230410.Arm

Created: 2023-04-26 23:54:07

Questions

- Types of ARM CPU
  - What are types of ARM CPU? Please also describe them in a table with 32 bits and 64 bits.`(2023/4/11)` `(Arm.0410.md)`
  - Does Armv8-A has 32 bits? `(2023/4/11)` `(Arm.0410.md)`
  - Does Armv7-A has 64 bits? `(2023/4/11)` `(Arm.0410.md)`
  - Can I get Armv8-A with only 32-bit for cost down? `(2023/4/12)` `(Arm.0410.md)`
  - Below ariticle “In aarch32 state, what is the mechanism to switch to aarch64 in software?” come from `(2023/4/13)` `(Arm.0410.md)`
  - How do I specify only AArch32 of Armv8-A at run-time? `(2023/4/13)` `(Arm.0410.md)`
  - Is uboot 32-bit or 64-bit? `(2023/4/13)` `(Arm.0410.md)`
- TF-A
  - Which types of ARM cores is supported by TF-A? `(2023/4/11)` `(FF-A.0409.md)`
  - Must AArch64 have TF-A? `(2023/4/11)` `(FF-A.0409.md)`
- TrustZone
  - Which types of ARM CPU have TrustZone? `(2023/4/10)` `(TrustZone.0329.md)`
  - Does ARM Cortex-R series have TrustZone? `(2023/4/10)` `(TrustZone.0329.md)`
- OP-TEE
  - Does OP-TEE support ARMv7 and ARMv8? `(OP-TEE.0327.md)`

Q & A

Types of ARM CPU

Question: What are types of ARM CPU? Please also describe them in a table with 32 bits and 64 bits.`(2023/4/11)` `(Arm.0410.md)`

Answer:

ARM CPUs come in a variety of architectures and designs, each tailored to specific needs and applications. Some common types of ARM CPUs are Cortex-A, Cortex-R, and Cortex-M series. Here’s a table describing these three series along with their support for 32-bit and 64-bit execution states:

Series	Target Applications	Description	32-bit	64-bit
Cortex-A	High-performance systems	Designed for high-performance devices like smartphones, tablets, and other consumer electronics, as well as servers. These processors provide high performance and power efficiency, suitable for running complex operating systems like Linux, Android, and iOS.	Yes (Armv7-A)	Yes (Armv8-A)
Cortex-R	Real-time systems	Targeted towards real-time systems that require fast, deterministic response times like automotive electronics, industrial control systems, and medical devices. These processors prioritize reliability, fault tolerance, and real-time performance.	Yes (Armv7-R)	No*
Cortex-M	Low-power microcontrollers	Optimized for low-power, cost-sensitive applications like IoT devices, wearables, and other embedded systems. These processors are designed to offer low-latency response times, energy efficiency, and ease of use.	Yes (Armv6-M, Armv7-M, Armv8-M Baseline)	Yes (Armv8-M Mainline)

Please note that within each series, there are multiple specific processor models with different performance levels, features, and target applications. The examples provided above are not exhaustive.

*As of my knowledge cutoff in September 2021, Cortex-R series does not have 64-bit variants. However, it's essential to check ARM's official resources for any updates or new additions to the Cortex-R series.

Question: Does Armv8-A has 32 bits? `(2023/4/11)` `(Arm.0410.md)`

Answer:

Yes, Armv8-A architecture includes both 64-bit (AArch64) and 32-bit (AArch32) execution states. The 32-bit execution state in Armv8-A is largely compatible with the Armv7-A architecture, which allows for backward compatibility with existing 32-bit software.

In AArch32 state, Armv8-A processors support the ARM and Thumb instruction sets, which are commonly used in previous 32-bit Arm architectures. This dual execution state design allows Armv8-A based systems to run a mix of 32-bit and 64-bit applications and operating systems, providing flexibility and a smooth transition for software ecosystems.

