Appendix D: Cortex-M0/M0+/M1 Instructions

Instruction	Operands	Description and Action
ADCS	{Rd,} Rn, Rm	Add with Carry, Rd ← Rn + Rm + Carry, update N,Z,C,V
ADD, ADDS	{Rd,} Rn, <Rm|#imm>	Add, Rd ← Rn + <Rm|#imm>, ADDS updates N,Z,C,V
ADR	Rd, label	Load PC-relative Address, Rd ← <label>
AND, ANDS	{Rd,} Rn, Rm	Logical AND, Rd ← Rn AND Rm, ANDS updates N,Z,C
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic Shift Right, Rd ← Rm>>(Rs|n), ASRS updates N,Z,C
B{cc}	label	Branch {conditionally}, PC ← label
BICS	{Rd,} Rn, Rm	Bit Clear, Rd ← Rn AND NOT Rm, BICS updates N,Z,C
BKPT	#imm	Breakpoint, prefetch abort or enter debug state
BL	label	Branch with Link, LR ← next instruction, PC ← label
BLX	Rm	Branch register with link, LR←next instr addr, PC←Rm[31:1]
BX	Rm	Branch register, PC ← Rm
CMN	Rn, Rm	Compare Negative, Update N,Z,C,V flags on Rn + Rm
CMP	Rn, <Rm|#imm>	Compare, Update N,Z,C,V flags on Rn – <Rm|#imm>
CPSID	i	Disable specified (i) interrupts, optional change mode
CPSIE	i	Enable specified (i) interrupts, optional change mode
DMB	-	Data Memory Barrier, ensure memory access order
DSB	-	Data Synchronization Barrier, ensure completion of access
EORS	{Rd,} Rn, Rm	Exclusive OR, Rd ← Rn XOR Rm, EORS updates N,Z,C
ISB	-	Instruction Synchronization Barrier
LDM	Rn{!}, reglist	Load Multiple Registers increment after, <reglist> = mem[Rn], Rn increments after each memory access
LDR	Rt, [Rn, <Rm|#imm>]	Load Register with Word, Rt ← mem[Rn + <Rm|#imm>]
LDRB	Rt, [Rn, <Rm|#imm>]	Load Register with Byte, Rt ← mem[Rn + <Rm|#imm>]
LDRH	Rt, [Rn, <Rm|#imm>]	Load Register with Halfword, Rt ← mem[Rn + <Rm|#imm>]
LDRSB	Rt, [Rn, <Rm|#imm>]	Load Register with Signed Byte, Rt ← mem[Rn + <Rm|#imm>]
LDRSH	Rt, [Rn, <Rm|#imm>]	Load Register with Signed Halfword, Rt ← mem[Rn+<Rm|#imm>]
LSLS	Rd, Rm, <Rs|#imm>	Logic Shift Left, Rd ← Rm << Rs|#imm, LSLS update N,Z,C
LSRS	Rd, Rm, <Rs|#imm>	Logic Shift Right, Rd ← Rm >> Rs|#imm, LSRS update N,Z,C
MOV, MOVS	Rd, <Rs|#imm>	Move, Rd ← <Rs|#imm>, MOVS updates N,Z,C
MRS	Rd, spec_reg	Move from Special Register, Rd ← spec_reg
MSR	spec_reg, Rm	Move to Special Register, spec_reg ← Rm, Updates N,Z,C,V
MULS	{Rd,} Rn, Rm	Multiply, Rd ← (Rn*Rm)[31:0], MULS updates N,Z
MVNS	Rd, Rm	Move NOT, Rd ← 0xFFFFFFFF EOR Rm, MVNS updates N,Z,C
NOP	-	No Operation
ORRS	{Rd,} Rn, Rm	Logical OR, Rd ← Rn OR Rm, ORRS updates N,Z,C
POP	reglist	Canonical form of LDM SP!, <reglist>
PUSH	reglist	Canonical form of STMDB SP!, <reglist>
REV	Rd, Rn	Reverse Byte Order in a Word, Rd[31:24]←Rn[7:0], Rd[23:16]←Rn[15:8], Rd[15:8]←Rn[23:16], Rd[7:0]←Rn[31:24]
REV16	Rd, Rn	Reverse Byte Order in a Half-word, Rd[15:8]←Rn[7:0], Rd[7:0]←Rn[15:8], Rd[31:24]←Rn[23:16], Rd[23:16]←Rn[31:24]
REVSH	Rd, Rn	Reverse Byte order in Low Half-word and sign extend, Rd[15:8]←Rn[7:0], Rd[7:0]←Rn[15:8], Rd[31:16]←Rn[7]*&0xFFFF
RORS	{Rd,} Rm, Rs	Rotate Right, Rd ← ROR(Rm, Rs), RORS updates N,Z,C
RSBS	{Rd,} Rn, #0	Reverse Subtract, Rd ← 0 - Rn, RSBS updates N,Z,C,V
SBCS	{Rd,} Rn, Rm	Subtract with Carry, Rd← Rn – Rm – NOT(Carry), updates NZCV
SEV	-	Send Event
STM	Rn{!}, reglist	Store Multiple Registers
STR	Rt, [Rn, <Rm|#imm>]	Store Register with Word, mem[Rn + <Rm|#imm>] = Rt
STRB	Rt, [Rn, <Rm|#imm>]	Store Register with Byte, mem[Rn + <Rm|#imm>] = Rt
STRH	Rt, [Rn, <Rm|#imm>]	Store Half-word, mem[Rn + <Rm|#imm>] ← Rt[15:0]
SUB, SUBS	{Rd,} Rn, <Rm|#imm>	Subtraction, Rd ← Rn – <Rm|#imm>, SUBS updates N,Z,C,V

SVC	#imm	Supervisor Call
SXTB	{Rd,} Rm	Sign Extend Byte, Rd ← SignExtend(Rm[7:0])
SXTH	{Rd,} Rm	Sign Extend Half-word, Rd ← SignExtend(Rm[15:0])
TST	Rn, Rm	Test, Update N,Z,C,V on Rn AND Rm
UXTB	{Rd,} Rm	Unsigned Extend Byte, Rd ← ZeroExtend(Rm[7:0])
UXTH	{Rd,} Rm	Unsigned Extend Halfword, Rd ← ZeroExtend(Rm[15:0])
WFE	-	Wait For Event and Enter Sleep Mode
WFI	-	Wait for Interrupt and Enter Sleep Mode