## 3.3.1. Cortex-M4 instructions

The processor implements the ARMv7-M Thumb instruction set. Table 3.1 shows the Cortex-M4 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 3.1 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 3.1. Cortex-M4 instruction set summary**

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

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| | Long signed | `SMULL RdLo, RdHi, Rn, Rm` | 1 |
| | Long unsigned | `UMULL RdLo, RdHi, Rn, Rm` | 1 |
| | Long signed accumulate | `SMLAL RdLo, RdHi, Rn, Rm` | 1 |
| | Long unsigned accumulate | `UMLAL RdLo, RdHi, Rn, Rm` | 1 |
| Divide | Signed | `SDIV Rd, Rn, Rm` | 2 to 12[a] |
| | Unsigned | `UDIV Rd, Rn, Rm` | 2 to 12[a] |
| 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 |
| | 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[b] |
| | To PC | `LDR PC, [Rn, <op2>]` | 2[b] + P |
| | Halfword | `LDRH Rd, [Rn, <op2>]` | 2[b] |
| | Byte | `LDRB Rd, [Rn, <op2>]` | 2[b] |
| | Signed halfword | `LDRSH Rd, [Rn, <op2>]` | 2[b] |

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| | Signed byte | `LDRSB Rd, [Rn, <op2>]` | 2[b] |
| | User word | `LDRT Rd, [Rn, #<imm>]` | 2[b] |
| | User halfword | `LDRHT Rd, [Rn, #<imm>]` | 2[b] |
| | User byte | `LDRBT Rd, [Rn, #<imm>]` | 2[b] |
| | User signed halfword | `LDRSHT Rd, [Rn, #<imm>]` | 2[b] |
| | User signed byte | `LDRSBT Rd, [Rn, #<imm>]` | 2[b] |
| | PC relative | `LDR Rd,[PC, #<imm>]` | 2[b] |
| | 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[b] |
| | Halfword | `STRH Rd, [Rn, <op2>]` | 2[b] |
| | Byte | `STRB Rd, [Rn, <op2>]` | 2[b] |
| | Signed halfword | `STRSH Rd, [Rn, <op2>]` | 2[b] |
| | Signed byte | `STRSB Rd, [Rn, <op2>]` | 2[b] |
| | User word | `STRT Rd, [Rn, #<imm>]` | 2[b] |
| | User halfword | `STRHT Rd, [Rn, #<imm>]` | 2[b] |
| | User byte | `STRBT Rd, [Rn, #<imm>]` | 2[b] |
| | User signed halfword | `STRSHT Rd, [Rn, #<imm>]` | 2[b] |
| | User signed byte | `STRSBT Rd, [Rn, #<imm>]` | 2[b] |
| | 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 |
| 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 |

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| | Clear exclusive monitor | `CLREX` | 1 |
| Branch | Conditional | `B<cc> <label>` | 1 or 1 + P[c] |
| | 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[c] |
| | Branch if non-zero | `CBNZ Rn, <label>` | 1 or 1 + P[c] |
| | 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[d] |
| | 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 |
| | 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 |

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| | Data memory | `DMB` | 1 + B |
| | Data synchronization | `DSB <flags>` | 1 + B |

[a] 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.

[b] Neighboring load and store single instructions can pipeline their address and data phases. This enables these instructions to complete in a single execution cycle.

[c] Conditional branch completes in a single cycle if the branch is not taken.

[d] An `IT` instruction can be folded onto a preceding 16-bit Thumb instruction, enabling execution in zero cycles.

Table 3.2 shows the DSP instructions that the Cortex-M4 processor implements.

