I'm not a robot

0 ratings0% found this document useful (0 votes)334 viewsThis document provides a summary of shortcut keys in Microsoft Word 2007. It lists shortcuts for common commands like opening and saving documents, formatting text, finding and replacing tex…SaveSave MS Word 2007 Shortcut Keys For Later0%0% found this document useful, undefined Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. 64-bit extension of the ARM architecture ARM AArch64 (64-bit)Introduced2011; 14 years ago (2011)VersionARMv8-R, ARMv8-A, ARMv8.1-A, ARMv8.2-A, ARMv8.3-A, ARMv8.4-A, ARMv8.5-A, ARMv8.6-A, ARMv8.7-A, ARMv8.8-A, ARMv8.9-A, ARMv9.0-A, ARMv9.1-A, ARMv9.2-A, ARMv9.3-A, ARMv9.4-A, ARMv9.5-A, ARMv9.6-AEncodingAArch64/A64 and AArch32/A32 use 32-bit instructions, AArch32/T32 (Thumb-2) uses mixed 16- and 32-bit instructions[1]EndiannessBi (little as default)ExtensionsSVE, SVE2, SME, AES, SM3, SM4, SHA, CRC32, RNDR, TME; All mandatory: Thumb-2, Neon, VFPv4-D16, VFPv4; obsolete: JazelleRegistersGeneral-purpose31 × 64-bit integer registers[1]Floating point32 × 128-bit registers[1] for scalar 32- and 64-bit FP or SIMD FP or integer; or cryptography AArch64 or ARM64 is the 64-bit Execution state of the ARM architecture family. It was first introduced with the Armv8-A architecture, and has had many extension updates.[2] An Execution state, in ARMv8-A, ARMv8-R, and ARMv9-A, defines the number of bits in the primary processor registers, the available instruction sets, and other aspects of the processor's execution environment. In those versions of the Arm architecture, there are two Execution states, the 64-bit AArch64 Execution state and the 32-bit AArch32 Execution state.[3] 64-bit: Execution state: AArch64. Instruction sets: A64. 32-bit: Execution state: AArch32. Instruction sets: A32 + T32. Example: ARMv8-R, Cortex-A32.[4] New instruction set, A64: Has 31 general-purpose 64-bit registers. Has dedicated zero or stack pointer (SP) register (depending on instruction). The program counter (PC) is no longer directly accessible as a register. Instructions are still 32 bits long and mostly the same as A32 (with LDM/STM instructions and most conditional execution dropped). Has paired loads/stores (in place of LDM/STM). No predication for most instructions (except branches). Most instructions can take 32-bit or 64-bit arguments. Addresses assumed to be 64-bit. Advanced SIMD (Neon) enhanced: Has 32 × 128-bit registers (up from 16), also accessible via VFPv4. Supports double-precision floating-point format. Fully IEEE 754 compliant. AES encrypt/decrypt and SHA-1/SHA-2 hashing instructions also use these registers. A new exception system: Fewer banked registers and modes. Memory translation from 48-bit virtual addresses based on the existing Large Physical Address Extension (LPAE), which was designed to be easily extended to 64-bit. Extension: Data gathering hint (ARMv8.0-DGH). AArch64 was introduced in ARMv8-A and is included in subsequent versions of ARMv8-A, and in all versions of ARMv9-A. It was also introduced in ARMv8-R as an option, after its introduction in ARMv8-A; it is not included in ARMv8-M. The main opcode for selecting which group an A64 instruction belongs to is at bits 25–28. A64 instruction formats Type Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Reserved 0 op0 0 0 0 0 op1 SME 1 op0 0 0 0 0 Varies Unallocated 0 0 0 1 SVE 0 0 1 0 Varies Unallocated 0 0 1 1 Data Processing — Immediate PC-rel. op immlo 1 0 0 0 0 immhi Rd Data Processing — Immediate Others sf 1 0 0 01–11 Rd Branches + System Instructions op0 1 0 1 op1 op2 Load and Store Instructions op0 1 op1 0 op2 op3 op4 Data Processing — Register sf op0 op1 1 0 1 op2 op3 Data Processing — Floating Point and SIMD op0 1 1 1 op1 op2 op3 See also: Comparison of ARMv8-A processors Armv8-A platform with Cortex-A57/A53 MPCore big.LITTLE CPU chip Announced in October 2011,[5] ARMv8-A represents a fundamental change to the ARM architecture. It adds an optional 64-bit Execution state, named "AArch64", and the associated new "A64" instruction set, in addition to a 32-bit Execution state, "AArch32", supporting the 32-bit "A32" (original 32-bit Arm) and "T32" (Thumb/Thumb-2) instruction sets. The latter instruction sets provide user-space compatibility with the existing 32-bit ARMv7-A architecture. ARMv8-A allows 32-bit applications to be executed in a 64-bit OS, and a 32-bit OS to be under the control of a 64-bit hypervisor.