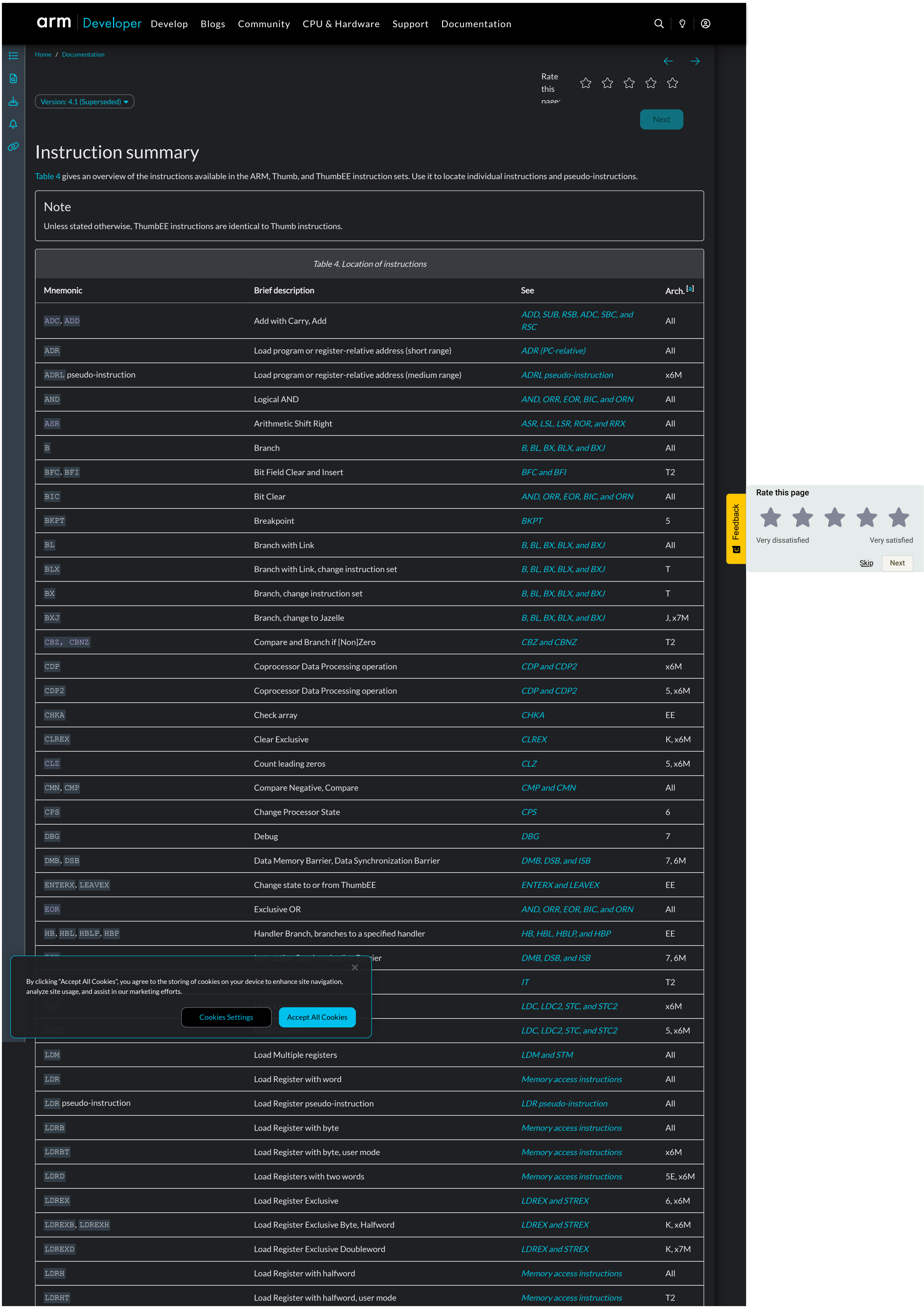

arm Developer Develop Blogs Community CPU & Hardware Support Documentation

Version: 4.1 (Superseded)

Instruction summary

Table 4 gives an overview of the instructions available in the ARM, Thumb, and ThumbEE instruction sets. Use it to locate individual instructions and pseudo-instructions.

