.W <Rd>, <Rn>, <Rm>`
Conditional branch	`B{cond}.W <label>`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Clear bit field	`BFC.W <Rd>, #<lsb>, #<width>`
Insert bit field from one register value into another	`BFI.W <Rd>, <Rn>, #<lsb>, #<width>`
Bitwise AND register value with complement of immediate 12-bit value	`BIC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Bitwise AND register value with complement of shifted register value	`BIC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Branch with link	`BL <label>`
Branch with link (immediate)	`BL<c> <label>`
Unconditional branch	`B.W <label>`
Clear exclusive clears the local record of the executing processor that an address has had a request for an exclusive access.	`CLREX <c>`
Return number of leading zeros in register value	`CLZ.W <Rd>, <Rn>`
Compare register value with two's complement of immediate 12-bit value	`CMN.W <Rn>, #<modify_constant(immed_12)>`
Compare register value with two's complement of shifted register value	`CMN.W <Rn>, <Rm>{, <shift>}`
Compare register value with immediate 12-bit value	`CMP.W <Rn>, #<modify_constant(immed_12)>`
Compare register value with shifted register value	`CMP.W <Rn>, <Rm>{, <shift>}`
Data memory barrier	`DMB <c>`
Data synchronization barrier	`DSB <c>`
Exclusive OR register value with immediate 12-bit value	`EOR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Exclusive OR register value with shifted register value	`EOR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Instruction synchronization barrier	`ISB <c>`
Load multiple memory registers, increment after or decrement before	`LDM{IA|DB}.W <Rn>{!}, <registers>`
Memory word from base register address + immediate 12-bit offset	`LDR.W <Rxf>, [<Rn>, #<offset_12>]`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Memory word to PC from register address + immediate 12-bit offset	`LDR.W PC, [<Rn>, #<offset_12>]`
Memory word to PC from base register address immediate 8-bit offset, postindexed	`LDR.W PC, [Rn], #<+/-<offset_8>`
Memory word from base register address immediate 8-bit offset, postindexed	`LDR.W <Rxf>, [<Rn>], #+/-<offset_8>`
Memory word from base register address immediate 8-bit offset, preindexed	`LDR.W <Rxf>, [<Rn>, #<+/-<offset_8>]!` `LDRT.W <Rxf>, [<Rn>, #<offset_8>]`
Memory word to PC from base register address immediate 8-bit offset, preindexed	`LDR.W PC, [<Rn>, #+/-<offset_8>]!`
Memory word from register address shifted left by 0, 1, 2, or 3 places	`LDR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Memory word to PC from register address shifted left by 0, 1, 2, or 3 places	`LDR.W PC, [<Rn>, <Rm>{, LSL #<shift>}]`
Memory word from PC address immediate 12-bit offset	`LDR.W <Rxf>, [PC, #+/-<offset_12>]`
Memory word to PC from PC address immediate 12-bit offset	`LDR.W PC, [PC, #+/-<offset_12>]`
Memory byte [7:0] from base register address + immediate 12-bit offset	`LDRB.W <Rxf>, [<Rn>, #<offset_12>]`
Memory byte [7:0] from base register address immediate 8-bit offset, postindexed	`LDRB.W <Rxf>. [<Rn>], #+/-<offset_8>`
Memory byte [7:0] from register address shifted left by 0, 1, 2, or 3 places	`LDRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Memory byte [7:0] from base register address immediate 8-bit offset, preindexed	`LDRB.W <Rxf>, [<Rn>, #<+/-<offset_8>]!`
Memory byte from PC address immediate 12-bit offset	`LDRB.W <Rxf>, [PC, #+/-<offset_12>]`
Memory doubleword from register address 8-bit offset 4, preindexed	`LDRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]{!}`
Memory doubleword from register address 8-bit offset 4, postindexed	`LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/-<offset_8> * 4`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Load register exclusive calculates an address from a base register value and an immediate offset, loads a word from memory, writes it to a register	`LDREX<c> <Rt>,[<Rn>{,#<imm>}]`
Load register exclusive halfword calculates an address from a base register value and an immediate offset, loads a halfword from memory, writes it to a register	`LDREXH<c> <Rt>,[<Rn>{,#<imm>}]`
Load register exclusive byte calculates an address from a base register value and an immediate offset, loads a byte from memory, writes it to a register	`LDREXB<c> <Rt>,[<Rn>{,#<imm>}]`
Memory halfword [15:0] from base register address + immediate 12-bit offset	`LDRH.W <Rxf>, [<Rn>, #<offset_12>]`