[1] ARM announced their Cortex-A53 and Cortex-A57 cores on 30 October 2012.[6] Apple was the first to release an ARMv8-A compatible core (Cyclone) in a consumer product (iPhone 5S). AppliedMicro, using an FPGA, was the first to demo ARMv8-A.[7] The first ARMv8-A SoC from Samsung is the Exynos 5433 used in the Galaxy Note 4, which features two clusters of four Cortex-A57 and Cortex-A53 cores in a big.LITTLE configuration; but it will run only in AArch32 mode.[8] ARMv8-A includes the VFPv3/v4 and advanced SIMD (Neon) as standard features in both AArch32 and AArch64. It also adds cryptography instructions supporting AES, SHA-1/SHA-256 and finite field arithmetic.[9] An ARMv8-A processor can support one or both of AArch32 and AArch64; it may support AArch32 and AArch64 at lower Exception levels and only AArch64 at higher Exception levels.[10] For example, the ARM Cortex-A32 supports only AArch32,[11] the ARM Cortex-A34 supports only AArch64,[12] and the ARM Cortex-A72 supports both AArch64 and AArch32.[13] An ARMv9-A processor must support AArch64 at all Exception levels, and may support AArch32 at EL0.[10] In December 2014, ARMv8.1-A,[14] an update with "incremental benefits over v8.0", was announced. The enhancements fell into two categories: changes to the instruction set, and changes to the exception model and memory translation. Instruction set enhancements included the following: A set of AArch64 atomic read-write instructions. Additions to the Advanced SIMD instruction set for both AArch32 and AArch64 to enable opportunities for some library optimizations: Signed Saturating Rounding Doubling Multiply Accumulate, Returning High Half. Signed Saturating Rounding Doubling Multiply Subtract, Returning High Half. The instructions are added in vector and scalar forms. A set of AArch64 load and store instructions that can provide memory access order that is limited to configurable address regions. The optional CRC instructions in v8.0 become a requirement in ARMv8.1. Enhancements for the exception model and memory translation system included the following: A new Privileged Access Never (PAN) state bit provides control that prevents privileged access to user data unless explicitly enabled. An increased VMID range for virtualization; supports a larger number of virtual machines. Optional support for hardware update of the page table access flag, and the standardization of an optional, hardware updated, dirty bit mechanism. The Virtualization Host Extensions (VHE). These enhancements improve the performance of Type 2 hypervisors by reducing the software overhead associated when transitioning between the Host and Guest operating systems. The extensions allow the Host OS to execute at EL2, as opposed to EL1, without substantial modification.[15] A mechanism to free up some translation table bits for operating system use, where the hardware support is not needed by the OS. Top byte ignore for memory tagging.[16] ARMv8.2-A was announced in January 2016.[17] Its enhancements fall into four categories: Optional half-precision floating-point data processing (half-precision was already supported, but not for processing, just as a storage format.) Memory model enhancements. Introduction of Reliability, Availability and Serviceability Extension (RAS Extension). Introduction of statistical profiling. The Scalable Vector Extension (SVE) is "an optional extension to the ARMv8.2-A architecture and newer" developed specifically for vectorization of high-performance computing scientific workloads.[18][19] The specification allows for variable vector lengths to be implemented from 128 to 2048 bits. The extension is complementary to, and does not replace, the NEON extensions. A 512-bit SVE variant has already been implemented on the Fugaku supercomputer using the Fujitsu A64FX ARM processor; this computer[20] was the fastest supercomputer in the world for two years, from June 2020[21] to May 2022.