**Table 3.2. Cortex-M4 DSP instruction set summary**

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| Multiply | 32-bit multiply with 32-most-significant-bit accumulate | `SMMLA` | 1 |
| | 32-bit multiply with 32-most-significant-bit subtract | `SMMLS` | 1 |
| | 32-bit multiply returning 32-most-significant-bits | `SMMUL` | 1 |
| | 32-bit multiply with rounded 32-most-significant-bit accumulate | `SMMLAR` | 1 |
| | 32-bit multiply with rounded 32-most-significant-bit subtract | `SMMLSR` | 1 |
| | 32-bit multiply returning rounded 32-most-significant-bits | `SMMULR` | 1 |
| Signed Multiply | Q setting 16-bit signed multiply with 32-bit accumulate, bottom by bottom | `SMLABB` | 1 |
| | Q setting 16-bit signed multiply with 32-bit accumulate, bottom by top | `SMLABT` | 1 |
| | 16-bit signed multiply with 64-bit accumulate, bottom by bottom | `SMLALBB` | 1 |
| | 16-bit signed multiply with 64-bit accumulate, bottom by top | `SMLALBT` | 1 |
| | Dual 16-bit signed multiply with single 64-bit accumulator | `SMLALD` | 1 |
| | 16-bit signed multiply with 64-bit accumulate, top by bottom | `SMLALTB` | 1 |
| | 16-bit signed multiply with 64-bit accumulate, top by top | `SMLALTT` | 1 |
| | 16-bit signed multiply yielding 32-bit result, bottom by bottom | `SMULBB` | 1 |
| | 16-bit signed multiply yielding 32-bit result, bottom by top | `SMULBT` | 1 |
| | 16-bit signed multiply yielding 32-bit result, top by bottom | `SMULTB` | 1 |

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| | 16-bit signed multiply yielding 32-bit result, top by bottom | SMULTT | 1 |
| | 16-bit by 32-bit signed multiply returning 32-most-significant-bits, bottom | SMULWB | 1 |
| | 16-bit by 32-bit signed multiply returning 32-most-significant-bits, top | SMULWT | 1 |
| | Dual 16-bit signed multiply returning difference | SMUSD | 1 |
| | Q setting 16-bit signed multiply with 32-bit accumulate, top by bottom | SMLATB | 1 |
| | Q setting 16-bit signed multiply with 32-bit accumulate, top by top | SMLATT | 1 |
| | Q setting dual 16-bit signed multiply with single 32-bit accumulator | SMLAD | 1 |
| | Q setting 16-bit by 32-bit signed multiply with 32-bit accumulate, bottom | SMLAWB | 1 |
| | Q setting 16-bit by 32-bit signed multiply with 32-bit accumulate, top | SMLAWT | 1 |
| | Q setting dual 16-bit signed multiply subtract with 32-bit accumulate | SMLSD | 1 |
| | Q setting dual 16-bit signed multiply subtract with 64-bit accumulate | SMLSLD | 1 |
| | Q setting sum of dual 16-bit signed multiply | SMUAD | 1 |
| | Q setting pre-exchanged dual 16-bit signed multiply with single 32-bit accumulator | SMLADX | 1 |
| Unsigned Multiply | 32-bit unsigned multiply with double 32-bit accumulation yielding 64-bit result | UMAAL | 1 |
| Saturate | Q setting dual 16-bit saturate | SSAT16 | 1 |
| | Q setting dual 16-bit unsigned saturate | USAT16 | 1 |
| Packing and Unpacking | Pack half word top with shifted bottom | PKHTB | 1 |
| | Pack half word bottom with shifted top | PKHBT | 1 |
| | Extract 8-bits and sign extend to 32-bits | SXTB | 1 |
| | Dual extract 8-bits and sign extend each to 16-bits | SXTB16 | 1 |
| | Extract 16-bits and sign extend to 32-bits | SXTH | 1 |
| | Extract 8-bits and zero-extend to 32-bits | UXTB | 1 |
| | Dual extract 8-bits and zero-extend to 16-bits | UXTB16 | 1 |
| | Extract 16-bits and zero-extend to 32-bits | UXTH | 1 |
| | Extract 8-bit to 32-bit unsigned addition | UXTAB | 1 |
| | Dual extracted 8-bit to 16-bit unsigned addition | UXTAB16 | 1 |
| | Extracted 16-bit to 32-bit unsigned addition | UXTAH | 1 |
| | Extracted 8-bit to 32-bit signed addition | SXTAB | 1 |

