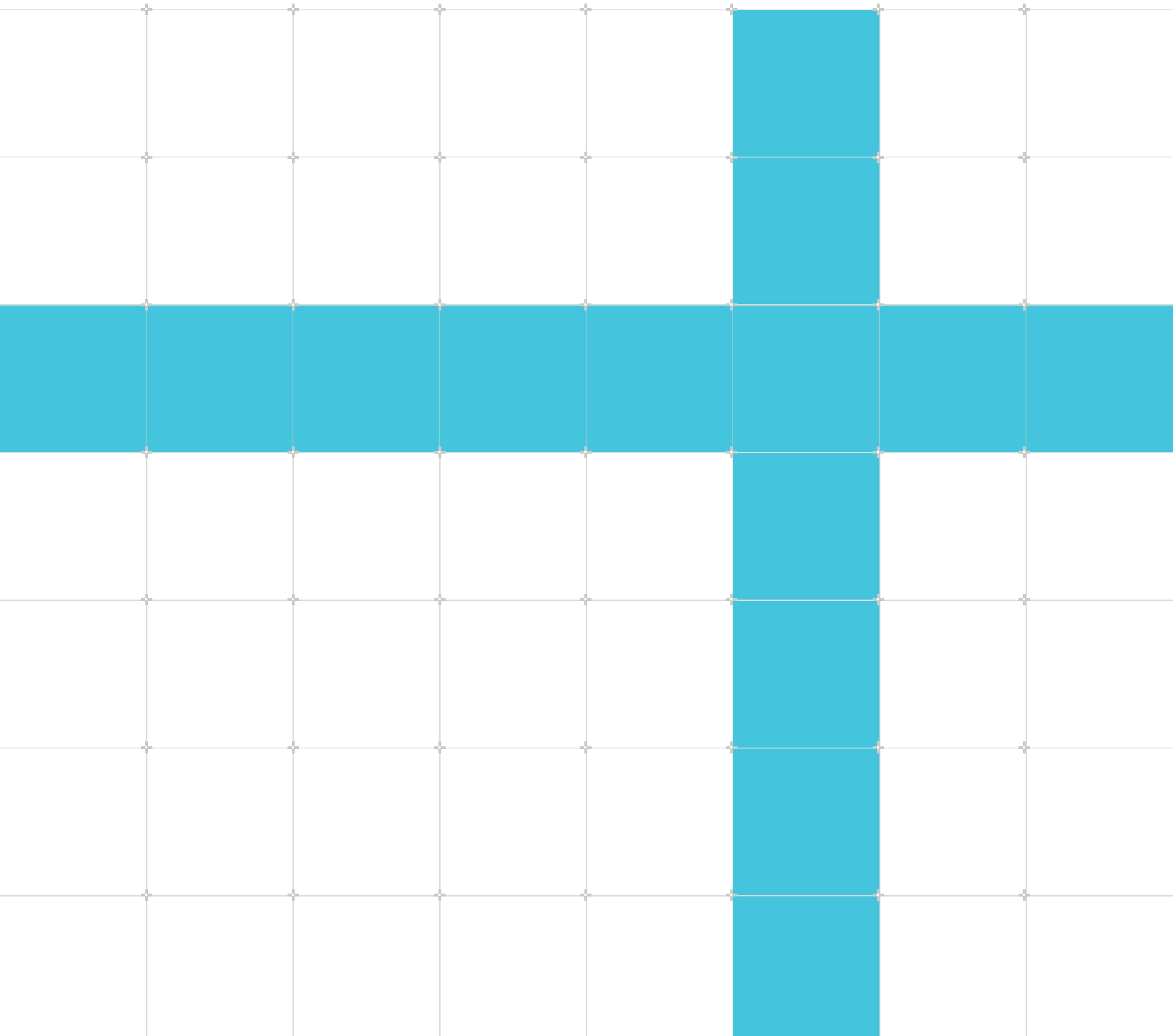
Release notes for the A64 Instruction Set Architecture for Arm A-profile Architecture

2024-09

Issue 01
109389_2024-09_01_en



Release notes for the A64 Instruction Set Architecture for Arm A-profile Architecture

Release information

Document history

Issue	Date	Confidentiality	Change
2024_09-01	30 September 2024	Non-Confidential	2024-09 release
2024_06-01	5 July 2024	Non-Confidential	2024-06 release
2024_04-01	27 March 2024	Non-Confidential	2024-03 release
2023_12-01	19 December 2023	Non-Confidential	2023-12 release
2023_09-01	29 September 2023	Non-Confidential	2023-09 release

Proprietary Notice

with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Release notes for A64 ISA XML for A-profile Architecture (2024-09)

30 September 2024

Product Status

The information relating to the 2024 Extensions is at Alpha quality. Alpha quality means that most major features of the specification are described in this release, but some features and details might be missing. The information relating to the rest of the A-profile Architecture is at Beta quality. Beta quality means that all major features of the specification are described, but some details might be missing.

