Appendix D: DSP Instructions on Cortex-M4 and Cortex-M7

T = Top/high halfword, B = Bottom/low halfword
SQ = Signed saturation, UQ = Unsigned saturation

Instruction	Operands	Description and Action
PKHBT	{Rd,} Rn, Rm, Op2	Pack halfword. Rd = Rn[B]:(Rm, Op2)[T]
PKHTB	{Rd,} Rn, Rm, Op2	Pack halfword. Rd = Rn[T]:(Rm, Op2)[B]
QADD	{Rd,} Rn, Rm	Saturating add signed 32-bit integers Rd = SQ32(Rn + Rm)
QADD16	{Rd,} Rn, Rm	Saturating add 2 pairs of 16-bit signed integers Rd[T] = SQ16(Rn[T] + Rm[T]) Rd[B] = SQ16(Rn[B] + Rm[B])
QADD8	{Rd,} Rn, Rm	Saturating add 4 pairs of 8-bit signed integers Rd[31:24] = Rn[31:24] + Rm[31:24] Rd[25:16] = Rn[25:16] + Rm[25:16] Rd[15:8] = Rn[15:8] + Rm[15:8] Rd[7:0] = Rn[7:0] + Rm[7:0]
QASX	{Rd,} Rn, Rm	Saturating add and subtract with exchange Rd[T] = SQ16(Rn[T] + Rm[B]) Rd[B] = SQ16(Rn[B] - Rm[T])
QDADD	{Rd,} Rn, Rm	Saturating double and add Rd = SQ32(Rn + SQ32(Rm *2))
QDSUB	{Rd,} Rn, Rm	Saturating double and subtract Rd = SQ32(Rn – SQ32(2*Rm))
QSAX	{Rd,} Rn, Rm	Saturating subtract and add with exchange Rd[T]=SQ16(Rn[T]-Rm[B]), Rd[B]=SQ16(Rn[B]+Rm[T])
QSUB	{Rd,} Rn, Rm	Signed saturating subtract two 32-bit signed integers Rd = SQ32(Rn – Rm)
QSUB16	{Rd,} Rn, Rm	Signed saturating subtract 2 pairs of 16-bit signed integers, Rd[T]=SQ16(Rn[T]-Rm[T]), Rd[B]=SQ16(Rn[B]-Rm[B])
QSUB8	{Rd,} Rn, Rm	Signed saturating subtract 4 pairs of 8-bit signed integers
SADD16	{Rd,} Rn, Rm	Signed add 2 pairs of 16-bit integers Rd[T] = truncate16(Rn[T] + Rm[T]) Rd[B] = truncate16(Rn[B] + Rm[B])
SADD8	{Rd,} Rn, Rm	Signed add 4 pairs of 8-bit signed integers
SASX	{Rd,} Rn, Rm	Signed add and subtract with exchange Rd[T] = truncate16(Rn[T] + Rm[B]) Rd[B] = truncate16(Rn[B] - Rm[T])
SEL	{Rd,} Rn, Rm	Select bytes based on GE bits of CPSR
SHADD16	{Rd,} Rn, Rm	Signed halving add 2 pairs of 16-bit integers Rd[T] = (Rn[T] + Rm[T])/2, Rd[B] = (Rn[B] + Rm[B])/2
SHADD8	{Rd,} Rn, Rm	Signed halving add 4 pairs of 8-bit integers
SHASX	{Rd,} Rn, Rm	Signed halving add and subtract with exchange Rd[T] = (Rn[T] + Rm[B])/2, Rd[B] = (Rn[B] - Rm[T])/2
SHSAX	{Rd,} Rn, Rm	Signed halving subtract and add with exchange Rd[T] = (Rn[T] - Rm[B])/2, Rd[B] = (Rn[B] + Rm[T])/2
SHSUB16	{Rd,} Rn, Rm	Signed halving subtract 2 pairs of 16-bit integers Rd[T] = (Rn[T] - Rm[T])/2, Rd[B] = (Rn[B] - Rm[B])/2
SHSUB8	{Rd,} Rn, Rm	Signed halving subtract 4 pairs of 8-bit integers
SMLABB, SMLABT, SMLATB, SMLATT	Rd, Rn, Rm, Ra	Signed multiply accumulate long (halfwords) Rd = Ra + Rn[B/T]*Rm[B/T] e.g. BT, Rd = Ra + Rn[B]*Rm[T]
SMLALBB, SMLALBT, SMLATLB, SMLALTT	RdLo, RdHi, Rn, Rm	Signed multiply accumulate long (halfwords) RdHi:RdLo = RdHi:RdLo + Rn[B/T]*Rm[B/T] e.g. BT, RdHi:RdLo = RdHi:RdLo + Rn[B]*Rm[T]

SMLAD	Rd, Rn, Rm, Ra	Signed multiply accumulate dual Rd = Ra + Rn[T]*Rm[T] + Rn[B]*Rm[B]