[22] A more flexible version, 2x256 SVE, was implemented by the AWS Graviton3 ARM processor. SVE is supported by GCC, with GCC 8 supporting automatic vectorization[19] and GCC 10 supporting C intrinsics. As of July 2020[update], LLVM and clang support C and IR intrinsics. ARM's own fork of LLVM supports auto-vectorization.[23] In October 2016, ARMv8.3-A was announced. Its enhancements fell into six categories:[24] Pointer authentication (PAC)[25][26] (AArch64 only); mandatory extension (based on a new block cipher, QARMA[27]) to the architecture (compilers need to exploit the security feature, but as the instructions are in NOP space, they are backwards compatible albeit providing no extra security on older chips). Nested virtualization (AArch64 only). Advanced SIMD complex number support (AArch64 and AArch32); e.g. rotations by multiples of 90 degrees. New FJCVTZS (Floating-point JavaScript Convert to Signed fixed-point, rounding toward Zero) instruction.[28] A change to the memory consistency model (AArch64 only); to support the (non-default) weaker RCpc (Release Consistent processor consistent) model of C++11/C11 (the default C++11/C11 consistency model was already supported in previous ARMv8). ID mechanism support for larger system-visible caches (AArch64 and AArch32). ARMv8.3-A architecture is now supported by (at least) the GCC 7 compiler.[29] In November 2017, ARMv8.4-A was announced. Its enhancements fell into these categories:[30][31][32] "SHA3 / SHA512 / SM3 / SM4 crypto extensions." I.e. optional instructions. Improved virtualization support.[33] Memory Partitioning and Monitoring (MPAM) capabilities. A new Secure EL2 state and Activity Monitors. Signed and unsigned integer dot product (SDOT and UDOT) instructions. In September 2018, ARMv8.5-A was announced. Its enhancements fell into these categories:[34][35][36] Memory Tagging Extension (MTE) (AArch64).[37] Branch Target Indicators (BTI) (AArch64) to reduce "the ability of an attacker to execute arbitrary code". Like pointer authentication, the relevant instructions are no-ops on earlier versions of ARMv8-A. Random Number Generator instructions – "providing Deterministic and True Random Numbers conforming to various National and International Standards". On 2 August 2019, Google announced Android would adopt Memory Tagging Extension (MTE).[38] In March 2021, ARMv9-A was announced. ARMv9-A's baseline is all the features from ARMv8.5.[39][40][41] ARMv9-A also adds: Scalable Vector Extension 2 (SVE2). SVE2 builds on SVE's scalable vectorization for increased fine-grain Data Level Parallelism (DLP), to allow more work done per instruction. SVE2 aims to bring these benefits to a wider range of software including DSP and multimedia SIMD code that currently use Neon.[42] The LLVM/Clang 9.0 and GCC 10.0 development codes were updated to support SVE2.[42][43] Transactional Memory Extension (TME). Following the x86 extensions, TME brings support for Hardware Transactional Memory (HTM) and Transactional Lock Elision (TLE). TME aims to bring scalable concurrency to increase coarse-grained Thread Level Parallelism (TLP), to allow more work done per thread.[42] The LLVM/Clang 9.0 and GCC 10.0 development codes were updated to support TME.[43] Confidential Compute Architecture (CCA).[44][45][46] In September 2019, ARMv8.6-A was announced. Its enhancements fell into these categories:[34][47] General Matrix Multiply (GEMM). Bfloat16 format support. SIMD matrix manipulation instructions, BFDOT, BFMMLA, BFMLAL and BFCVT. Enhancements for virtualization, system management and security. And the following extensions (that LLVM 11 already added support for[48]): Enhanced Counter Virtualization (ARMv8.6-ECV). Fine-Grained Traps (ARMv8.6-FGT). Activity Monitors virtualization (ARMv8.6-AMU). For example, fine-grained traps, Wait-for-Event (WFE) instructions, EnhancedPAC2 and FPAC. The bfloat16 extensions for SVE and Neon are mainly for deep learning use.[49] In September 2020, ARMv8.7-A was announced. Its enhancements fell into these categories:[34][50] Scalable Matrix Extension (SME)(ARMv9.2 only).