Change history

The following changes are made to instruction descriptions:

- In SVE FEXPA, the formula provided for the coefficient table entries is corrected.
- The encoding field names in the following instructions are changed to match the standard ISA format:
  - ADDG.
  - SUBG.
  - STG.
  - STGP.
  - STZG.
  - ST2G.
  - STZ2G.
  - STGM.
  - STZGM.
  - CMPP.
  - GMI.
  - IRG.
  - LDG.
  - LDGM.
  - SUBP.
  - SUBPS.
- Advanced SIMD instructions not previously guarded by a feature are guarded by FEAT_AdvSIMD. Floating-point instructions not previously guarded by a feature are guarded by FEAT_FP. FEAT_AdvSIMD and FEAT_FP are added where required to other instructions.
- The description of the following instructions are updated to clarify the behavior when a write is not performed:

    - CAS, CASA, CASAL, CASL.
    - CASB, CASAB, CASALB, CASLB.
    - CASH, CASAH, CASALH, CASLH.
    - CASP, CASPA, CASPAL, CASPL.
- The description of the following instructions are updated to clarify the behavior of instructions if the compare fails or the RCW Checks fail:
    - RCWCASP, RCWCASPA, RCWCASPAL, RCWCASPL.
    - RCWSCAS, RCWSCASA, RCWSCASAL, RCWSCASL.
- The description of the following instructions are updated to clarify the behavior of instructions if the compare fails, the RCW Checks fail or the RCWS Checks fail.
    - RCWSCAS, RCWSCASA, RCWSCASAL, RCWSCASL.
    - RCWSCASP, RCWSCASPA, RCWSCASPAL, RCWSCASPL.

The following changes are made to the pseudocode:

- Descriptions that previously referred to the Arm FP8 E4M3 and E5M2 formats are changed to use the equivalent OCP OFP8 formats.
- The pseudocode is updated to relax the memory type requirements for atomic accesses with FEAT_LSE2.
- TLBIMatch() uses the function UseVMID(), which wrongly uses current configuration of the target PE of the TLBI broadcast. This is corrected by comparing use_vmid, which is calculated at the time of caching the TLB entry, similarly use_vmid is calculated when issuing a TLBI operation to correspond with the configuration of the PE broadcasting TLBI.
- The pseudocode function AArch64.IncrementCycleCounter() is corrected to signal an overflow from bit 31 when AArch32 is implemented and the PMCR_EL0.LC field is 0.
- The pseudocode function AArch64.FaultSyndrome() is corrected to set the ESR_EL2 syndrome for a Watchpoint triggered by an MRS or MSR instruction converted to a memory read or write by FEAT_NV2.
- The decode pseudocode for certain A64 instructions is updated to show that a Branch Target exception is higher priority than an **UNDEFINED** instruction exception.
- The decode pseudocode for the HLT instruction is updated to show that an **UNDEFINED** exception has priority than a Branch Target Exception.
- The instructions that perform a check for an **UNDEFINED** instruction exception in the decode phase are updated to call EndOfDecode(Decode_UNDEF), which will ensure that a Branch Target exception is higher priority than an **UNDEFINED** instruction exception.
- In the pseudocode function AArch64.CheckTimerConditions(), for the Physical Secure timer comparing CNTPS_CVAL_EL1 in IsTimerConditionMet(), the offset is corrected to be 0.
- The pseudocode function AArch64.UnalignedAccessFaults() is corrected to generate an alignment fault when SCTLR_ELx.nAA is 0 and the access crosses a 16-byte boundary.
- The assigned cachetype by the pseudocode function AArch64.DC() is corrected.
- The function DecodeSW() is enhanced to decode the input value to return the appropriate index of the cache set and way indexes.

- The IsFeatureImplemented() checks are added in decode pseudocode for the instructions belonging to the following features:
  - FEAT_BTI.
  - FEAT_LOR.
  - FEAT_LRCPC.
  - FEAT_LRCPC2.
  - FEAT_LRCPC3.
  - FEAT_PAuth.
  - FEAT_RPRFM.
- The operational pseudocode for LDIAPP and STILP is corrected to perform a single memory access.
- The pseudocode function AArch64.S1Translate() is updated to show all effects on tagged accesses when SCTLR_ELx.C is 0.
- The instructions SDIV, UDIV are updated using integer division.
- In the SYSP instruction alias condition, the usage of SysOp() is corrected to SysOp128() so that it leads to TLBIP instructions.