Question: Does Armv7-A has 64 bits? `(2023/4/11)` `(Arm.0410.md)`

Answer:

No, Armv7-A architecture does not have 64-bit support. Armv7-A is a 32-bit architecture that was used in many smartphones, tablets, and other devices before the introduction of Armv8-A. Armv7-A supports the ARM and Thumb instruction sets and includes features like NEON SIMD (Single Instruction, Multiple Data) and hardware floating-point support.

Armv8-A is the successor to Armv7-A and introduced the 64-bit execution state (AArch64) alongside the 32-bit execution state (AArch32) for backward compatibility with Armv7-A software. The 64-bit support in Armv8-A allows for improved performance, larger addressable memory, and additional architectural enhancements compared to Armv7-A.

Question: Can I get Armv8-A with only 32-bit for cost down? `(2023/4/12)` `(Arm.0410.md)`

Answer:

While the Armv8-A architecture includes both 64-bit (AArch64) and 32-bit (AArch32) execution states, there is no specific variant of Armv8-A that exclusively supports 32-bit operation for cost reduction. The idea behind Armv8-A's dual execution states is to provide compatibility with existing 32-bit software while taking advantage of 64-bit enhancements.

If you are looking for a lower-cost, 32-bit ARM processor option, you may want to consider Cortex-M or Cortex-R series processors. These processors are designed for cost-sensitive, low-power, and real-time applications, respectively.

Cortex-M series processors, in particular, are widely used in microcontroller and Internet of Things (IoT) applications, where power efficiency and lower costs are essential. Cortex-R series processors cater to real-time systems that require fast, deterministic response times and high reliability. While both of these processor families are primarily 32-bit, the Armv8-M architecture for Cortex-M series also introduced a 64-bit variant (Armv8-M Mainline).

Question: Below ariticle “In aarch32 state, what is the mechanism to switch to aarch64 in software?” come from `(2023/4/13)` `(Arm.0410.md)`

Answer:

The discussion in the linked forum post provides a deeper insight into how execution states can be switched between AArch32 and AArch64 in ARMv8-A architecture.

To summarize the key points from the discussion:

1. Any exception type in AArch32 state could potentially lead to the execution state changing to AArch64. However, no exception taken while in AArch64 will result in the execution state changing to AArch32.
2. For exception returns, an exception return in AArch64 might cause the execution state to change to AArch32. No exception return in AArch32 can lead to the execution state changing to AArch64, except indirectly through triggering another exception that causes the switch.
3. A change of execution state can only occur if there is also a change in Exception Level (EL). For example, an exception from EL0 to EL1 could lead to a change in the execution state, but an exception from EL1 to EL1 could not.
4. The ARMv8-A architecture manual provides pseudocode for handling various exceptions and switching between AArch32 and AArch64 states. The manual can be referred to for understanding the specific conditions for each type of exception.

The exact mechanism for switching between AArch32 and AArch64 execution states depends on the exception type and the system configuration, and can be found in the ARMv8-A architecture manual.

Question: How do I specify only AArch32 of Armv8-A at run-time?`(2023/4/13)` `(Arm.0410.md)`

Answer:

In ARMv8-A architecture, you can use the Execution state (AArch32 or AArch64) depending on the current Exception Level (EL) and the settings of specific configuration registers. Here’s how you can ensure the system operates in AArch32 mode at runtime:

1. Configure the relevant registers: - If you are in the Secure state, you need to set the `SCR_EL3.RW` bit to 0. - If you are in the Non-secure state, you need to set the `HCR_EL2.RW` bit to 0.
2. When executing in the context of an OS or hypervisor, ensure that your software configures the registers correctly to maintain the AArch32 mode. This might include configuring the `SCTLR_ELx` register and other system registers as needed.
3. When running applications, ensure they are compiled for AArch32 mode. When an exception occurs (e.g., an interrupt or a system call), and the system transitions to a higher EL, it will remain in AArch32 mode if the settings mentioned in step 1 are properly configured.

By following these steps, you ensure that your ARMv8-A system operates in AArch32 mode at runtime. Keep in mind that, depending on your use case, certain performance and features available in AArch64 mode might not be accessible when operating in AArch32 mode.