Note

Unless stated otherwise, ThumbEE instructions are identical to Thumb instructions.

Table 4. Location of instructions

Mnemonic	Brief description	See	Arch. [a]
`ADC`, `ADD`	Add with Carry, Add	*ADD, SUB, RSB, ADC, SBC, and RSC*	All
`ADR`	Load program or register-relative address (short range)	*ADR (PC-relative)*	All
`ADRL` pseudo-instruction	Load program or register-relative address (medium range)	*ADRL pseudo-instruction*	x6M
`AND`	Logical AND	*AND, ORR, EOR, BIC, and ORN*	All
`ASR`	Arithmetic Shift Right	*ASR, LSL, LSR, ROR, and RRX*	All
`B`	Branch	*B, BL, BX, BLX, and BXJ*	All
`BFC`, `BFI`	Bit Field Clear and Insert	*BFC and BFI*	T2
`BIC`	Bit Clear	*AND, ORR, EOR, BIC, and ORN*	All
`BKPT`	Breakpoint	*BKPT*	5
`BL`	Branch with Link	*B, BL, BX, BLX, and BXJ*	All
`BLX`	Branch with Link, change instruction set	*B, BL, BX, BLX, and BXJ*	T
`BX`	Branch, change instruction set	*B, BL, BX, BLX, and BXJ*	T
`BXJ`	Branch, change to Jazelle	*B, BL, BX, BLX, and BXJ*	J, x7M
`CBZ, CBNZ`	Compare and Branch if {Non}Zero	*CBZ and CBNZ*	T2
`CDP`	Coprocessor Data Processing operation	*CDP and CDP2*	x6M
`CDP2`	Coprocessor Data Processing operation	*CDP and CDP2*	5, x6M
`CHKA`	Check array	*CHKA*	EE
`CLREX`	Clear Exclusive	*CLREX*	K, x6M
`CLZ`	Count leading zeros	*CLZ*	5, x6M
`CMN`, `CMP`	Compare Negative, Compare	*CMP and CMN*	All
`CPS`	Change Processor State	*CPS*	6
`DBG`	Debug	*DBG*	7
`DMB`, `DSB`	Data Memory Barrier, Data Synchronization Barrier	*DMB, DSB, and ISB*	7, 6M
`ENTERX`, `LEAVEX`	Change state to or from ThumbEE	*ENTERX and LEAVEX*	EE
`EOR`	Exclusive OR	*AND, ORR, EOR, BIC, and ORN*	All
`HB`, `HBL`, `HBLP`, `HBP`	Handler Branch, branches to a specified handler	*HB, HBL, HBLP, and HBP*	EE
`ISB`	Instruction Synchronization Barrier	*DMB, DSB, and ISB*	7, 6M
[illegible]	[illegible]	*IT*	T2
`LDC`	Load Coprocessor	*LDC, LDC2, STC, and STC2*	x6M
`LDC2`	Load Coprocessor	*LDC, LDC2, STC, and STC2*	5, x6M
`LDM`	Load Multiple registers	*LDM and STM*	All
`LDR`	Load Register with word	*Memory access instructions*	All
`LDR` pseudo-instruction	Load Register pseudo-instruction	*LDR pseudo-instruction*	All
`LDRB`	Load Register with byte	*Memory access instructions*	All
`LDRBT`	Load Register with byte, user mode	*Memory access instructions*	x6M
`LDRD`	Load Registers with two words	*Memory access instructions*	5E, x6M
`LDREX`	Load Register Exclusive	*LDREX and STREX*	6, x6M
`LDREXB`, `LDREXH`	Load Register Exclusive Byte, Halfword	*LDREX and STREX*	K, x6M
`LDREXD`	Load Register Exclusive Doubleword	*LDREX and STREX*	K, x7M
`LDRH`	Load Register with halfword	*Memory access instructions*	All
`LDRHT`	Load Register with halfword, user mode	*Memory access instructions*	T2

