A3.2 Alignment support

Instructions in the ARM architecture are aligned as follows:

- ARM instructions are word-aligned
- Thumb and ThumbEE instructions are halfword-aligned
- Java bytecodes are byte-aligned.

The data alignment behavior supported by the ARM architecture has changed significantly between ARMv4 and ARMv7. This behavior is indicated by the SCTLR.U bit, see:

- *c1, System Control Register (SCTLR)* on page B3-96 for a VMSAv7 implementation
- *c1, System Control Register (SCTLR)* on page B4-45 for a PMSAv7 implementation
- *c1, System Control Register (SCTLR)* on page AppxG-34 for architecture versions before ARMv7.

This bit defines the alignment behavior of the memory system for data accesses. Table A3-1 shows the values of SCTLR.U for the different architecture versions.

Table A3-1 SCTLR.U bit values for different architecture versions

Architecture version	SCTLR.U value
Before ARMv6	0
ARMv6	0 or 1
ARMv7	1

On an ARMv6 processor, the SCTLR.U bit indicates which of two possible alignment models is selected:

U == 0 The processor implements the legacy alignment model. This is described in *Alignment* on page AppxG-6.

Note

The use of U == 0 is deprecated in ARMv6T2, and is obsolete from ARMv7.

U == 1 The processor implements the alignment model described in this section. This model supports unaligned data accesses.

ARMv7 requires the processor to implement the alignment model described in this section.

A3.2.1 Unaligned data access

An ARMv7 implementation must support unaligned data accesses. The SCTLR.U bit is RAO to indicate this support. The SCTLR.A bit, the strict alignment bit, controls whether strict alignment is required. The checking of load and store alignment depends on the value of this bit. For more information, see *c1, System Control Register (SCTLR)* on page B3-96 for a VMSA implementation, or *c1, System Control Register (SCTLR)* on page B4-45 for a PMSA implementation.

Table A3-2 shows how the checking of load and store alignment depends on the instruction type and the value of SCTLR.A.

Table A3-2 Alignment requirements of load/store instructions

Instructions	Alignment check	Result if check fails when:	
		SCTLR.A == 0	SCTLR.A == 1
LDRB, LDREXB, LDRBT, LDRSB, LDRSBT, STRB, STREXB, STRBT, SWPB, TBB	None	-	-
LDRH, LDRHT, LDRSH, LDRSHT, STRH, STRHT, TBH	Halfword	Unaligned access	Alignment fault
LDREXH, STREXH	Halfword	Alignment fault	Alignment fault
LDR, LDRT, STR, STRT	Word	Unaligned access	Alignment fault
LDREX, STREX	Word	Alignment fault	Alignment fault
LDREXD, STREXD	Doubleword	Alignment fault	Alignment fault
All forms of LDM, LDRD, PUSH, POP, RFE, SRS, all forms of STM, STRD, SWP	Word	Alignment fault	Alignment fault
LDC, LDC2, STC, STC2	Word	Alignment fault	Alignment fault
VLDM, VLDR, VSTM, VSTR	Word	Alignment fault	Alignment fault
VLD1, VLD2, VLD3, VLD4, VST1, VST2, VST3, VST4, all with standard alignment[a]	Element size	Unaligned access	Alignment fault
VLD1, VLD2, VLD3, VLD4, VST1, VST2, VST3, VST4, all with @<align> specified[a]	As specified by @<align>	Alignment fault	Alignment fault

a. These element and structure load/store instructions are only in the Advanced SIMD extension to the ARMv7 ARM and Thumb instruction sets. ARMv7 does not support the pre-ARMv6 alignment model, so you cannot use that model with these instructions.

A3.2.2 Cases where unaligned accesses are UNPREDICTABLE

The following cases cause the resulting unaligned accesses to be UNPREDICTABLE, and overrule any successful load or store behavior described in *Unaligned data access* on page A3-5:

- Any load instruction that is not faulted by the alignment restrictions and that loads the PC has UNPREDICTABLE behavior if ~~it~~ the address it loads from is not word-aligned.
- Any unaligned access that is not faulted by the alignment restrictions and that accesses memory with the Strongly-ordered or Device attribute has UNPREDICTABLE behavior.

Note

These memory attributes are described in *Memory types and attributes and the memory order model* on page A3-24.

A3.2.3 Unaligned data access restrictions in ARMv7 and ARMv6

ARMv7 and ARMv6 have the following restrictions on unaligned data accesses:

- Accesses are not guaranteed to be single-copy atomic, see *Atomicity in the ARM architecture* on page A3-26. An access can be synthesized out of a series of aligned operations in a shared memory system without guaranteeing locked transaction cycles.
- Unaligned accesses typically take a number of additional cycles to complete compared to a naturally aligned transfer. The real-time implications must be analyzed carefully and key data structures might need to have their alignment adjusted for optimum performance.
- If an unaligned access occurs across a page boundary, the operation can abort on either or both halves of the access.

Shared memory schemes must not rely on seeing monotonic updates of non-aligned data of loads and stores for data items larger than byte wide. For more information, see *Atomicity in the ARM architecture* on page A3-26.

Unaligned access operations must not be used for accessing Device memory-mapped registers. They must only be used with care in shared memory structures that are protected by aligned semaphores or synchronization variables.