Memory halfword [15:0] from base register address immediate 8-bit offset, preindexed	`LDRH.W <Rxf>, [<Rn>, #<+/-<offset_8>]!`
Memory halfword [15:0] from base register address immediate 8-bit offset, postindexed	`LDRH.W <Rxf>. [<Rn>], #+/-<offset_8>`
Memory halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places	`LDRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Memory halfword from PC address immediate 12-bit offset	`LDRH.W <Rxf>, [PC, #+/-<offset_12>]`
Memory signed byte [7:0] from base register address + immediate 12-bit offset	`LDRSB.W <Rxf>, [<Rn>, #<offset_12>]`
Memory signed byte [7:0] from base register address immediate 8-bit offset, postindexed	`LDRSB.W <Rxf>. [<Rn>], #+/-<offset_8>`
Memory signed byte [7:0] from base register address immediate 8-bit offset, preindexed	`LDRSB.W <Rxf>, [<Rn>, #<+/-<offset_8>]!`
Memory signed byte [7:0] from register address shifted left by 0, 1, 2, or 3 places	`LDRSB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Memory signed byte from PC address immediate 12-bit offset	`LDRSB.W <Rxf>, [PC, #+/-<offset_12>]`
Memory signed halfword [15:0] from base register address + immediate 12-bit offset	`LDRSH.W <Rxf>, [<Rn>, #<offset_12>]`
Memory signed halfword [15:0] from base register address immediate 8-bit offset, postindexed	`LDRSH.W <Rxf>. [<Rn>], #+/-<offset_8>`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Memory signed halfword [15:0] from base register address immediate 8-bit offset, preindexed	`LDRSH.W <Rxf>, [<Rn>, #<+/-<offset_8>]!`
Memory signed halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places	`LDRSH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Memory signed halfword from PC address immediate 12-bit offset	`LDRSH.W <Rxf>, [PC, #+/-<offset_12>]`
Logical shift left register value by number in register	`LSL{S}.W <Rd>, <Rn>, <Rm>`
Logical shift right register value by number in register	`LSR{S}.W <Rd>, <Rn>, <Rm>`
Multiply two signed or unsigned register values and add the low 32 bits to a register value	`MLA.W <Rd>, <Rn>, <Rm>, <Racc>`
Multiply two signed or unsigned register values and subtract the low 32 bits from a register value	`MLS.W <Rd>, <Rn>, <Rm>, <Racc>`
Move immediate 12-bit value to register	`MOV{S}.W <Rd>, #<modify_constant(immed_12)>`
Move shifted register value to register	`MOV{S}.W <Rd>, <Rm>{, <shift>}`
Move immediate 16-bit value to top halfword [31:16] of register	`MOVT.W <Rd>, #<immed_16>`
Move immediate 16-bit value to bottom halfword [15:0] of register and clear top halfword [31:16]	`MOVW.W <Rd>, #<immed_16>`
Move to register from status	`MRS<c> <Rd>, <psr>`
Move to status register	`MSR<c> <psr>_<fields>,<Rn>`
Multiply two signed or unsigned register values	`MUL.W <Rd>, <Rn>, <Rm>`
No operation	`NOP.W`
Logical OR NOT register value with immediate 12-bit value	`ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Logical OR NOT register value with shifted register value	`ORN[S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Logical OR register value with immediate 12-bit value	`ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Logical OR register value with shifted register value	`ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Reverse bit order	`RBIT.W <Rd>, <Rm>`
Reverse bytes in word	`REV.W <Rd>, <Rm>`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Reverse bytes in each halfword	`REV16.W <Rd>, <Rn>`
Reverse bytes in bottom halfword and sign-extend	`REVSH.W <Rd>, <Rn>`
Rotate right by number in register	`ROR{S}.W <Rd>, <Rn>, <Rm>`
Rotate right with extend	`RRX{S}.W <Rd>, <Rm>`
Subtract a register value from an immediate 12-bit value	`RSB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Subtract a register value from a shifted register value	`RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Subtract immediate 12-bit value and C bit from register value	`SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Subtract shifted register value and C bit from register value	`SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Copy selected bits to register and sign-extend	`SBFX.W <Rd>, <Rn>, #<lsb>, #<width>`
Signed divide	`SDIV<c> <Rd>,<Rn>,<Rm>`
Send event	`SEV<c>`
Multiply signed words and add signed-extended value to 2-register value	`SMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>`