Table 4. Location of instructions

Mnemonic	Brief description	See	Arch. [a]
`LDRSB`	Load Register with signed byte	*Memory access instructions*	All
`LDRSBT`	Load Register with signed byte, user mode	*Memory access instructions*	T2
`LDRSH`	Load Register with signed halfword	*Memory access instructions*	All
`LDRSHT`	Load Register with signed halfword, user mode	*Memory access instructions*	T2
`LDRT`	Load Register with word, user mode	*Memory access instructions*	x6M
`LSL`, `LSR`	Logical Shift Left, Logical Shift Right	*ASR, LSL, LSR, ROR, and RRX*	All
`MAR`	Move from Registers to 40-bit Accumulator	*MAR and MRA*	XScale
`MCR`	Move from Register to Coprocessor	*MCR, MCR2, MCRR, and MCRR2*	x6M
`MCR2`	Move from Register to Coprocessor	*MCR, MCR2, MCRR, and MCRR2*	5, x6M
`MCRR`	Move from Registers to Coprocessor	*MCR, MCR2, MCRR, and MCRR2*	5E, x6M
`MCRR2`	Move from Registers to Coprocessor	*MCR, MCR2, MCRR, and MCRR2*	6, x6M
`MIA`, `MIAPH`, `MIAxy`	Multiply with Internal 40-bit Accumulate	*MIA, MIAPH, and MIAxy*	XScale
`MLA`	Multiply Accumulate	*MUL, MLA, and MLS*	x6M
`MLS`	Multiply and Subtract	*MUL, MLA, and MLS*	T2
`MOV`	Move	*MOV and MVN*	All
`MOVT`	Move Top	*MOVT*	T2
`MOV32` pseudo-instruction	Move 32-bit immediate to register	*MOV32 pseudo--instruction*	T2
`MRA`	Move from 40-bit Accumulator to Registers	*MAR and MRA*	XScale
`MRC`	Move from Coprocessor to Register	*MRC, MRC2, MRRC and MRRC2*	x6M
`MRC2`	Move from Coprocessor to Register	*MRC, MRC2, MRRC and MRRC2*	5, x6M
`MRRC`	Move from Coprocessor to Registers	*MRC, MRC2, MRRC and MRRC2*	5E, x6M
`MRRC2`	Move from Coprocessor to Registers	*MRC, MRC2, MRRC and MRRC2*	6, x6M
`MRS`	Move from PSR to register	*MRS*	All
`MRS`	Move from system Coprocessor to Register	*MRS*	7A, 7R
`MSR`	Move from register to PSR	*MSR*	All
`MSR`	Move from Register to system Coprocessor	*MSR*	7A, 7R
`MUL`	Multiply	*MUL, MLA, and MLS*	All
`MVN`	Move Not	*MOV and MVN*	All
`NOP`	No Operation	*NOP*	All
`ORN`	Logical OR NOT	*AND, ORR, EOR, BIC, and ORN*	T2
`ORR`	Logical OR	*AND, ORR, EOR, BIC, and ORN*	All
`PKHBT`, `PKHTB`	Pack Halfwords	*PKHBT and PKHTB*	6, 7EM
`PLD`	Preload Data	*PLD, PLDW, and PLI*	5E, x6M
`PLDW`	Preload Data with intent to Write	*PLD, PLDW, and PLI*	7MP
`PLI`	Preload Instruction	*PLD, PLDW, and PLI*	7
`PUSH`, `POP`	PUSH registers to stack, POP registers from stack	*PUSH and POP*	All
`QADD`, `QDADD`, `QDSUB`, `QSUB`	Saturating Arithmetic	*QADD, QSUB, QDADD, and QDSUB*	5E, 7EM
`QADD8`, `QADD16`, `QASX`, `QSUB8`, `QSUB16`, `QSAX`	Parallel signed Saturating Arithmetic	*Parallel add and subtract*	6, 7EM
`RBIT`	Reverse Bits	*REV, REV16, REVSH, and RBIT*	T2
`REV`, `REV16`, `REVSH`	Reverse byte order	*REV, REV16, REVSH, and RBIT*	6
`RFE`	Return From Exception	*RFE*	T2, x7M
`ROR`	Rotate Right Register	*ASR, LSL, LSR, ROR, and RRX*	All
`RRX`	Rotate Right with Extend	*ASR, LSL, LSR, ROR, and RRX*	x6M
`RSB`	Reverse Subtract	*ADD, SUB, RSB, ADC, SBC, and RSC*	All
`RSC`	Reverse Subtract with Carry	*ADD, SUB, RSB, ADC, SBC, and RSC*	x7M
`SADD8`, `SADD16`, `SASX`	Parallel signed arithmetic	*Parallel add and subtract*	6, 7EM
`SBC`	Subtract with Carry	*ADD, SUB, RSB, ADC, SBC, and RSC*	All
`SBFX`, `UBFX`	Signed, Unsigned Bit Field eXtract	*SBFX and UBFX*	T2
`SDIV`	Signed divide	*SDIV and UDIV*	7M, 7R