XML Changes

The following changes are made to the SVE and SME Instruction Set Architecture XML:

- The arch_variant tag and its attributes name and feature are updated to have consistent attributes.
- The title and id information are updated to be aligned to base and SIMD&FP instructions.
- The brief tag is updated to contain para elements.
- The description tag is replaced with an authored tag, to be consistent with base and SIMD&FP instructions.
- The ps_name property is updated to be consistent with base and SIMD&FP instructions.
- In the SME EXT and SPLICE insstructions, the value of takes_movprfx is updated to be implicit (False), as for other instructions.
- encodedin is updated to reference the whole encoding field.
- encodedin is updated to precisely follow the fields order.
- Additional sm_policy attributes have been added where needed.
- The instructions PMULLB and PMULLT are updated to remove sm_policy attribute information.
- The aliastablehook tag is generated for all aliases.
- Fields that are defined in the current instruction diagram appear.
- The symbol_link tag is updated to be consistent with base and SIMD&FP instructions.
- Some class id names are updated.
- The attributes name, mylink, and enclabels in the ps tag are updated.

- The syntax for the “For the variant” text in symbol accounts is updated to be consistent.
- The attributes aliaspageid and hover in the aliasref tag are updated to be consistent with base and SIMD&FP instructions.
- Rendering is updated to show separation bars only between fields at an instruction’s own level. Fields that are defined in the current instruction diagram appear.
- Some adjacent field names are combined to form one 2-bit field.
- Arm-defined words are updated to be standarised across A64 ISA.
- In the Assembler Symbols section, the order of the symbols appearing is updated to be more consistent with the order used for base A64 instructions.
- The rendering for encoding conditions such as “size != ‘00’” and “Rm != ‘11111’” is updated to show in the regdiagram. This applies to all the instructions where the condition is present at group level.
- In the Assembler Symbols section, the encoding compression in tables is updated.
- In SME ZERO (tiles), the assembly syntax is updated to make the curly braces explicit.
- In the Assembler Symbols section, the rendering for tables is updated not to show a reserved row. Many simple clarifications and corrections are also present, but are too small to be listed here. Some minor formatting changes are suppressed and not highlighted in the diff output.

Limitations of Arm pseudocode

The pseudocode statements IMPLEMENTATION_DEFINED, SEE, **UNDEFINED**, and **UNPREDICTABLE** indicate behavior that differs from that indicated by the pseudocode being executed. If one of them is encountered:

- Earlier behavior indicated by the pseudocode is only specified as occurring to the extent required to determine that the statement is executed.
- No subsequent behavior indicated by the pseudocode occurs. For more information, see Special statements in Arm ARM. The pseudocode descriptions have several limitations. These are mainly since, for clarity and brevity, the pseudocode is a sequential and mostly deterministic language. These limitations include:
- Pseudocode does not describe the ordering requirements when an instruction generates multiple memory accesses. For a description of the ordering requirements on memory accesses, see External ordering constraints in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487).
- Pseudocode does not describe the exact ordering requirements when a single floating-point instruction generates more than one floating-point exception and one or more of those floating-point exceptions is trapped. Combinations of floating-point exceptions in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487) describes the exact rules. Note: There is no limitation in the case where all the floating-point exceptions are untrapped, because the pseudocode specifies the same behavior as the cross-referenced section.
- When a vector operation or another operation could be performed via a concurrent set of operations that are not architecturally ordered, the pseudocode still presents a sequential deterministic detail.

- An exception can be taken during execution of the pseudocode for an instruction, either explicitly as a result of the execution of a pseudocode function such as Abort(), or implicitly, for example if an interrupt is taken during execution of an LDM instruction, pair load-store instructions, SVE vector instructions, Memory copy and memory set instructions etc.. If this happens, the pseudocode does not describe the extent to which the normal behavior of the instruction occurs. To determine that, see the descriptions of the exceptions in Handling exceptions that are taken to an Exception level using AArch32 and Definition of a precise exception and imprecise exception in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487).
- Pseudocode does not describe the exact rules when an AArch32 instruction that generates any of the following fails its condition code check:
  - **UNDEFINED** instruction.
  - Hyp trap.
  - Monitor trap.
  - Trap to AArch64 exception.
  - Conditional execution of undefined instructions.
  - EL2 configurable controls
  - EL3 configurable controls
  - Configurable instruction controls
- Where a significant aspect of the behavior is IMPLEMENTATION DEFINED, pseudocode may present only the declarations of the functions - the details of these functions is implementation defined.
- Pseudocode does not present the possible observability due to speculative execution.
- Pseudocode presents various details of the architecture - but does not show how the details can be combined to form a single definition. There are various aspects of such a detail that are also not presented. Notably:
  - Pseudocode does not show the details of fetching, decoding, and linking to instruction execution.
  - Pseudocode for combining the instruction shared decode, decode, and operation is not shown.
- Pseudocode presents all the architectural state as global state. The possible implications of multiple PEs or other components is not shown.
- Below details are either not shown or have noted limitations in the pseudocode:
  - Self-hosted trace and external trace.
  - Pseudocode for modeling the register state or side effects. The accessibility details of a direct or external access such as traps etc are shown.
  - Generation of all architectural and micro-architectural Performance Monitoring Events. **Note**: Some architectural event generation is shown.
  - Construction of Statistical Profiling Extension records.
  - Statistical Profiling functionality for 2023 features.