[51] SME adds new features to process matrices efficiently, such as: Matrix tile storage. On-the-fly matrix transposition. Load/store/insert/extract tile vectors. Matrix outer product of SVE vectors. "Streaming mode" SVE. Enhanced support for PCIe hot plug (AArch64). Atomic 64-byte load and stores to accelerators (AArch64). Wait For Interrupt (WFI) and Wait For Event (WFE) with timeout (AArch64). Branch-Record recording (ARMv9.2 only). Call Stack Recorder In September 2021, ARMv8.8-A and ARMv9.3-A were announced. Their enhancements fell into these categories:[34][52] Non-maskable interrupts (AArch64). Instructions to optimize memcpy() and memset() style operations (AArch64). Enhancements to PAC (AArch64). Hinted conditional branches (AArch64). LLVM 15 supports ARMv8.8-A and ARMv9.3-A.[53] In September 2022, ARMv8.9-A and ARMv9.4-A were announced, including:[54] Virtual Memory System Architecture (VMSA) enhancements. Permission indirection and overlays. Translation hardening. 128-bit translation tables (ARMv9 only). Scalable Matrix Extension 2 (SME2) (ARMv9 only). Multi-vector instructions. Multi-vector predicates. 2b/4b weight compression. 1b binary networks. Range Prefetch. Guarded Control Stack (GCS) (ARMv9 only). Confidential Computing. Memory Encryption Contexts. Device Assignment. In October 2023, ARMv9.5-A was announced, including:[55] FP8 support (E5M2 and E4M3 formats) added to: SME2 SVE2 Advanced SIMD (Neon) Live migration of Virtual Machines using Hardware Dirty state tracking structures (FEAT_HDBSS) Checked Point Arithmetic Support for using a combination of the PC and SP as the modifier when generating or checking Pointer Authentication codes. Support for Realm Management Extension (RME) enabled designs, support for non-secure only in the Granule Protection Tables and the ability to disable certain Physical Address Spaces (PAS). EL3 configuration write-traps. Breakpoint support for address range and mismatch triggering without the need for linking. Support for efficiently delegating SErrors from EL3 to EL2 or EL1. In October 2024, ARMv9.6-A was announced, including:[56] Improved SME efficiency with structured sparsity and quarter tile operations MPAM Domains to better support shared-memory computer systems on multi-chiplet and multi-chip systems Hypervisor memory control for Trace and Statistical Profiling on virtual machines Improved Caching and Data Placement Granular Data Isolation for Confidential Compute Bitwise locking of EL1 system registers Improved scaling of Granular Protection Tables (GPT) for large memory systems New SVE instructions for expand/compact and finding first/last active element Additional unprivileged load and store instructions to enable OS to interact with application memory New compare and branch instruction Injection of Undefined-Instruction exceptions from EL3 This section needs expansion with: examples and additional citations. You can help by adding to it. Relevant discussion may be found on Talk:AArch64. (May 2021) The ARM-R architecture, specifically the Armv8-R profile, is designed to address the needs of real-time applications, where predictable and deterministic behavior is essential. This profile focuses on delivering high performance, reliability, and efficiency in embedded systems where real-time constraints are critical. With the introduction of optional AArch64 support in the Armv8-R profile, the real-time capabilities have been further enhanced. The Cortex-R82[57] is the first processor to implement this extended support, bringing several new features and improvements to the real-time domain.[58] AArch64 Instruction Set (A64): The A64 instruction[28] set in the Cortex-R82 provides 64-bit data handling and operations, which improves performance for certain computational tasks and enhances overall system efficiency.[57] Example Instruction: ADD X0, X1, X2 adds the values in 64-bit registers X1 and X2 and stores the result in X0. This 64-bit operation allows for larger and more complex calculations compared to the 32-bit operations of the previous A32 instruction set. Enhanced Memory Management: Memory Barrier Instructions: The Cortex-R82 introduces improved memory barrier instructions to ensure proper ordering of memory operations, which is critical in real-time systems where the timing of memory operations must be strictly controlled.