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| | Dual extracted 8-bit to 16-bit signed addition | SXTAB16 | 1 |
| | Extracted 16-bit to 32-bit signed addition | SXTAH | 1 |
| Miscellaneous Data Processing | Select bytes based on GE bits | SEL | 1 |
| | Unsigned sum of quad 8-bit unsigned absolute difference | USAD8 | 1 |
| | Unsigned sum of quad 8-bit unsigned absolute difference with 32-bit accumulate | USADA8 | 1 |
| Addition | Dual 16-bit unsigned saturating addition | UQADD16 | 1 |
| | Quad 8-bit unsigned saturating addition | UQADD8 | 1 |
| | Q setting saturating add | QADD | 1 |
| | Q setting dual 16-bit saturating add | QADD16 | 1 |
| | Q setting quad 8-bit saturating add | QADD8 | 1 |
| | Q setting saturating double and add | QDADD | 1 |
| | GE setting quad 8-bit signed addition | SADD8 | 1 |
| | GE setting dual 16-bit signed addition | SADD16 | 1 |
| | Dual 16-bit signed addition with halved results | SHADD16 | 1 |
| | Quad 8-bit signed addition with halved results | SHADD8 | 1 |
| | GE setting dual 16-bit unsigned addition | UADD16 | 1 |
| | GE setting quad 8-bit unsigned addition | UADD8 | 1 |
| | Dual 16-bit unsigned addition with halved results | UHADD16 | 1 |
| | Quad 8-bit unsigned addition with halved results | UHADD8 | 1 |
| Subtraction | Q setting saturating double and subtract | QDSUB | 1 |
| | Dual 16-bit unsigned saturating subtraction | UQSUB16 | 1 |
| | Quad 8-bit unsigned saturating subtraction | UQSUB8 | 1 |
| | Q setting saturating subtract | QSUB | 1 |
| | Q setting dual 16-bit saturating subtract | QSUB16 | 1 |
| | Q setting quad 8-bit saturating subtract | QSUB8 | 1 |
| | Dual 16-bit signed subtraction with halved results | SHSUB16 | 1 |
| | Quad 8-bit signed subtraction with halved results | SHSUB8 | 1 |
| | GE setting dual 16-bit signed subtraction | SSUB16 | 1 |
| | GE setting quad 8-bit signed subtraction | SSUB8 | 1 |
| | Dual 16-bit unsigned subtraction with halved results | UHSUB16 | 1 |
| | Quad 8-bit unsigned subtraction with halved results | UHSUB8 | 1 |
| | GE setting dual 16-bit unsigned subtract | USUB16 | 1 |
| | GE setting quad 8-bit unsigned subtract | USUB8 | 1 |

| Operation | Description | Assembler | Cycles |
|---|---|---|---|
| Parallel Addition and Subtraction | Dual 16-bit unsigned saturating addition and subtraction with exchange | UQASX | 1 |
| | Dual 16-bit unsigned saturating subtraction and addition with exchange | UQSAX | 1 |
| | GE setting dual 16-bit addition and subtraction with exchange | SASX | 1 |
| | Q setting dual 16-bit add and subtract with exchange | QASX | 1 |
| | Q setting dual 16-bit subtract and add with exchange | QSAX | 1 |
| | Dual 16-bit signed addition and subtraction with halved results | SHASX | 1 |
| | Dual 16-bit signed subtraction and addition with halved results | SHSAX | 1 |
| | GE setting dual 16-bit signed subtraction and addition with exchange | SSAX | 1 |
| | GE setting dual 16-bit unsigned addition and subtraction with exchange | UASX | 1 |
| | Dual 16-bit unsigned addition and subtraction with halved results and exchange | UHASX | 1 |
| | Dual 16-bit unsigned subtraction and addition with halved results and exchange | UHSAX | 1 |
| | GE setting dual 16-bit unsigned subtract and add with exchange | USAX | 1 |