SMLADX	Rd, Rn, Rm, Ra	Signed multiply accumulate dual with exchange Rd = Ra + Rn[T]*Rm[B] + Rn[B]*Rm[T]
SMLALD	RdLo, RdHi, Rn, Rm	Signed multiply accumulate long dual RdHi:RdLo = RdHi:RdLo + Rn[T]*Rm[T] + Rn[B]*Rm[B]
SMLALDX	RdLo, RdHi, Rn, Rm	Signed multiply accumulate long dual with exchange RdHi:RdLo = RdHi:RdLo + Rn[T]*Rm[B] + Rn[B]*Rm[T]
SMLAWB	Rd, Rn, Rm, Ra	Signed multiply accumulate (word by bottom halfword), Rd = Ra + (Rn*Rm[B])>>16
SMLAWT	Rd, Rn, Rm, Ra	Signed multiply accumulate (word by top halfword), Rd = Ra + (Rn*Rm[T])>>16
SMLSD	Rd, Rn, Rm, Ra	Signed multiply subtract dual Rd = Ra + Rn[B]*Rm[B] - Rn[T]* Rm[T]
SMLSDX	Rd, Rn, Rm, Ra	Signed multiply subtract dual with exchange Rd = Ra + Rn[B]*Rm[T] - Rn[T]* Rm[B]
SMLSLD	RdLo, RdHi, Rn, Rm	Signed multiply subtract long dual RdHi:RdLo = RdHi:RdLo + Rn[T]* Rm[T] - Rn[B]*Rm[B]
SMLSLDX	RdLo, RdHi, Rn, Rm	Signed multiply subtract long dual with exchange RdHi:RdLo = RdHi:RdLo + Rn[B]* Rm[T] - Rn[T]*Rm[B]
SMMLA, SMMLAR	Rd, Rn, Rm, Ra	Signed most significant word multiply accumulate, Rd = Ra + (Rn*Rm)>>32. If R exists, round to nearest; otherwise, truncate.
SMMLS, SMMLSR	Rd, Rn, Rm, Ra	Signed most significant word multiply subtract, Rd = Ra – (Rn*Rm)>>32. See above for R.
SMMUL, SMMULR	{Rd,} Rn, Rm	Signed most significant word multiply Rd = (Rn*Rm)>>32. See above for R.
SMULBB, SMULBT SMULTB, SMULTT	{Rd,} Rn, Rm	Signed multiply (halfwords), Rd = Rn[B/T]*Rm[B/T] e.g. BT, Rd = Rn[B]*Rm[T]
SMUAD	{Rd,} Rn, Rm	Signed dual multiply then add Rd = Rn[B]*Rm[B] + Rn[T]*Rm[T]
SMUADX	{Rd,} Rn, Rm	Signed dual multiply add with exchange Rd = Rn[T]*Rm[B] + Rn[B]*Rm[T]
SMULWB	{Rd,} Rn, Rm	Signed multiply word by bottom halfword Rd = (Rn*Rm[B])>>16
SMULWT	{Rd,} Rn, Rm	Signed multiply word by top halfword Rd = (Rn*Rm[T])>>16
SMUSD	{Rd,} Rn, Rm	Signed dual multiply then subtract Rd = Rn[B]*Rm[B] – Rn[T]*Rm[T]
SMUSDX	{Rd,} Rn, Rm	Signed dual multiply (with exchange) subtract Rd = Rn[B]*Rm[T] – Rn[T]*Rm[B]
SSAT16	Rd, #imm4, Rm	Signed saturate two 16-bit values #imm4 = saturation bit position, $-2^{imm4 - 1} \le x \le 2^{imm4 - 1} -1$
SSAX	{Rd,} Rn, Rm	Signed subtract and add with exchange Rd[T] = truncate16(Rn[T] - Rm[B]) Rd[B] = truncate16(Rn[B] + Rm[T])
SSUB16	{Rd,} Rn, Rm	Signed subtract 2 pairs of 16-bit integers Rd[T] = truncate16(Rn[T] - Rm[T]) Rd[B] = truncate16(Rn[B] - Rm[B])
SSUB8	{Rd,} Rn, Rm	Signed subtract 4 pairs of 8-bit integers
SXTAB	{Rd,} Rn, Rm{,ROR #}	Extend 8 bits to 32 bits and add Rd = Rn + sign_extend ((Rm, ROR #)[7:0])
SXTAB16	{Rd,} Rn, Rm{,ROR #}	Dual extend 8 bits to 16 bits and add Rd[T] = Rn[T] + sign_extend ((Rm, ROR #)[23:16]) Rd[B] = Rn[B] + sign_extend ((Rm, ROR #)[7:0])
SXTAH	{Rd,} Rn, Rm{,ROR #}	Extend 16 bits to 32 and add Rd = Rn + sign_extend ((Rm, ROR #)[15:0])
SXTB16	{Rd,} Rm {,ROR #n}	Signed extend byte to 16-bit value Rd[T] = sign_extend ((Rm, ROR #)[23:16]) Rd[B] = sign_extend ((Rm, ROR #)[7:0])