[59] Data Synchronization Barrier (DSB): Ensures that all data accesses before the barrier are completed before continuing with subsequent operations. Data Memory Barrier (DMB): Guarantees that all memory accesses before the barrier are completed before any memory accesses after the barrier can proceed. Example: In a real-time automotive control system, DSB might be used to ensure that sensor data is fully written to memory before the system proceeds with processing or decision-making, preventing data corruption or inconsistencies. Improved Address Space: 64-bit Addressing: AArch64 allows the Cortex-R82 to address a much larger memory space compared to its 32-bit predecessors, making it suitable for applications requiring extensive memory. Example: A complex industrial automation system can utilize the expanded address space to manage large data sets and buffers more efficiently, improving system performance and capability. Real-Time Performance Enhancements: Interrupt Handling: With AArch64 support, the Cortex-R82 can handle interrupts with lower latency and improved predictability, crucial for real-time operations. Example: In a robotics application, the Cortex-R82's enhanced interrupt handling can ensure timely responses to external stimuli, such as changes in sensor data or control commands.

^ a b c d Grisenthwaite, Richard (2011). "ARMv8-A Technology Preview" (PDF). Archived from the original (PDF) on 11 November 2011. Retrieved 31 October 2011. ^ "Overview". Learn the architecture: Understanding the Armv8.x and Armv9.x extensions. ^ "ARM Cortex-A57 MPCore Processor Technical Reference Manual". Arm Ltd. ^ "Cortex-A32 Processor – ARM". Retrieved 18 December 2016. ^ "ARM Discloses Technical Details Of The Next Version Of The ARM Architecture" (Press release). Arm Holdings. 27 October 2011. Archived from the original on 1 January 2019. Retrieved 20 September 2013. ^ "ARM Launches Cortex-A50 Series, the World's Most Energy-Efficient 64-bit Processors" (Press release). Arm Holdings. Retrieved 31 October 2012. ^ "AppliedMicro Showcases World's First 64-bit ARM v8 Core" (Press release). AppliedMicro. 28 October 2011. Retrieved 11 February 2014. ^ "Samsung's Exynos 5433 is an A57/A53 ARM SoC". AnandTech. Retrieved 17 September 2014. ^ "ARM Cortex-A53 MPCore Processor Technical Reference Manual: Cryptography Extension". ARM. Retrieved 11 September 2016. ^ a b "Impact of implemented Exception levels". Learn the architecture - AArch64 Exception Model. Arm. ^ "Cortex-A32". Arm Developer. ^ "Cortex-A34". Arm Developer. ^ "Cortex-A72". Arm Developer. ^ Brash, David (2 December 2014). "The ARMv8-A architecture and its ongoing development". Retrieved 23 January 2015. ^ "ARM Virtualization: Performance and Architectural Implications". ISCA 2016. 18 June 2016. ^ "Top-byte ignore (TBI)". WikiChip. ^ Brash, David (5 January 2016). "ARMv8-A architecture evolution". Retrieved 7 June 2016. ^ "The scalable vector extension sve for the ARMv8 a architecture". Arm Community. 22 August 2016. Retrieved 8 July 2018. ^ a b "GCC 8 Release Series – Changes, New Features, and Fixes – GNU Project – Free Software Foundation (FSF)". gcc.gnu.org. Retrieved 9 July 2018. ^ "Fujitsu Completes Post-K Supercomputer CPU Prototype, Begins Functionality Trials – Fujitsu Global". www.fujitsu.com (Press release). Retrieved 8 July 2018. ^ "Japan's Fugaku gains title as world's fastest supercomputer" (Press release). www.riken.jp. 23 June 2020. Retrieved 7 December 2020. ^ "ORNL's Frontier First to Break the Exaflop Ceiling". Top500. 30 May 2022. Retrieved 30 May 2022. ^ "❁ D71712 Downstream SVE/SVE2 implementation (LLVM)". reviews.llvm.org. ^ David Brash (26 October 2016). "ARMv8-A architecture – 2016 additions". ^ "[Ping~,AArch64] Add commandline support for -march=armv8.3-a". pointer authentication extension is defined to be mandatory extension on ARMv8.3-A and is not optional ^ "Pointer Authentication on Arm". ARM. Retrieved 5 March 2025. ^ "Qualcomm releases whitepaper detailing pointer authentication on ARMv8.3". 