Table 4. Location of instructions

Mnemonic	Brief description	See	Arch. [a]
`SEL`	Select bytes according to APSR GE flags	*SEL*	6, 7EM
`SETEND`	Set Endianness for memory accesses	*SETEND*	6, x7M
`SEV`	Set Event	*SEV, WFE, WFI, and YIELD*	K, 6M
`SHADD8`, `SHADD16`, `SHASX`, `SHSUB8`, `SHSUB16`, `SHSAX`	Parallel signed Halving arithmetic	*Parallel add and subtract*	6, 7EM
`SMC`	Secure Monitor Call	*SMC*	Z
`SMLAxy`	Signed Multiply with Accumulate (32 <= 16 x 16 + 32)	*SMULxy and SMLAxy*	5E, 7EM
`SMLAD`	Dual Signed Multiply Accumulate	*SMLAD and SMLSD*	6, 7EM
	(32 <= 32 + 16 x 16 + 16 x 16)		
`SMLAL`	Signed Multiply Accumulate (64 <= 64 + 32 x 32)	*UMULL, UMLAL, SMULL, and SMLAL*	x6M
`SMLALxy`	Signed Multiply Accumulate (64 <= 64 + 16 x 16)	*SMLALxy*	5E, 7EM
`SMLALD`	Dual Signed Multiply Accumulate Long	*SMLALD and SMLSLD*	6, 7EM
	(64 <= 64 + 16 x 16 + 16 x 16)		
`SMLAWy`	Signed Multiply with Accumulate (32 <= 32 x 16 + 32)	*SMULWy and SMLAWy*	5E, 7EM
`SMLSD`	Dual Signed Multiply Subtract Accumulate	*SMLAD and SMLSD*	6, 7EM
	(32 <= 32 + 16 x 16 - 16 x 16)		
`SMLSLD`	Dual Signed Multiply Subtract Accumulate Long	*SMLALD and SMLSLD*	6, 7EM
	(64 <= 64 + 16 x 16 - 16 x 16)		
`SMMLA`	Signed top word Multiply with Accumulate (32 <= TopWord(32 x 32 + 32))	*SMMUL, SMMLA, and SMMLS*	6, 7EM
`SMMLS`	Signed top word Multiply with Subtract (32 <= TopWord(32 - 32 x 32))	*SMMUL, SMMLA, and SMMLS*	6, 7EM
`SMMUL`	Signed top word Multiply (32 <= TopWord(32 x 32))	*SMMUL, SMMLA, and SMMLS*	6, 7EM
`SMUAD`, `SMUSD`	Dual Signed Multiply, and Add or Subtract products	*SMUAD{X} and SMUSD{X}*	6, 7EM
`SMULxy`	Signed Multiply (32 <= 16 x 16)	*SMULxy and SMLAxy*	5E, 7EM
`SMULL`	Signed Multiply (64 <= 32 x 32)	*UMULL, UMLAL, SMULL, and SMLAL*	x6M
`SMULWy`	Signed Multiply (32 <= 32 x 16)	*SMULWy and SMLAWy*	5E, 7EM
`SRS`	Store Return State	*SRS*	T2, x7M
`SSAT`	Signed Saturate	*SSAT and USAT*	6, x6M
`SSAT16`	Signed Saturate, parallel halfwords	*SSAT16 and USAT16*	6, 7EM
`SSUB8`, `SSUB16`, `SSAX`	Parallel signed arithmetic	*Parallel add and subtract*	6, 7EM
`STC`	Store Coprocessor	*LDC, LDC2, STC, and STC2*	x6M
`STC2`	Store Coprocessor	*LDC, LDC2, STC, and STC2*	5, x6M
`STM`	Store Multiple registers	*LDM and STM*	All
`STR`	Store Register with word	*Memory access instructions*	All
`STRB`	Store Register with byte	*Memory access instructions*	All
`STRBT`	Store Register with byte, user mode	*Memory access instructions*	x6M
`STRD`	Store Registers with two words	*Memory access instructions*	5E, x6M
`STREX`	Store Register Exclusive	*LDREX and STREX*	6, x6M
`STREXB`, `STREXH`	Store Register Exclusive Byte, Halfword	*LDREX and STREX*	K, x6M
`STREXD`	Store Register Exclusive Doubleword	*LDREX and STREX*	K, x7M
`STRH`	Store Register with halfword	*Memory access instructions*	All
`STRHT`	Store Register with halfword, user mode	*Memory access instructions*	T2
`STRT`	Store Register with word, user mode	*Memory access instructions*	x6M
`SUB`	Subtract	*ADD, SUB, RSB, ADC, SBC, and RSC*	All
`SUBS pc, lr`	Exception return, no stack	*SUBS pc, lr*	T2, x7M
`SVC` (formerly `SWI`)	SuperVisor Call	*SVC*	All
`SWP`, `SWPB`	Swap registers and memory (ARM only)	*SWP and SWPB*	All, x7M
`SXTAB, SXTAB16, SXTAH`	Signed extend, with Addition	*SXT, SXTA, UXT, and UXTA*	6, 7EM