UADD16	{Rd,} Rn, Rm	Unsigned add 2 pairs of 16-bit integers Rd[T] = truncate16(Rn[T] + Rm[T]) Rd[B] = truncate16(Rn[B] + Rm[B])
UADD8	{Rd,} Rn, Rm	Unsigned add 4 pairs of 8-bit integers
UASX	{Rd,} Rn, Rm	Unsigned add and subtract with exchange Rd[T] = truncate16(Rn[T] + Rm[B]) Rd[B] = truncate16(Rn[B] - Rm[T])
UHADD16	{Rd,} Rn, Rm	Unsigned halving add 2 pairs of 16-bit integers Rd[T] = (Rn[T] + Rm[T])/2, Rd[B] = (Rn[B] + Rm[B])/2
UHADD8	{Rd,} Rn, Rm	Unsigned halving add 4 pairs of 8-bit integers
UHASX	{Rd,} Rn, Rm	Unsigned halving add and subtract with exchange Rd[T] = (Rn[T] + Rm[B])/2, Rd[B] = (Rn[B] - Rm[T])/2
UHSAX	{Rd,} Rn, Rm	Unsigned halving subtract and add with exchange Rd[T] = (Rn[T] - Rm[B])/2, Rd[B] = (Rn[B] + Rm[T])/2
UHSUB16	{Rd,} Rn, Rm	Unsigned halving subtract 2 pairs of 16-bit integers Rd[T] = (Rn[T] – Rm[T])/2, Rd[B] = (Rn[B] – Rm[B])/2
UHSUB8	{Rd,} Rn, Rm	Unsigned halving subtract 4 pairs of 8-bit integers
UMAAL	RdLo, RdHi, Rn, Rm	Unsigned multiply accumulate long RdHi:RdLo = Rn*Rm + RdHi + RdLo
UQADD16	{Rd,} Rn, Rm	Unsigned saturating add 2 pairs of 16-bit integers Rd[T] = UQ(Rn[T] + Rm[T]), Rd[B] = UQ(Rn[B] + Rm[B])
UQADD8	{Rd,} Rn, Rm	Unsigned saturating add 4 pairs of 8-bit integers
UQASX	{Rd,} Rn, Rm	Unsigned saturating add and subtract with exchange Rd[T] = saturate16(Rn[T] + Rm[B]) Rd[B] = saturate16(Rn[B] - Rm[T])
UQSAX	{Rd,} Rn, Rm	Unsigned saturating subtract and add with exchange Rd[T] = saturate16(Rn[T] – Rm[B]) Rd[B] = saturate16(Rn[B] + Rm[T])
UQSUB16	{Rd,} Rn, Rm	Unsigned saturating subtract 2 pairs of 16-bit integers Rd[T] = UQ(Rn[T] - Rm[T]), Rd[B] = UQ(Rn[B] - Rm[B])
UQSUB8	{Rd,} Rn, Rm	Unsigned saturating subtract 4 pairs of 8-bit integers
USAD8	{Rd,} Rn, Rm	Unsigned sum of absolute differences
USADA8	{Rd,} Rn, Rm, Ra	Unsigned sum of absolute differences and accumulate
USAT16	Rd, #imm4, Rm	Unsigned saturate two 16-bit integers #imm4 = saturation bit position, $0 \le x \le 2^{imm4} - 1$
USAX	{Rd,} Rn, Rm	Unsigned subtract and add with exchange Rd[T] = truncate16(Rn[T] - Rm[B]) Rd[B] = truncate16(Rn[B] + Rm[T])
USUB16	{Rd,} Rn, Rm	Unsigned subtract 2 pairs of 16-bit integers Rd[T] = truncate16(Rn[T] - Rm[T]) Rd[B] = truncate16(Rn[B] - Rm[B])
USUB8	{Rd,} Rn, Rm	Unsigned subtract 4 pairs of 8-bit integers
UXTAB	{Rd,} Rn, Rm{, ROR #}	Rotate, extend 8 bits to 32 bits and Add Rd = Rn + zero_extend ((Rm, ROR #)[7:0])
UXTAB16	{Rd,} Rn, Rm{, ROR #}	Rotate, dual extend 8 bits to 16 bits and add Rd[T] = Rn[T] + zero_extend ((Rn, ROR #)[23:16]) Rd[B] = Rn[B] + zero_extend ((Rn, ROR #)[7:0])
UXTAH	{Rd,} Rn, Rm{, ROR #}	Rotate, unsigned extend and add halfword Rd = Rn + zero_extend ((Rm, ROR #)[15:0])
UXTB16	{Rd,} Rm{, ROR #n}	Unsigned extend byte to 16-bit value Rd[T] = zero_extend ((Rm, ROR #)[23:16]) Rd[B] = zero_extend ((Rm, ROR #)[7:0])