10 January 2017. ^ a b "A64 Floating-point Instructions: FJCVTZS". arm.com. Retrieved 11 July 2019. ^ "GCC 7 Release Series – Changes, New Features, and Fixes". The ARMv8.3-A architecture is now supported. It can be used by specifying the -march=armv8.3-a option. [..] The option -msign-return-address= is supported to enable return address protection using ARMv8.3-A Pointer Authentication Extensions. ^ "Introducing 2017's extensions to the Arm Architecture". community.arm.com. 2 November 2017. Retrieved 15 June 2019. ^ "Exploring dot product machine learning". community.arm.com. 6 December 2017. Retrieved 15 June 2019. ^ "ARM Preps ARMv8.4-A Support For GCC Compiler – Phoronix". www.phoronix.com. Retrieved 14 January 2018. ^ "NEVE: Nested Virtualization Extensions for ARM". SOSP 2017. 14 October 2017. ^ a b c d "ARMv8.x and ARMv9.x extensions and features". Learn the architecture: Understanding the ARMv8.x and ARMv9.x extensions. ^ "Arm Architecture ARMv8.5-A Announcement – Processors blog – Processors – Arm Community". community.arm.com. Retrieved 26 April 2019. ^ "Arm Architecture Reference Manual ARMv8, for ARMv8-A architecture profile". ARM Developer. Retrieved 6 August 2019. ^ "Arm MTE architecture: Enhancing memory safety". community.arm.com. 5 August 2019. Retrieved 27 July 2021. ^ "Adopting the Arm Memory Tagging Extension in Android". Google Online Security Blog. Retrieved 6 August 2019. ^ "Arm's solution to the future needs of AI, security and specialized computing is v9". Arm | The Architecture for the Digital World. Retrieved 27 July 2021. ^ Schor, David (30 March 2021). "Arm Launches ARMv9". WikiChip Fuse. Retrieved 27 July 2021. ^ Frumusanu, Andrei. "Arm Announces ARMv9 Architecture: SVE2, Security, and the Next Decade". www.anandtech.com. Retrieved 27 July 2021. ^ a b c "Arm releases SVE2 and TME for A-profile architecture – Processors blog – Processors – Arm Community". community.arm.com. 18 April 2019. Retrieved 25 May 2019. ^ a b "Arm SVE2 Support Aligning For GCC 10, LLVM Clang 9.0 – Phoronix". www.phoronix.com. Retrieved 26 May 2019. ^ "Design and Verification of the Arm Confidential Compute Architecture". OSDI 2022. Retrieved 11 July 2022. ^ "Unlocking the power of data with Arm CCA". community.arm.com. 23 June 2021. Retrieved 27 July 2021. ^ "Arm Introduces Its Confidential Compute Architecture". WikiChip Fuse. 23 June 2021. Retrieved 27 July 2021. ^ "Arm A profile architecture update 2019". community.arm.com. 25 September 2019. Retrieved 26 September 2019. ^ "LLVM 11.0.0 Release Notes". releases.llvm.org. Retrieved 11 March 2021. ^ "BFloat16 extensions for ARMv8-A". community.arm.com. 29 August 2019. Retrieved 30 August 2019. ^ Weidmann, Martin (21 September 2020). "Arm A-Profile Architecture Developments 2020". community.arm.com. ARM. Retrieved 28 September 2022. ^ "Scalable Matrix Extension for the ARMv9-A Architecture". community.arm.com. 14 July 2021. Retrieved 27 July 2021. ^ Weidmann, Martin (8 September 2021). "Arm A-Profile Architecture Developments 2021". community.arm.com. ARM. Retrieved 28 September 2022. ^ "What is New in LLVM 15? - Architectures and Processors blog - Arm Community blogs - Arm Community". 27 February 2023. Retrieved 15 April 2023. ^ "Arm A-Profile Architecture Developments 2022 - Architectures and Processors blog - Arm Community blogs - Arm Community". community.arm.com. 29 September 2022. Retrieved 9 December 2022. ^ "Arm A-Profile Architecture Developments 2023 - Architectures and Processors blog - Arm Community blogs - Arm Community". community.arm.com. 5 October 2023. Retrieved 14 October 2024. ^ "Arm A-Profile Architecture Developments 2024 - Architectures and Processors blog - Arm Community blogs - Arm Community". community.arm.com. 1 October 2024. Retrieved 14 October 2024. ^ a b Frumusanu, Andrei (3 September 2020). "ARM Announced Cortex-R82: First 64-bit Real Time Processor". AnandTech. ^ "Arm Architecture Reference Manual Supplement - Armv8, for Armv8-R AArch64 architecture profile". Arm Ltd. ^ "Cortex-R82 Technical Reference Manual". Arm Developer Retrieved from "