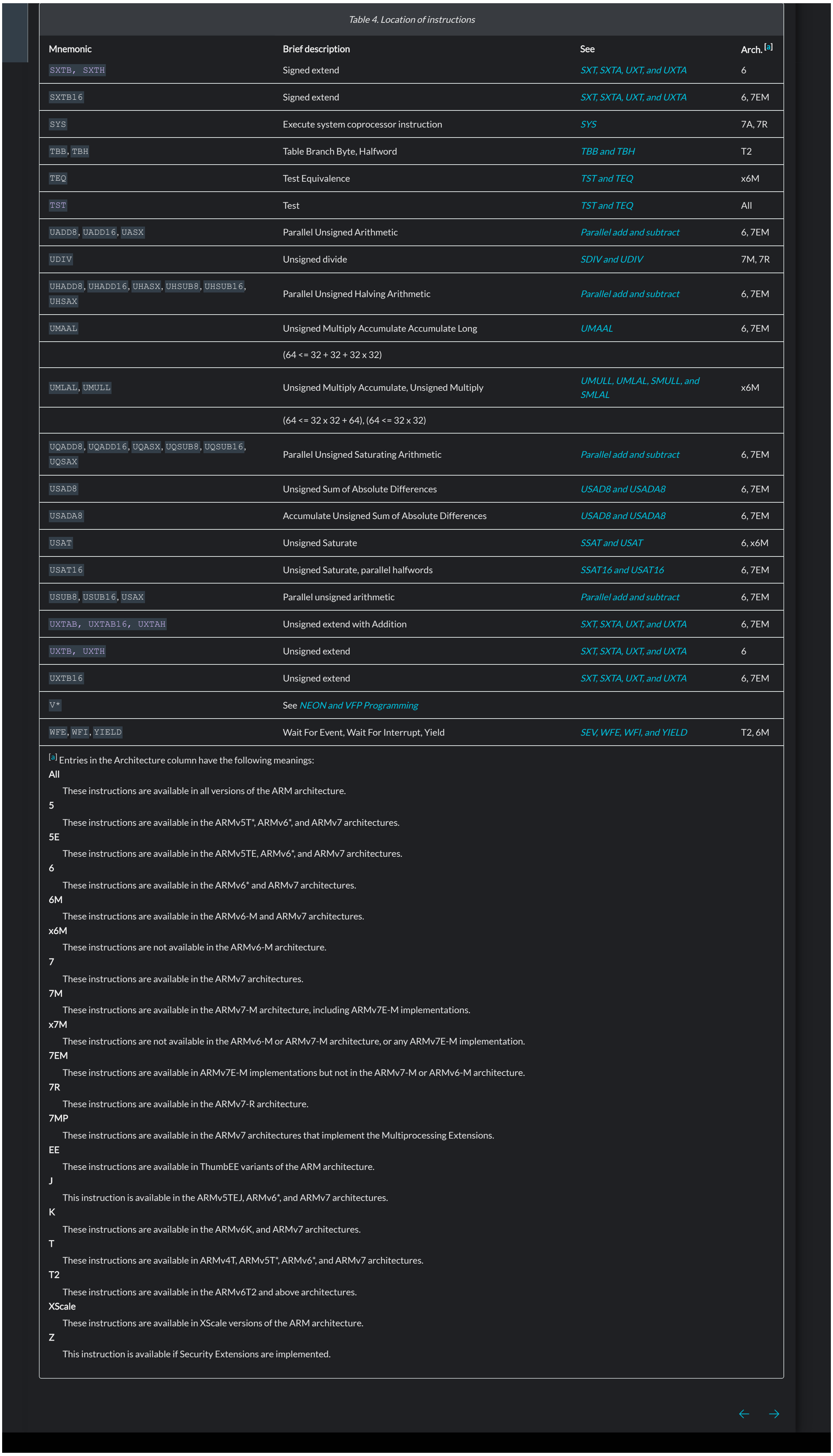

Table 4. Location of instructions

Mnemonic	Brief description	See	Arch. [a]
`SXTB, SXTH`	Signed extend	*SXT, SXTA, UXT, and UXTA*	6
`SXTB16`	Signed extend	*SXT, SXTA, UXT, and UXTA*	6, 7EM
`SYS`	Execute system coprocessor instruction	*SYS*	7A, 7R
`TBB`, `TBH`	Table Branch Byte, Halfword	*TBB and TBH*	T2
`TEQ`	Test Equivalence	*TST and TEQ*	x6M
`TST`	Test	*TST and TEQ*	All
`UADD8`, `UADD16`, `UASX`	Parallel Unsigned Arithmetic	*Parallel add and subtract*	6, 7EM
`UDIV`	Unsigned divide	*SDIV and UDIV*	7M, 7R
`UHADD8`, `UHADD16`, `UHASX`, `UHSUB8`, `UHSUB16`, `UHSAX`	Parallel Unsigned Halving Arithmetic	*Parallel add and subtract*	6, 7EM
`UMAAL`	Unsigned Multiply Accumulate Accumulate Long	*UMAAL*	6, 7EM
	(64 <= 32 + 32 + 32 x 32)		
`UMLAL`, `UMULL`	Unsigned Multiply Accumulate, Unsigned Multiply	*UMULL, UMLAL, SMULL, and SMLAL*	x6M
	(64 <= 32 x 32 + 64), (64 <= 32 x 32)		
`UQADD8`, `UQADD16`, `UQASX`, `UQSUB8`, `UQSUB16`, `UQSAX`	Parallel Unsigned Saturating Arithmetic	*Parallel add and subtract*	6, 7EM
`USAD8`	Unsigned Sum of Absolute Differences	*USAD8 and USADA8*	6, 7EM
`USADA8`	Accumulate Unsigned Sum of Absolute Differences	*USAD8 and USADA8*	6, 7EM
`USAT`	Unsigned Saturate	*SSAT and USAT*	6, x6M
`USAT16`	Unsigned Saturate, parallel halfwords	*SSAT16 and USAT16*	6, 7EM
`USUB8`, `USUB16`, `USAX`	Parallel unsigned arithmetic	*Parallel add and subtract*	6, 7EM