Multiply two signed register values	`SMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>`
Signed saturate	`SSAT.W <c> <Rd>, #<imm>, <Rn>{, <shift>}`
Multiple register words to consecutive memory locations	`STM{IA|DB}.W <Rn>{!}, <registers>`
Register word to register address + immediate 12-bit offset	`STR.W <Rxf>, [<Rn>, #<offset_12>]`
Register word to register address immediate 8-bit offset, postindexed	`STR.W <Rxf>, [<Rn>], #+/-<offset_8>`
Register word to register address shifted by 0, 1, 2, or 3 places	`STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Register word to register address immediate 8-bit offset, preindexed Store, preindexed	`STR.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}` `STRT.W <Rxf>, [<Rn>, #<offset_8>]`
Register byte [7:0] to register address immediate 8-bit offset, preindexed	`STRB{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}`
Register byte [7:0] to register address + immediate 12-bit offset	`STRB.W <Rxf>, [<Rn>, #<offset_12>]`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Register byte [7:0] to register address immediate 8-bit offset, postindexed	`STRB.W <Rxf>, [<Rn>], #+/-<offset_8>`
Register byte [7:0] to register address shifted by 0, 1, 2, or 3 places	`STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Store doubleword, preindexed	`STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]{!}`
Store doubleword, postindexed	`STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]`
Store register exclusive calculates an address from a base register value and an immediate offset, and stores a word from a register to memory if the executing processor has exclusive access to the memory addressed.	`STREX <c> <Rd>,<Rt>,[<Rn>{,#<imm>}]`
Store register exclusive byte derives an address from a base register value, and stores a byte from a register to memory if the executing processor has exclusive access to the memory addressed	`STREXB <c> <Rd>,<Rt>,[<Rn>]`
Store register exclusive halfword derives an address from a base register value, and stores a halfword from a register to memory if the executing processor has exclusive access to the memory addressed.	`STREXH <c> <Rd>,<Rt>,[<Rn>]`
Register halfword [15:0] to register address + immediate 12-bit offset	`STRH.W <Rxf>, [<Rn>, #<offset_12>]`
Register halfword [15:0] to register address shifted by 0, 1, 2, or 3 places	`STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]`
Register halfword [15:0] to register address immediate 8-bit offset, preindexed	`STRH{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}`
Register halfword [15:0] to register address immediate 8-bit offset, postindexed	`STRH.W <Rxf>, [<Rn>], #+/-<offset_8>`
Subtract immediate 12-bit value from register value	`SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>`
Subtract shifted register value from register value	`SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}`
Subtract immediate 12-bit value from register value	`SUBW.W <Rd>, <Rn>, #<immed_12>`
Sign extend byte to 32 bits	`SXTB.W <Rd>, <Rm>{, <rotation>}`
Sign extend halfword to 32 bits	`SXTH.W <Rd>, <Rm>{, <rotation>}`

Table 1-2 32-bit Cortex-M3 instruction summary (continued)

Operation	Assembler
Table branch byte	`TBB [<Rn>, <Rm>]`
Table branch halfword	`TBH [<Rn>, <Rm>, LSL #1]`
Exclusive OR register value with immediate 12-bit value	`TEQ.W <Rn>, #<modify_constant(immed_12)>`
Exclusive OR register value with shifted register value	`TEQ.W <Rn>, <Rm>{, <shift}`
Logical AND register value with 12-bit immediate value	`TST.W <Rn>, #<modify_constant(immed_12)>`
Logical AND register value with shifted register value	`TST.W <Rn>, <Rm>{, <shift>}`
Copy bit field from register value to register and zero-extend to 32 bits	`UBFX.W <Rd>, <Rn>, #<lsb>, #<width>`
Unsigned divide	`UDIV<c> <Rd>,<Rn>,<Rm>`
Multiply two unsigned register values and add to a 2-register value	`UMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>`
Multiply two unsigned register values	`UMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>`
Unsigned saturate	`USAT <c> <Rd>, #<imm>, <Rn>{, <shift>}`
Copy unsigned byte to register and zero-extend to 32 bits	`UXTB.W <Rd>, <Rm>{, <rotation>}`
Copy unsigned halfword to register and zero-extend to 32 bits	`UXTH.W <Rd>, <Rm>{, <rotation>}`
Wait for event	`WFE.W`
Wait for interrupt	`WFI.W`