    - Behavior of instructions in Debug state when the behavior is **UNPREDICTABLE** is presented as if the instruction is executed identical to how it is when not in Debug state.
    - Activity Monitor Events and Counters.
    - Generic Interrupt Controller functionality.
    - External memory system.
    - External agents such as debugger.
    - PE behaviors that would lead to unrecoverable or uncontainable errors.
    - Where the behavior is **IMPLEMENTATION DEFINED** or CONSTRAINED **UNPREDICTABLE**, not all possibilities may be shown. Sometimes the pseudocode may present a simplified, but architecturally compatible view. In some situations, the possible behaviors may be outlined in a comment.
- The following architectural features are at Alpha quality and are above the limitations described below due to recency and completion of validation:
    - Halting Debug
    - Statistical Profiling Extension.
    - Performance Monitoring Events.

Known issues

All issues identified in the below list will be fixed in a future release.

- The assembler syntax for MRS and MSR instructions with unnamed registers will be clarified.
- For the following SME and SVE2.1 instructions, the feature conditions stated in the pseudocode are correct, but the feature conditions stated above the encoding diagrams are incomplete:
    - PSEL.
    - REVD.
    - BFMLSLB, BFMLSLT.
    - FDOT, SDOT (two-way), UDOT (two-way).
    - FCLAMP, SCLAMP, UCLAMP.
    - SQCVTN, SQCVTUN, SQRSHRN, SQRSHRUN, UQCVTN, UQRSHRN.
    - WHILEGE, WHILEGT, WHILEHI, WHILEHS, WHILELE, WHILELO, WHILELS, WHILELT (predicate pair).
- The pseudocode needs to be updated for a watchpoint debug event triggered by an SVE contiguous load/store instruction when the PE is in Streaming SVE mode, or by an SME load/store instruction, where the value in FAR_ELx or EDWAR must be the highest watchpointed address, not the highest rounded address.
- In the following SME instructions, the feature conditions stated in the pseudocode are correct, but the feature conditions stated above the encoding diagrams are incomplete:
    - SMLALL (multiple vectors), SMLALL (multiple and single vector).
    - SMLSLL (multiple vectors), SMLSLL (multiple and single vector).

  - UMLALL (multiple vectors), UMLALL (multiple and single vector).
  - UMLSLL (multiple vectors), UMLSLL (multiple and single vector).
  - SDOT (4-way, multiple and single vector), SDOT (4-way, multiple vectors).
  - UDOT (4-way, multiple and single vector), UDOT (4-way, multiple vectors).
  - FMLA (multiple and single vector), FMLA (multiple vectors).
  - FMLS (multiple and single vector), FMLS (multiple vectors).
  - ADD (array results, multiple and single vector), ADD (array results, multiple vectors), ADD (array accumulators).
  - SUB (array results, multiple and single vector), SUB (array results, multiple vectors), SUB (array accumulators).
  - FADD.
  - FSUB.
- Inconsistencies regarding FEAT_FP and FEAT_AdvSIMD feature dependencies in encoding diagrams and pseudocode will be corrected.

Potential Upcoming Changes

The details of the architecture are presented in pseudocode in Architecture Specification Language (ASL). Arm is defining a new version of the Architecture Specification Language, ASL1, to improve and expand the capabilities of the language. Please see https://developer.arm.com/Architectures/Architecture%20Specification%20Language for details on this language and a previous preview release of the XML. Arm will be publishing an equivalent release in ASL1 format next year. Additional headings, including “Encoding” and “Decode for this encoding” will be included on instruction pages. Potential upcoming improvements to the pseudocode are as follows:

- Clarify when pseudocode integer divide operator (DIV) is exact or if there is a remainder, how the rounding is done.