`UXTAB, UXTAB16, UXTAH`	Unsigned extend with Addition	*SXT, SXTA, UXT, and UXTA*	6, 7EM
`UXTB, UXTH`	Unsigned extend	*SXT, SXTA, UXT, and UXTA*	6
`UXTB16`	Unsigned extend	*SXT, SXTA, UXT, and UXTA*	6, 7EM
`V*`	See *NEON and VFP Programming*		
`WFE`, `WFI`, `YIELD`	Wait For Event, Wait For Interrupt, Yield	*SEV, WFE, WFI, and YIELD*	T2, 6M

[a] Entries in the Architecture column have the following meanings:

All

These instructions are available in all versions of the ARM architecture.

5

These instructions are available in the ARMv5T*, ARMv6*, and ARMv7 architectures.

5E

These instructions are available in the ARMv5TE, ARMv6*, and ARMv7 architectures.

6

These instructions are available in the ARMv6* and ARMv7 architectures.

6M

These instructions are available in the ARMv6-M and ARMv7 architectures.

x6M

These instructions are not available in the ARMv6-M architecture.

7

These instructions are available in the ARMv7 architectures.

7M

These instructions are available in the ARMv7-M architecture, including ARMv7E-M implementations.

x7M

These instructions are not available in the ARMv6-M or ARMv7-M architecture, or any ARMv7E-M implementation.

7EM

These instructions are available in ARMv7E-M implementations but not in the ARMv7-M or ARMv6-M architecture.

7R

These instructions are available in the ARMv7-R architecture.

7MP

These instructions are available in the ARMv7 architectures that implement the Multiprocessing Extensions.