Question: Is uboot 32-bit or 64-bit? `(2023/4/13)` `(Arm.0410.md)`

Answer:

U-Boot, or the Universal Bootloader, is a widely used open-source bootloader that can be built for various target platforms and architectures. It can be built for both 32-bit and 64-bit ARM architectures depending on the target system and its requirements.

When building U-Boot for a specific ARM platform, you can configure it to support either 32-bit (AArch32) or 64-bit (AArch64) execution states, depending on the underlying hardware and your requirements. If your target system is based on ARMv8-A or a newer architecture, you may choose to build U-Boot in 64-bit mode to take advantage of the features and performance improvements offered by the AArch64 execution state.

In summary, U-Boot can be built as either 32-bit or 64-bit, depending on the target platform and requirements. The specific execution state depends on the build configuration and the target hardware capabilities.

TF-A

Question: Which types of ARM cores is supported by TF-A? `(2023/4/11)` `(FF-A.0409.md)`

Answer:

Trusted Firmware-A (TF-A) supports Armv8-A and Armv7-A processors. The Armv8-A architecture encompasses both the 64-bit (AArch64) and the 32-bit (AArch32) execution states, while the Armv7-A architecture is a 32-bit (AArch32) execution state. TF-A provides a reference implementation of Secure World software for a variety of Arm Cortex-A cores within these architectures, including Cortex-A53, Cortex-A57, Cortex-A72, Cortex-A75, and Cortex-A76, among others.

Question: Must AArch64 have TF-A? `(2023/4/11)` `(FF-A.0409.md)`

Answer:

No, AArch64 does not require TF-A (Arm Trusted Firmware-A). TF-A is an open-source project that provides a reference implementation of the software components needed for the secure boot and runtime firmware on Armv8-A platforms. While it is commonly used and recommended for secure firmware development on Armv8-A platforms, it is not mandatory. You can use other firmware solutions or develop custom secure firmware for AArch64 platforms, depending on your requirements and platform constraints.

TrustZone

Question: Which types of ARM CPU have TrustZone? `(2023/4/10)` `(TrustZone.0329.md)`

Answer:

TrustZone technology is available in various ARM CPU cores, particularly in the ARM Cortex-A and ARM Cortex-M series.

1. ARM Cortex-A series: TrustZone is supported in many Cortex-A cores, including Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15, Cortex-A17, Cortex-A32, Cortex-A35, Cortex-A53, Cortex-A55, Cortex-A57, Cortex-A72, Cortex-A73, Cortex-A75, Cortex-A76, and Cortex-A77.
2. ARM Cortex-M series: The ARMv8-M architecture introduces TrustZone technology for Cortex-M processors. Examples of Cortex-M cores with TrustZone support include Cortex-M23 and Cortex-M33.

Keep in mind that ARM continues to develop new processor cores, and the list above might not be exhaustive. Always refer to the official ARM documentation for the most up-to-date information on specific cores and their features.

Question: Does ARM Cortex-R series have TrustZone? `(2023/4/10)` `(TrustZone.0329.md)`

Answer:

Yes, some ARM Cortex-R series processors also have TrustZone support. TrustZone for Cortex-R processors is designed for real-time systems that require high performance, safety, and security. One example of a Cortex-R processor with TrustZone support is the Cortex-R52.

As with other series, it’s essential to consult the official ARM documentation or the specific processor datasheet to verify TrustZone support for each Cortex-R processor.

OP-TEE

Question: Does OP-TEE support ARMv7 and ARMv8? `(OP-TEE.0327.md)`

Answer:

Yes, OP-TEE supports both ARMv7 and ARMv8 architectures. OP-TEE is designed to work with Arm TrustZone technology, which is available on both ARMv7 and ARMv8-based processors. Arm TrustZone provides hardware-based isolation between the normal world and the secure world, allowing OP-TEE to implement a Trusted Execution Environment (TEE) that can run trusted applications securely.

While ARMv7 and ARMv8 have some differences in terms of instruction sets and features, OP-TEE is compatible with both architectures and can be used to build secure systems on devices using either ARMv7 or ARMv8 processors.