EE

These instructions are available in ThumbEE variants of the ARM architecture.

J

This instruction is available in the ARMv5TEJ, ARMv6*, and ARMv7 architectures.

K

These instructions are available in the ARMv6K, and ARMv7 architectures.

T

These instructions are available in ARMv4T, ARMv5T*, ARMv6*, and ARMv7 architectures.

T2

These instructions are available in the ARMv6T2 and above architectures.

XScale

These instructions are available in XScale versions of the ARM architecture.

Z

This instruction is available if Security Extensions are implemented.

I Develop For

- AI
- Automotive
- Embedded and Microcontrollers
- Internet of Things
- Laptops and Desktops
- Mobile, Graphics, and Gaming
- Servers and Cloud Computing

Learn

- Documentation
- Learning Paths
- On-Demand Videos

Community and Blogs

- Communities
- Blogs
- Developer Events
- Discord
- Forums
- Groups

Support

- Open a Support Case
- My Support Cases
- Contact Support
- Product Support Forums
- Documentation
- Downloads
- Licensing
- Support and Training Options

CPU and Hardware

- Architectures
- Compare IP
- Compute Subsystems
- Corstone
- Memory Model Tool
- Performance Analysis
- Processors
- Reference Designs

About Arm

- Leadership
- Arm Offices
- Careers
- Contact Us
- Newsroom
- Trust Center

Cookie Policy | Glossary | Terms of Use | Privacy Policy | Accessibility | Subscription Center | Trademarks